FIRST PART: NEURAL NETWORK PROJECT
Tratamiento de datos clínicos para la detección de ictus mediante redes neuronales artificiales

Autor: Javier Porcel Marí

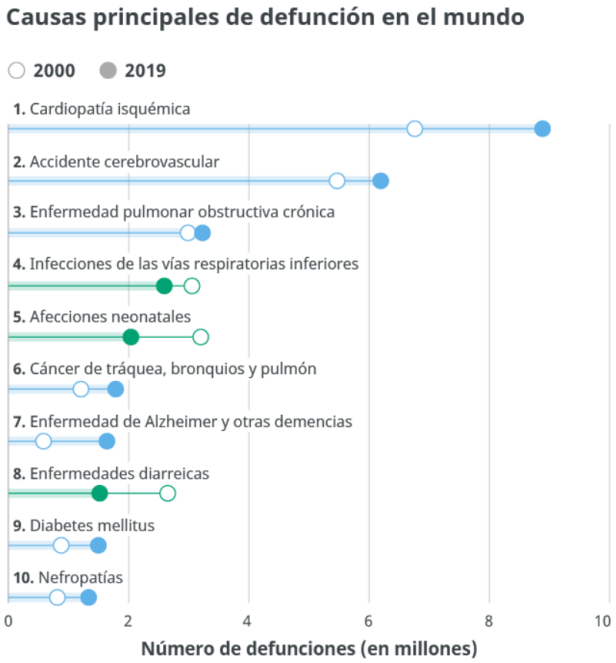
Abstract: En este trabajo se analizan variables de diferente tipología (edad, género, lugar de residencia, vida profesional, índice de masa corporal, hipertensión, nivel medio de glucosa en sangre…) con el fin de ser capaces de predecir mediante redes neuronales artificiales si un paciente ha sufrido o no un ictus o accidente cerebrovascular con anterioridad. Ser capaces de predecir si un paciente ha sufrido esta dolencia o no puede ser de gran utilidad ya no solo para poder detectar a las personas que lo han sufrido o que tienen altas probabilidades de sufrirlo sino también para entender mejor desde un punto de vista médico a esta terrible enfermedad que hoy en día es una de las principales causas de muerte natural en el mundo. De este modo se construirán cinco redes neuronales de veinticinco neuronas cada una y se compararán las tasas de acierto para quedarnos con aquella que tenga mayor efectividad.

Keywords: Ictus, Red Neuronal, Medicina

1. INTRODUCCIÓN

En la actualidad los accidentes cerebrovasculares son la segunda causa de muerte natural en el mundo según la Organización Mundial de la Salud [1]. Concretamente en el 2019 ha habido más de seis millones de defunciones causadas por accidentes cerebrovasculares tal y como se muestra en la siguiente gráfica:

Figura 1. Causas principales de defunción en el mundo según la OMS [1].



Por esto mismo es por lo que comprender mejor esta enfermedad y ser capaces tanto de prevenirla como de diagnosticarla a tiempo conllevaría un gran beneficio para la salud pública. De aquí surge la motivación de este trabajo que trata de predecir si un paciente ha sufrido de un ictus en algún momento de su vida mediante la técnica de inteligencia artificial por excelencia, la red neuronal. Para ello se analizará una base de datos [2], obtenida de la página web www.kaggle.com.

2. MATERIAL Y MÉTODOS

La base de datos utilizada se ha obtenido en la web www.kaggle.com y está compuesta por 5110 pacientes y por 12 variables. Concretamente las variables son las siguientes:

- "id": variable numérica que permite la identificación del paciente.
- "gender": variable categórica que muestra el género al que pertenece al paciente.
- "age": variable númerica que nos muestra la edad del paciente.
- "hypertension": variable categórica que nos muestra si el paciente sufre de hipertensión.
- "heart_desease": variable categórica que nos muestra si el paciente ha sufrido o sufre de algún problema cardiovascular.
- "ever_married": variable categórica que nos enseña si el paciente ha estado casado alguna vez.
- "work_type": variable categórica que denota qué tipo de profesión ejerce el paciente pudiendo clasificarse esta en cinco tipos: empleado por cuenta propia, empleado en la empresa privada, funcionario, cuida a sus hijos o nunca ha trabajado.
- "Residence_type": variable categórica que nos muestra si el paciente vive en un entorno rural o en un entorno urbano.
- "avg_glucose_level": variable numérica que denota el nivel medio de glucosa en sangre del paciente medido en mg/dL.
- "bmi": variable numérica asignada al índice de masa corporal del paciente.
- "smoking_status": variable categórica en la que se muestra si el paciente es o no es fumador, o si ha sido exfumador.
- "stroke": variable categórica que indica si el paciente ha sufrido un ictus con anterioridad. Es la variable que se va a tratar de predecir.

Para preparar la base de datos se han realizado diferentes operaciones mediante el software estadístico R, con el cual se programan todas las redes neuronales que se utilizan en este proyecto. En primer lugar se ha cargado el archivo .csv en el que se encuentra la base de datos mediante la librería readr [3] y se han eliminado de la base de datos aquellos pacientes que tenían valores faltantes en las variables "bmi", "smoking_status" y "gender". Posteriormente se han convertido las variables categóricas en variables dummy, además de que se ha eliminado la columna de la variable "id" ya que no es útil para la predicción. Finalmente se han eliminado algunos pacientes por ser observaciones atípicas (esto se ha observado mediante la realización de un PCA) las cuales nos iban a dificultar la clasificación y se ha eliminado la variable dummy "work_never" debido a que todos los valores de la columna eran ceros.

Al terminar este procesamiento de la base de datos se nos quedan 3411 pacientes con 16 variables, la mayoría de las cuales son variables dummy. Con la matriz de datos ya lista, se busca conseguir una muestra de datos de entrenamiento que corresponderá al 60% de los datos y una muestra de datos de entrenamiento con el 40% de los datos de manera que podamos validar las redes neuronales que se desarrollan en este trabajo mediante un hold out. Sin embargo al realizar este muestro hay que tener en cuenta que las clases están altamente desbalanceadas pues la mayoría de los pacientes de la base de datos no han sufrido un ictus por lo que se procede a crear una muestra de entrenamiento que esté balanceada. Para que ambas muestras, tanto entrenamiento como validación, estén preparadas es necesario escalar y centrar ambas muestras.

Una vez tenemos lista tanto la muestra de entrenamiento como la de validación se procede a ejecutar cinco redes neuronales que poseerán veinticinco neuronas cada una mediante la librería neuralnet [4]. La diferencia entre cada una de las redes neuronales radica en que están neuronas se encuentran distribuidas de diferente modo en las capas ocultas. Concretamente la primera red neuronal tiene las veinticinco neuronas en la capa oculta, la segunda tiene veinte neuronas en la primera capa oculta y cinco en la segunda, la tercera tiene quince neuronas en la primera capa oculta y diez en la segunda, la cuarta tiene diez neuronas en la primera capa oculta y quince en la segunda, y finalmente la quinta

tiene cinco neuronas en la primera capa oculta y veinte en la segunda. Para comprender mejor las diferencias entre las cinco redes neuronales se muestra la siguiente tabla explicativa:

Tabla 1. Configuración de cada red neuronal.

Numeración	Primera capa oculta	Segunda capa oculta
Red Neuronal 1	25 neuronas	0 neuronas
Red Neuronal 2	20 neuronas	5 neuronas
Red Neuronal 3	15 neuronas	10 neuronas
Red Neuronal 4	10 neuronas	15 neuronas
Red Neuronal 5	5 neuronas	20 neuronas

3. RESULTADOS

Tras ejecutar las redes neuronales se han obtenido estas gráficas al utilizar la librería ggplot 2 [5]:

Figura 2. Gráfico de la primera red neuronal.

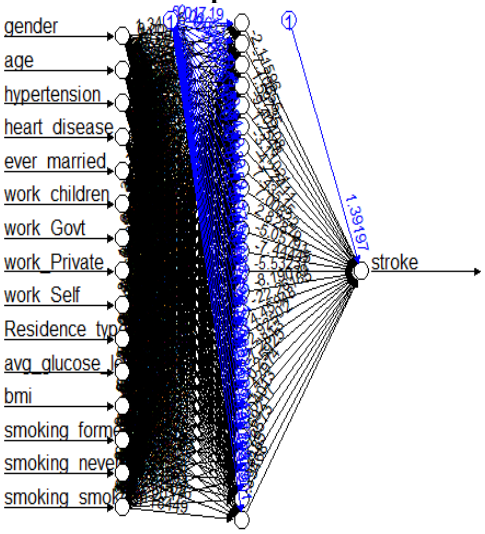


Figura 3. Gráfico de la segunda red neuronal.

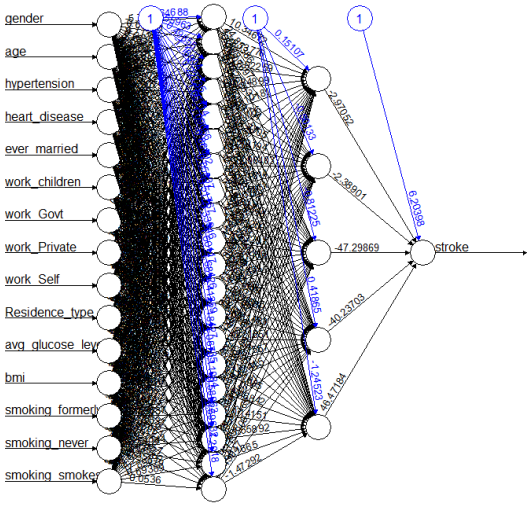


Figura 4. Gráfico de la tercera red neuronal.

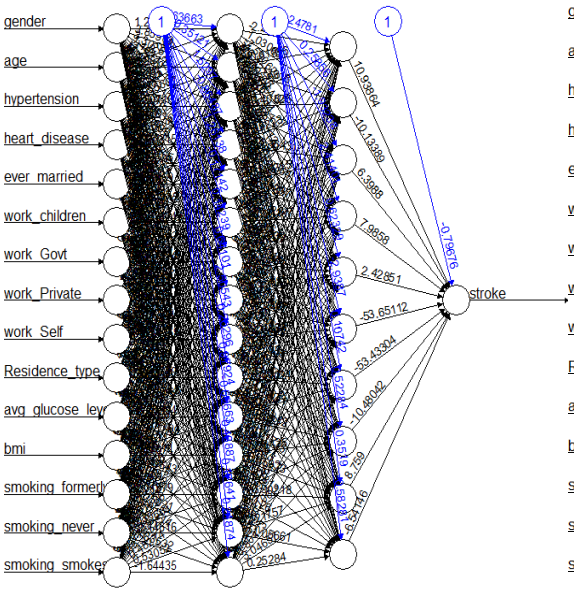


Figura 5. Gráfico de la cuarta red neuronal.

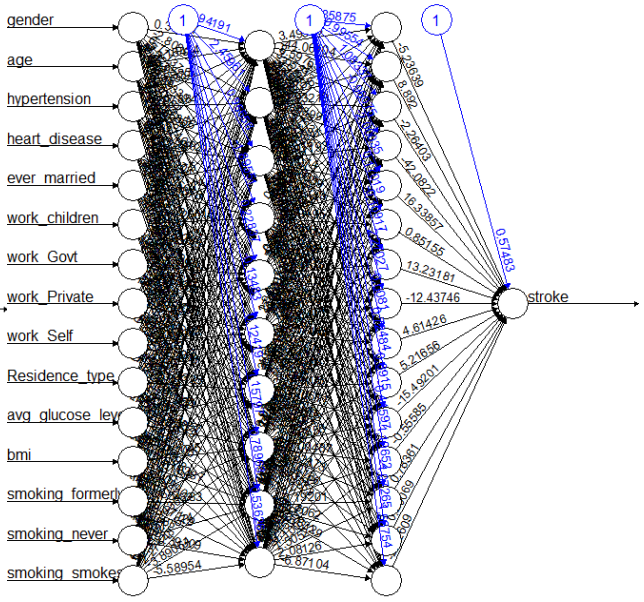
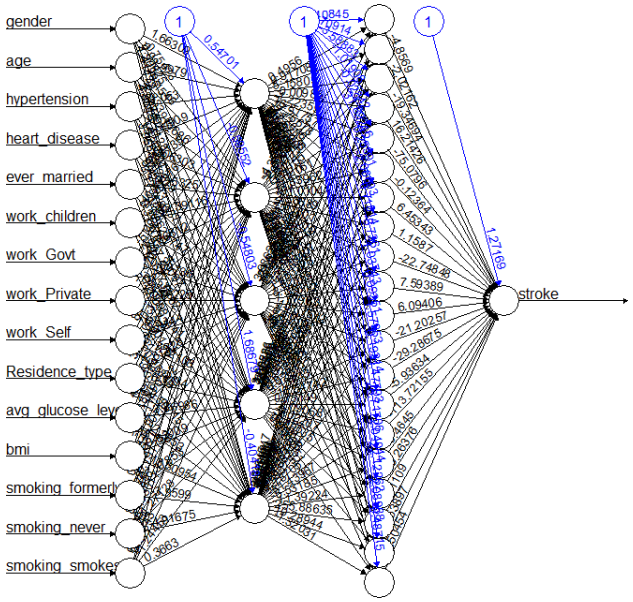


Figura 6. Gráfico de la quinta red neuronal.

Figura 6. Gráfico de la quinta red neuronal.



También se han obtenido las siguientes cinco matrices de confusión junto con las cinco tablas que nos muestran tanto la sensibilidad y la especificidad como la tasa de acierto para cada red neuronal:

Tabla 2. Matriz de confusión de la primera red neuronal.

Red neuronal 1	Valor real -> 0	Valor real -> 1
Valor predicho -> 0	1126	47
Valor predicho -> 1	174	18

Tabla 3. Resultado de la red neuronal 1.

Red neuronal 1	
Sensibilidad	27,69%
Especificidad	86,62%
Tasa de acierto	83,81%

Tabla 4. Matriz de confusión de la segunda red neuronal.

Red neuronal 2	Valor real -> 0	Valor real -> 1
Valor predicho -> 0	1163	43
Valor predicho -> 1	137	22

Tabla 5. Resultado de la red neuronal 2.

Red neuronal 2	
Sensibilidad	33,85%
Especificidad	89,46%
Tasa de acierto	86,81%

Tabla 6. Matriz de confusión de la tercera red neuronal.

Red neuronal 3	Valor real -> 0	Valor real -> 1
Valor predicho -> 0	1155	47
Valor predicho -> 1	145	18

Tabla 7. Resultado de la red neuronal 3.

Red neuronal 3	
Sensibilidad	27,69%
Especificidad	88,85%
Tasa de acierto	85,93%

Tabla 8. Matriz de confusión de la cuarta red neuronal.

Red neuronal 4	Valor real -> 0	Valor real -> 1
Valor predicho -> 0	1153	48
Valor predicho -> 1	147	17

Tabla 9. Resultado de la red neuronal 4.

Red neuronal 4	
Sensibilidad	26,15%
Especificidad	88,69%
Tasa de acierto	85,71%

Tabla 10. Matriz de confusión de la quinta red neuronal.

Red neuronal 5	Valor real -> 0	Valor real -> 1
Valor predicho -> 0	1082	36
Valor predicho -> 1	218	29

Tabla 11. Resultado de la red neuronal 5.

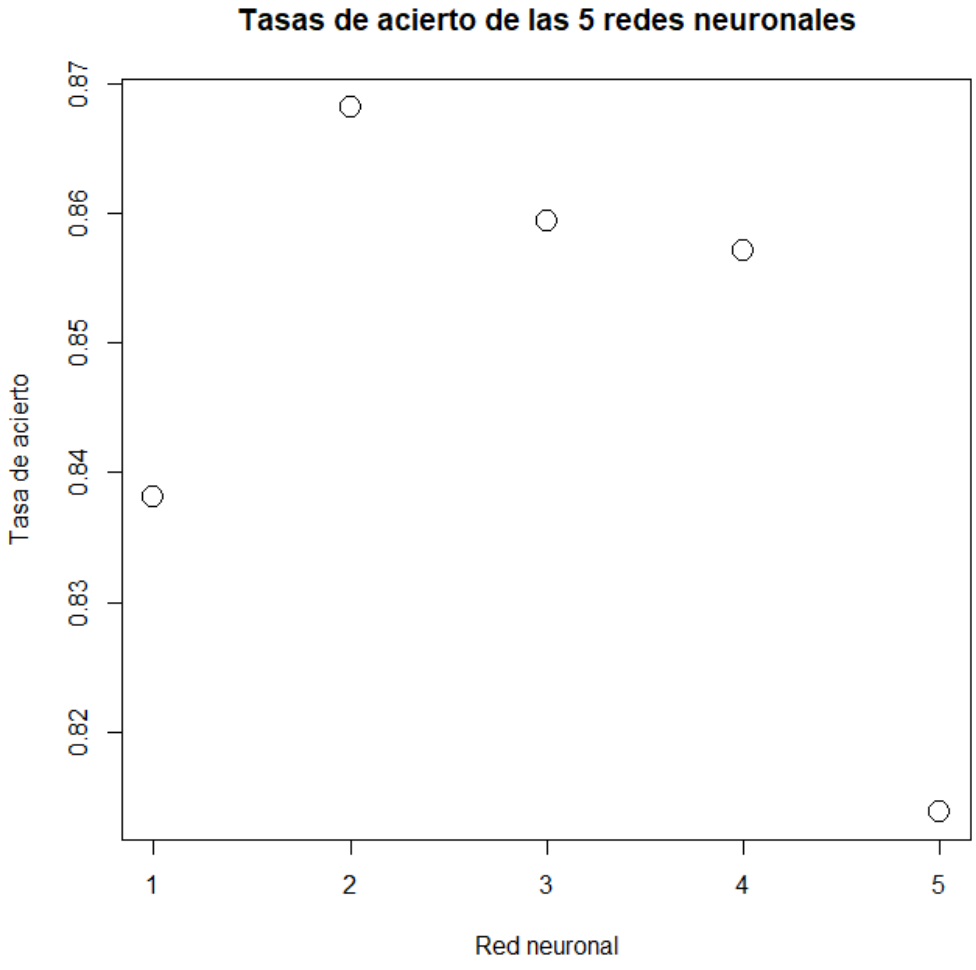
Red neuronal 5	
Sensibilidad	44,62%
Especificidad	83,23%
Tasa de acierto	81,39%

Tabla 12. Tasas de acierto para cada red neuronal.

Numeración	Tasa de acierto
Red Neuronal 1	83,81%
Red Neuronal 2	86,81%
Red Neuronal 3	85,93%
Red Neuronal 4	85,71%
Red Neuronal 5	81,39%

Como se observa en las tablas anteriores, la sensibilidad es muy baja en las cinco redes neuronales mientras que la especificidad es bastante alta. Esto significa que estas redes neuronales son bastante eficaces para acertar al clasificar a un paciente sin ictus como tal pero tiene bastantes dificultades para clasificar correctamente a un paciente con ictus. De todas maneras las cinco tasas de acierto superan el 80%. Para visualizar mejor los resultados se muestra la siguiente gráfica:

Figura 7. Gráfico que compara las tasas de acierto de las cinco redes neuronales.



En la gráfica se observa como el mejor modelo predictivo corresponde a la segunda red neuronal mientras que el peor modelo predictivo corresponde a la quinta red neuronal. Así pues en este caso, resulta óptimo colocar veinte neuronas en la primera capa oculta y otras cinco neuronas en la segunda capa oculta. Es curioso que a pesar de que la quinta red neuronal es la que menor tasa de acierto posee es la que mayor sensibilidad tiene. Es decir, hace un esfuerzo mayor por detectar casos positivos (mayor sensibilidad) pero esto acaba conllevando en que se equivoque más al predecir casos negativos.

4. CONCLUSION

Para concluir este trabajo cabe mencionar que se ha seleccionar la mejor red neuronal de veinticinco neuronas, de entre las propuestas, para clasificar a los pacientes según hayan tenido, o no, un ictus. Para ello se ha limpiado correctamente la base de datos, se han ejecutado los modelos propuestos y se han validado los resultados mediante un hold out obteniéndose así la mejor red neuronal. Concretamente la red neuronal que mejores resultados ha ofrecido ha sido aquella que trabaja con cinco neuronas en la primera capa oculta y veinte neuronas en la segunda ofreciendo una tasa de acierto del 86,81%. Sin embargo este resultado se puede mejorar pues, aunque posee una tasa de acierto tolerable, la sensibilidad de este modelo sigue siendo mejorable.

Referencias

[1] Organización Mundial de la Salud (9 de diciembre del 2020). *Las 10 principales causas de defunción*. https://www.who.int/es/news-room/fact-sheets/detail/the-top-10-causes-of-death

[2] Federico Soriano (Febrero del 2021). *Stroke Prediction Dataset*. https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

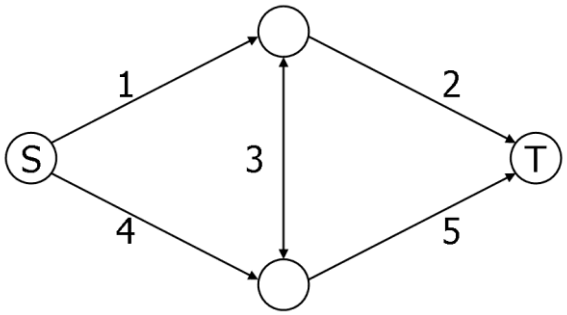
[3] Hadley Wickham, Jim Hester and Romain Francois (2018). readr: Read Rectangular Text Data. R package version 1.3.1. https://CRAN.R-project.org/package=readr

[4] Stefan Fritsch, Frauke Guenther and Marvin N. Wright (2019). neuralnet: Training of Neural Networks. R package version 1.44.2. https://CRAN.R-project.org/package=neuralnet

[5] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

SECOND PART: MONTE CARLO SIMULATION

1-Consideramos el sistema red aqui:



Número Arco i	*Probabilidad de falla P_i*
1	0.050
2	0.025
3	0.050
4	0.020
5	0.075

- Calcular la solución analítica para la probabilidad de falla de la red (= la probabilidad de que los nodos S y T no estén conectados).

Para calcular la solución analítica se realiza el siguiente cálculo a partir de los método de los cortes para analizar la fiabilidad en sistemas complejos teniendo en cuenta que existen cuatro cortes posibles (el corte de los tramos 1 y 4; el corte de los tramos 2 y 5; el corte de los tramos 1, 3 y 5; y finalmente el corte de los tramos 2,3 y 4):

$P_{fallo} = p_1p_4 + p_2p_5 + p_1p_3p_5 + p_2p_3p_4 - p_1p_2p_4p_5 - p_1p_3p_4p_5 - p_1p_2p_3p_4 - p_1p_2p_3p_5 - p_2p_3p_4p_5 - p_1p_2p_3p_4p_5 + p_1p_2p_3p_4p_5 + p_1p_2p_3p_4p_5 - p_1p_2p_3p_4p_5 = p_1p_4 + p_2p_5 + p_1p_3p_5 + p_2p_3p_4 - p_1p_2p_4p_5 - p_1p_3p_4p_5 - p_1p_2p_3p_4 - p_1p_2p_3p_5 - p_2p_3p_4p_5 + p_1p_2p_3p_4p_5 = 0{,}003074156$

$R_{red} = 1 - 0{,}003074156 = 0{,}996925844$

La probabilidad de falla de la red es del 0,3074156%.

- Escribir el código y repetir el cálculo con simulación Monte Carlo.

Utilizando el software R se introduce el siguiente código:

```
estado <- numeric(200000)+1
camino1 <- 0
camino2 <- 0
camino3 <- 0
camino4 <- 0
camino5 <- 0
i <- 0
for (i in 1:200000) {
  camino1 <- sample(1:1000, 1)/1000
  camino2 <- sample(1:1000, 1)/1000
```

```
  camino3 <- sample(1:1000, 1)/1000
  camino4 <- sample(1:1000, 1)/1000
  camino5 <- sample(1:1000, 1)/1000
  if (camino1 > 0.05) {
    camino1 <- 1
  } else {
    camino1 <- 0
  }
  if (camino2 > 0.025) {
    camino2 <-1
  } else {
    camino2 <- 0
  }
  if (camino3 > 0.05) {
    camino3 <- 1
  } else {
    camino3 <- 0
  }
  if (camino4 > 0.02) {
    camino4 <- 1
  } else {
    camino4 <- 0
  }
  if (camino5 > 0.075) {
    camino5 <- 1
  } else {
    camino5 <- 0
  }
  if (camino1 == 0 && camino4 == 0){
    estado[i] <- 0
  }
  if (camino2 == 0 && camino5 == 0){
    estado[i] <- 0
  }
  if (camino1 == 0 && camino3 == 0 && camino5 == 0){
    estado[i] <- 0
  }
  if (camino2 == 0 && camino3 == 0 && camino4 == 0){
    estado[i] <- 0
  }
}

Fiabilidad <- mean(estado)*100
Probabilidad_de_fallo <- 100 – Fiabilidad
```

Se obtiene una probabilidad de fallo del 0.2985%. Este valor es extremadamente cercano al valor obtenido analíticamente.

- Escribir el código Monte Carlo para calcular y dibujar el grafico de la confiabilidad tiempo-dependiente R(t) de la red por todos los tiempo hasta el tiempo de misión Tm=300 horas, asumiendo las siguientes tasas de falla por los arcos:

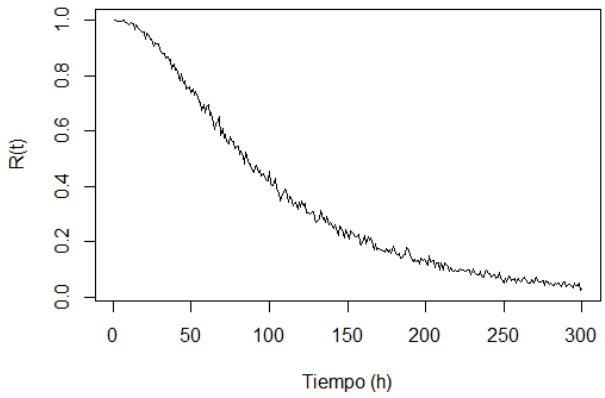
	1	2	3	4	5
l	$1\cdot 10^{-3}\ h^{-1}$	$2\cdot 10^{-2}\ h^{-1}$	$5\cdot 10^{-2}\ h^{-1}$	$1\cdot 10^{-2}\ h^{-1}$	$8\cdot 10^{-4}\ h^{-1}$

Para esta situación se ha programado el siguiente código con R:

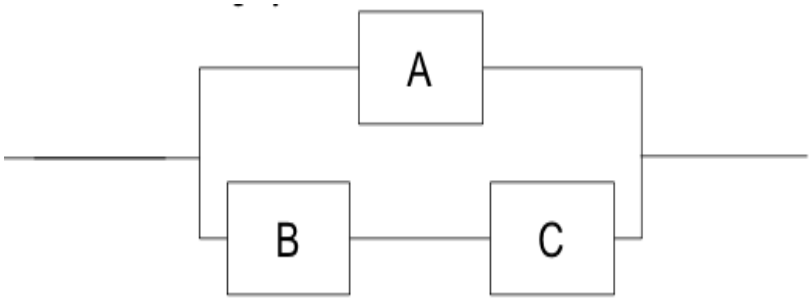
```
FiabilidadMatriz <- matrix(1, ncol=1000, nrow=300)
Fiabilidad1 <- 0
Fiabilidad2 <- 0
Fiabilidad3 <- 0
Fiabilidad4 <- 0
Fiabilidad5 <- 0
lambda1 <- 1e-3
lambda2 <- 2e-2
lambda3 <- 5e-2
lambda4 <- 1e-2
lambda5 <- 8e-4
i <- 0
t <- 0
for (t in 1:300) {
  for (i in 1:1000) {
    Fiabilidad1 <- exp(-lambda1*t)
    Fiabilidad2 <- exp(-lambda2*t)
    Fiabilidad3 <- exp(-lambda3*t)
    Fiabilidad4 <- exp(-lambda4*t)
    Fiabilidad5 <- exp(-lambda5*t)
    camino1 <- sample(1:1000, 1)/1000
    camino2 <- sample(1:1000, 1)/1000
    camino3 <- sample(1:1000, 1)/1000
    camino4 <- sample(1:1000, 1)/1000
    camino5 <- sample(1:1000, 1)/1000
    if (camino1 <= Fiabilidad1) {
      camino1 <- 1
    } else {
      camino1 <- 0
    }
    if (camino2 <= Fiabilidad2) {
      camino2 <- 1
    } else {
      camino2 <- 0
    }
    if (camino3 <= Fiabilidad3) {
      camino3 <- 1
```

```
    } else {
      camino3 <- 0
    }
    if (camino4 <= Fiabilidad4) {
      camino4 <- 1
    } else {
      camino4 <- 0
    }
    if (camino5 <= Fiabilidad5) {
      camino5 <- 1
    } else {
      camino5 <- 0
    }
    if (camino1 == 0 && camino4 == 0){
      FiabilidadMatriz[t, i] <- 0
    }
    if (camino2 == 0 && camino5 == 0){
      FiabilidadMatriz[t, i] <- 0
    }
    if (camino1 == 0 && camino3 == 0 && camino5 == 0){
      FiabilidadMatriz[t, i] <- 0
    }
    if (camino2 == 0 && camino3 == 0 && camino4 == 0){
      FiabilidadMatriz[t, i] <- 0
    }
  }
}
Media <- 1:300
for (j in 1:300){
  Media[j] <- mean(FiabilidadMatriz[j,])
}
plot(1:300, Media, xlab = "Tiempo (h)", ylab = "R(t)", type ="l")
```

Obteniéndose la siguiente gráfica que representa la fiabilidad en función del tiempo (valiendo R(300)=3,9%):



Consideramos el sistema:



Initial \ Arrival	1	2	3
1(nominal)	0	$\lambda_{1\to2}^{A(B,C)}$	$\lambda_{1\to3}^{A(B,C)}$
2 (degraded)	0	0	$\lambda_{2\to3}^{A(B,C)}$
3 (failed)	$\lambda_{3\to1}^{A(B,C)}$	$\lambda_{3\to2}^{A(B,C)}$	0

A	1	2	3
1	-	$3\ 10^{-3}$	10^{-3}
2	-	-	$6\ 10^{-3}$
3	$8\ 10^{-3}$	$5\ 10^{-3}$	-

B	1	2	3
1	-	$1\ 10^{-3}$	$5\ 10^{-3}$
2	-	-	$4\ 10^{-3}$
3	$7.5\ 10^{-3}$	$3.5\ 10^{-3}$	-

C	1	2	3
1	-	$8\ 10^{-3}$	$2.5\ 10^{-3}$
2	-	-	$2\ 10^{-3}$
3	$4\ 10^{-3}$	$1.5\ 10^{-3}$	-

- Sobre un tiempo de misión T_{miss} = 1000, calcular la solución analítica para:
  - la confiabilidad en función del tiempo t
  - la confiabilidad a t = T_{miss}
  - la indisponibilidad instantánea del componente

La confiabilidad en función del tiempo, R(t), se obtiene del siguiente modo:

$$p_A^{(t)}=p_A^{(0)}*T_A{}^t=(1,0,0)*T_A{}^t=(p_{1,A};p_{2,A};p_{3,A})$$
$$p_B^{(t)}=p_B^{(0)}*T_B{}^t=(1,0,0)*T_B{}^t=(p_{1,B};p_{2,B};p_{3,B})$$
$$p_C^{(t)}=p_C^{(0)}*T_C{}^t=(1,0,0)*T_C{}^t=(p_{1,C};p_{2,C};p_{3,C})$$

donde $p_A^{(t)}$, $p_B^{(t)}$, $p_C^{(t)}$ son tres vectores cuyas componentes corresponde a la probabilidad para las componentes A,B y C se encuentren en el estado 1,2 y 3 en el tiempo t. Además $p_A^{(0)}$, $p_B^{(0)}$, $p_C^{(0)}$ corresponden a los estados iniciales de las tres componentes y T_A, T_B y T_C a sus matrices de Markov.

A partir de aquí, se realiza el siguiente cálculo:

$$R(t)=P(\text{A funcione})+P(\text{B y C funcionen}) - P(\text{A,B y C funcionen})$$

$$R(t)=(p_{1,A}+p_{2,A})+(p_{1,B}+p_{2,B})*(p_{1,C}+p_{2,C})- (p_{1,A}+p_{2,A})*(p_{1,B}+p_{2,B})*(p_{1,C}+p_{2,C})$$

Y así es como se puede obtener el valor de R para cualquier tiempo.

Mediante esta metodología analítica se puede obtener R(t=Tmiss=1000) con el siguiente código de R donde se implementa los cálculos expuestos anteriormente:

```
lambdaA <- matrix(0, nrow=3, ncol=3)
lambdaA[1,2] <- 3e-3
lambdaA[1,3] <- 1e-3
lambdaA[1,1] <- 1 - lambdaA[1,2] - lambdaA[1,3]
lambdaA[2,3] <- 6e-3
lambdaA[2,2] <- 1 - lambdaA[2,3]
lambdaA[3,1] <- 8e-3
lambdaA[3,2] <- 5e-3
lambdaA[3,3] <- 1 - lambdaA[3,1] - lambdaA[3,2]

lambdaB <- matrix(0, nrow=3, ncol=3)
lambdaB[1,2] <- 1e-3
lambdaB[1,3] <- 5e-3
lambdaB[1,1] <- 1 - lambdaB[1,2] - lambdaB[1,3]
lambdaB[2,3] <- 4e-3
lambdaB[2,2] <- 1 - lambdaB[2,3]
lambdaB[3,1] <- 7.5e-3
lambdaB[3,2] <- 3.5e-3
lambdaB[3,3] <- 1 - lambdaB[3,1] - lambdaB[3,2]

lambdaC <- matrix(0, nrow=3, ncol=3)
lambdaC[1,2] <- 8e-3
lambdaC[1,3] <- 2.5e-3
lambdaC[1,1] <- 1 - lambdaC[1,2] - lambdaC[1,3]
lambdaC[2,3] <- 2e-3
lambdaC[2,2] <- 1 - lambdaC[2,3]
lambdaC[3,1] <- 4e-3
lambdaC[3,2] <- 1.5e-3
lambdaC[3,3] <- 1 - lambdaC[3,1] - lambdaC[3,2]

lambda1<-diag(3)
lambda2<-diag(3)
lambda3<-diag(3)
library(Biodem)
for (t in 1:1000){
  lambda1<-lambda1%*%lambdaA
  lambda2<-lambda2%*%lambdaB
  lambda3<-lambda3%*%lambdaC
  pA <- c(1,0,0)%*%lambda1
  pB <- c(1,0,0)%*%lambda2
  pC <- c(1,0,0)%*%lambda3
```

```
  Fiabilidad[t] <- (pA[1]+pA[2])+(pB[1]+pB[2])*(pC[1]+pC[2])-
(pA[1]+pA[2])*(pB[1]+pB[2])*(pC[1]+pC[2])
}
Fiabilidad[1000]
```

Y se obtiene R(1000)= 0.8997311.

Finalmente para calcular la fiabilidad instantánea de cada componente en t=1000 se realiza el siguiente cálculo:

$$p_A^{(t)}=p_A^{(0)}*T_A^t=(1,0,0)*T_A^t=(p_{1,A};p_{2,A};p_{3,A})=(0.4141444, 0.3791114, 0.2067443)$$
$$p_B^{(t)}=p_B^{(0)}*T_B^t=(1,0,0)*T_B^t=(p_{1,B};p_{2,B};p_{3,B})=(0.3656539, 0.3432305, 0.2911156)$$
$$p_C^{(t)}=p_C^{(0)}*T_C^t=(1,0,0)*T_C^t=(p_{1,C};p_{2,C};p_{3,C})=(0.1041141\ 0.6223933\ 0.2734926)$$

Así pues, la fiabilidad de cada componente para t = 1000 es:

$$R_A(t=1000)=(p_{1,A}+p_{2,A})/(p_{1,A}+p_{2,A}+p_{3,A})=0.7932557=79.32557\%$$
$$R_B(t=1000)=(p_{1,B}+p_{2,B})/(p_{1,B}+p_{2,B}+p_{3,B})=0.7088844=70.88844\%$$
$$R_C(t=1000)=(p_{1,C}+p_{2,C})/(p_{1,C}+p_{2,C}+p_{3,C})=0.7265074=72.65074\%$$

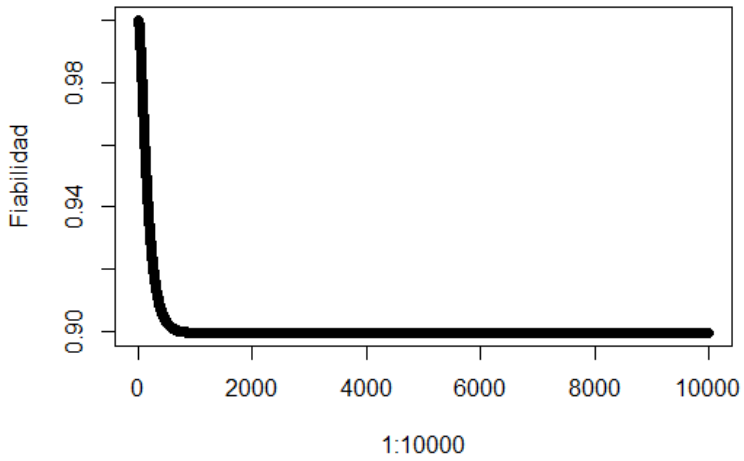
- Calcular la indisponibilidad asintótica

Para calcular la disponibilidad asintótica habría que calcular $R(t\to\infty)$. Como esto no es posible se realiza el cálculo para t = 10000, valor bastante elevado y como se observa en la gráfica extremadamente cerca de la asíntota.

Mediante el siguiente código se calcula (utilizando las matrices definidas anteriormente):

```
for (t in 1:10000){
  lambda1<-lambda1%*%lambdaA
  lambda2<-lambda2%*%lambdaB
  lambda3<-lambda3%*%lambdaC
  pA <- c(1,0,0)%*%lambda1
  pB <- c(1,0,0)%*%lambda2
  pC <- c(1,0,0)%*%lambda3
  Fiabilidad[t] <- (pA[1]+pA[2])+(pB[1]+pB[2])*(pC[1]+pC[2])-
(pA[1]+pA[2])*(pB[1]+pB[2])*(pC[1]+pC[2])
}
Fiabilidad[1000]
plot(1:10000, Fiabilidad)
```

Se obtiene R(t=10000)= 0.8997311. Extremadamente cerca de la asíntota como se observa:



- Escribir el código y repetir todos los cálculos con simulación Monte Carlo

Para realizar la simulación de Montecarlo se utiliza el siguiente código:

```
lambdaA <- matrix(0, nrow=3, ncol=3)
lambdaA[1,2] <- 3e-3
lambdaA[1,3] <- 1e-3
lambdaA[2,3] <- 6e-3
lambdaA[3,1] <- 8e-3
lambdaA[3,2] <- 5e-3

lambdaB <- matrix(0, nrow=3, ncol=3)
lambdaB[1,2] <- 1e-3
lambdaB[1,3] <- 5e-3
lambdaB[2,3] <- 4e-3
lambdaB[3,1] <- 7.5e-3
lambdaB[3,2] <- 3.5e-3

lambdaC <- matrix(0, nrow=3, ncol=3)
lambdaC[1,2] <- 8e-3
lambdaC[1,3] <- 2.5e-3
lambdaC[2,3] <- 2e-3
lambdaC[3,1] <- 4e-3
lambdaC[3,2] <- 1.5e-3

lambdaA123 = c(lambdaA[1,2]+lambdaA[1,3],lambdaA[2,3],
lambdaA[3,1]+lambdaA[3,2])
lambdaB123 = c(lambdaB[1,2]+lambdaB[1,3],lambdaB[2,3],
lambdaB[3,1]+lambdaB[3,2])
lambdaC123 = c(lambdaC[1,2]+lambdaC[1,3],lambdaC[2,3],
lambdaC[3,1]+lambdaC[3,2])

Tmiss <- 1000
Tfallo <- 0
```

```
TfalloA <- 0
TfalloB <- 0
TfalloC <- 0

for (i in 1:10000){
  estadoactual <- c(1,1,1)
  t <- 0
  while (t<=Tmiss){
    Tasa_a <- lambdaA123[estadoactual[1]]
    Tasa_b <- lambdaB123[estadoactual[2]]
    Tasa_c <- lambdaC123[estadoactual[3]]
    Tasa_Sistema <- Tasa_a + Tasa_b + Tasa_c
    Rt = sample(1:1000, 1)/1000
    Rc = sample(1:1000, 1)/1000
    Rs = sample(1:1000, 1)/1000
    t1 <- t - log(1-Rt)/Tasa_Sistema
    if (estadoactual[1]==3 && (estadoactual[2]==3 || estadoactual[3]==3)){
      if (t1 < Tmiss){
        Tfallo <- Tfallo + t1 - t
      } else {
        Tfallo <- Tfallo + Tmiss - t
      }
    }
    if (estadoactual[1]==3){
      if (t1 < Tmiss){
        TfalloA <- TfalloA + t1 - t
      } else {
        TfalloA <- TfalloA + Tmiss - t
      }
    }
    if (estadoactual[2]==3){
      if (t1 < Tmiss){
        TfalloB <- TfalloB + t1 - t
      } else {
        TfalloB <- TfalloB + Tmiss - t
      }
    }
    if (estadoactual[3]==3){
      if (t1 < Tmiss){
        TfalloC <- TfalloC + t1 - t
      } else {
        TfalloC <- TfalloC + Tmiss - t
      }
    }
    if (Rc < Tasa_a/Tasa_Sistema){
      if (Rs < lambdaA[estadoactual[1], 1]/lambdaA123[estadoactual[1]]){
        estadoactual[1]<-1
      } else if (Rs < (lambdaA[estadoactual[1], 1]+lambdaA[estadoactual[1],
2])/lambdaA123[estadoactual[1]]){
        estadoactual[1]<-2
```

```
      } else {
        estadoactual[1]<-3
      }
    } else if (Rc < (Tasa_a + Tasa_b)/Tasa_Sistema){
        if (Rs < lambdaB[estadoactual[2], 1]/lambdaB123[estadoactual[2]]){
          estadoactual[2]<-1
      } else if (Rs < (lambdaB[estadoactual[2], 1]+lambdaB[estadoactual[2],
2])/lambdaB123[estadoactual[2]]){
          estadoactual[2]<-2
      } else {
        estadoactual[2]<-3
      }
    } else {
      if (Rs < lambdaC[estadoactual[3], 1]/lambdaC123[estadoactual[3]]){
        estadoactual[3]<-1
      } else if (Rs < (lambdaC[estadoactual[3], 1]+lambdaC[estadoactual[3],
2])/lambdaC123[estadoactual[3]]){
        estadoactual[3]<-2
      } else {
        estadoactual[3]<-3
      }
    }
    t <- t1
  }
}
P <- Tfallo / (10000*Tmiss)
1-P
```

Obteniéndose R(t=1000=Tmiss)= 0.9181727=91.181727%, valor muy cercano al obtenido mediante el cálculo analítico.

Para cada componente se obtienen las siguientes fiabilidades:

$$R_A(t=1000)= 0.8248935=82.248935\%$$
$$R_B(t=1000)= 0.7218664=72.18664\%$$
$$R_C(t=1000)= 0.7549696=75.49696\%$$

Ligeramente más altas a las obtenidas analíticamente.

Para calcular el valor asintótico se calcula para t =10000 y se obtiene R(t=10000)= 0.8965275=89.65275%, valor muy cercano al obtenido anteriormente.