Estimación de un modelo lineal generalizado del salario mediante métodos numéricos

Taller de Simulación

Universidad de la República

Valentina Caldiroli, Maximiliano Saldaña

Noviembre 2020

Índice

1. Resumen ejecutivo

Se busca estimar los coeficientes de un modelo lineal generalizado del salario, siendo el contexto de la base de datos Estados Unidos en el año 1976. Se utiliza el método de máxima verosimilitud, empleando métodos numéricos de optimización. El foco está en la aplicación y comparación de algunos algoritmos de optimización, en particular el método del ascenso más rápido, el de Newton y el de Broyden-Fletcher-Goldfarg-Shanno. La comparación se hace en términos de velocidad (cantidad de iteraciones que toman los algoritmos en converger) y precisión respecto a replicar el resultado del método de mínimos cuadrados iterativamente re-ponderados empleado por el sofware R para la estimación de modelos lineales generalizados.

2. Presentación del problema

El objetivo planteado es estimar un modelo logístico mediante un método de máxima verosimilitud que aplique los métodos de optimización del curso y comparar el desempeño de nuestra estimación con el de las funciones base de R. Se hará uso de una base de salarios de Estados Unidos del año 1976, extraída de Wooldridge (2009). La variable a ser explicada es una indicadora que vale 1 cuando el salario de la persona es mayor o igual del promedio y 0 cuando es menor -sal_med-. Las variables explicativas (que se eligieron en base a lo ya trabajado en el curso Modelos Lineales) serán el sexo de la persona -$female$-, su educación -$educ$-, experiencia -$exper$-, antigüedad -$tenure$-, región (dividida en Norte -$northcen$-, Sur -$south$-, Oeste -$west$- y Este -categoría de referencia-), si vive en una región metropolitana -$reg.metro$-, rama de actividad (dividida en Construcción -$cosntruc$-, Comercio -$trade$-, Servicios -$services$- y Otros -categoría de referencia-) y Ocupación (dividida en Profesional -$profocc$-, Administrativos -$clerocc$-, Servicios -$servocc$- y Otros -categoría de referencia-).

El modelo en cuestión es de la forma siguiente *(Nalbarte, 2020)*:

$$P(Y_i = 1) = \frac{exp(x_i'\beta)}{1 + exp(x_i'\beta)} = \pi_i = E(y_i)$$

Donde se tiene que:

$$y_i = sal\_med_i$$

$$x_i'\beta = \beta_0 + \beta_1 female_i + \beta_2 educ_i + \beta_3 exper_i + \beta_4 tenure_i + \beta_5 northcen$$

$$+\beta_6 south_i + \beta_7 west_i + \beta_8 reg.metro + \beta_9 construc_i + \beta_{10} services_i$$

$$+\beta_{11} trade_i + \beta_{12} profocc_i + \beta_{13} clerocc_i + \beta_{14} servocc_i$$

La función de unión (la que vincula $E(y_i)$ con x_i') es:

$$log_e \left[\frac{\pi_i}{1 - \pi_i} \right] = exp(x_i'\beta)$$

El problema de estimar el vector de parámetros β se traduce en un problema de optimización, ya que buscamos que dicho vector sea el que maximice la función de verosimilitud de Y. Como Y_i son variables aleatorias Bernoulli independientes e idénticamente distribuidas $\forall\ i = 1, \ldots, n$ la función de verosimilitud es:

$$L(\pi_1, \ldots, \pi_n | Y_1, \ldots, Y_n) = \prod_{i=1}^{n} f_{Y_i}(y_i) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i^{y_i})^{1 - y_i}$$

Resulta equivalente trabajar con el logaritmo de la verosimilitud, que es:

$$log\, L(\pi_1, \ldots, \pi_n | Y_1, \ldots, Y_n) = log \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i^{y_i})^{1-y_i} = \sum_{i=1}^{n} \left[Y_i log \left(\frac{\pi}{1+\pi} \right) \right] + \sum_{i=1}^{n} log(1 - \pi_i)$$

Teniendo en cuenta que:

$$1 - \pi_i = [1 + exp(x_i' \beta)]^{-1}$$

$$\frac{\pi}{1+\pi} = exp(x_i' \beta)$$

$$log\, L(\beta | Y) = \sum_{i=1}^{n} Y_i (x_i' \beta) - \sum_{i=1}^{n} log(1 + exp(x_i' \beta))$$

Estimaremos el vector β que maximice la función anterior.

3. Metodología usada

Para lograr optimizar la función de verosimilitud, se emplearán tres métodos estudiados en el curso, denominados: Ascenso más rápido, de Newton y de quasi-Newton en particular el algoritmo de Broyden-Fletcher-Goldfarg-Shanno (BFGS).

El primer método, el del ascenso más rápido, es utilizado en funciones contínuas con derivadas parciales de primer y segundo orden; su implementación se da a partir de una búsqueda lineal mediante la recursión:

$$X_{n+1} = X_n + \alpha_n p_n$$

De donde se denota α_n a la longitud de paso con la que se realizará la búsqueda, y p_n es la dirección de búsqueda, determinada por $\nabla f(x)$ (vector gradiente), de modo que cuando la $||\nabla f(x)|| > 0$ la función crece, y una vez que $||\nabla f(x)|| = 0$ ese crecimiento se detiene, lo que indicaría un máximo (aunque dependiendo de la función puede ser un mínimo o un punto silla). Una vez definida la dirección del paso, se procede a la elección de la longitud, mediante:

$$\alpha_n = max_\alpha \phi(\alpha) = max_\alpha f(x_n + \alpha f(x_n))$$

Para una elección correcta del α además es necesario el cumplimiento de las condiciones de Wolfe:

- Condición del ascenso suficiente:

$$f(x_n + \alpha p_n) \geq f(x_n) + c_1 \alpha \nabla f(x_n)^T p_n$$

- Condición de curvatura:

$$\nabla f(x_n + \alpha_n p_n)^T p_n \leq c_2 \nabla f(x_n)^T p_n$$

La primera impone que los valores aceptables de α son aquellos donde $f(x_{n+1})$ está por encima de la recta con pendiente $c_1 \nabla f(x_n)^T p_n$. La segunda impone que el gradiente de $f(x_n + \alpha_n p_n)$ no sea demasiado "empinado" con respecto al gradiente de $f(x_n)$ (Álvarez y Massa, 2020). Una manera práctica de encontrar un posible valor de α es mediante el algoritmo de *backtracking*. Este algoritmo comienza con un valor inicial α_{max} y lo contrae iterativamente hasta que se cumplan las condiciones de Wolfe, con la posibilidad de expandir (hacer *forward-tracking*) en el caso de que no lo hagan.

El método de Newton por su parte, agiliza el algortimo del ascenso más rápido mediante la inclusión de la Hessiana en el mismo, utilizando un polinomio de Taylor de segundo orden llegamos a la equivalencia:

$$f(x_{n+1}) \approx f(x_n) - \nabla f(x_n)^T H_{f(x_n)} \nabla f(x_n)$$

Para que se cumpla que la función sea creciente, se debe cumplir que $\nabla f(x_n)^T H_{f(x_n)} \nabla f(x_n) \leq 0$, y esto solo sucede cuando $H_{f(x_n)}$ es definida negativa. Por lo que la recursión de Newton está definida de la siguiente manera:

$$x_{n+1} = x_n - \alpha_n H^{-1}_{f(x_n)} \nabla f(x_n)$$

Para la correcta utilización de este método se debe buscar un vector inicial x_0 próximo al vector que maximiza la función, o por lo menos utilizar uno que al evaluarlo en la hessiana de como resultado que esta sea definida negativa.

Los métodos de Cuasi-Newton solo necesitan de la función a maximizar y de su gradiente, que mediante las iteraciones va aproximando la hessiana, alcanzando un orden de convergencia superlineal. A pesar de que dicho orden es menor al cuadrático del de Newton, puede llegar a ser más veloz que este último método al no tener que evaluar la hessiana en cada paso. Los métodos de Cuasi-Newon usan una recursión de la forma:

$$x_{n+1} = x_n + \alpha_n B_n^{-1} \nabla f(x_n)$$

De donde B_n es la aproximación de la Hessiana. Su actualización se da por $B_{n+1}(x_{n+1} - x_n) = \nabla f(x_{n+1}) - \nabla f(x_n)$, que al definir $S_n = x_{n+1} - x_n$ y $y_n = \nabla f(x_{n+1}) - \nabla f(x_{n+1})$ se llega a la ecuación secante $B_{n+1} s_n = y_n$. Para actualización de la Hessiana existen varios métodos:

- Davidon

$$B_{n+1} = \left(I - \frac{1}{y_n^T s_n} y_n s_n^T\right) B_n \left(I - \frac{1}{y_n^T s_n} s_n y_n^T\right) + \frac{y_n y_n^T}{y_n^T s_n}$$

Entonces la inversa de de la matriz resultante se emplea en la recursión para obtener x_{n+1} a partir de x_n.

- Broyden-Fletcher-Goldfarg-Shanno (BFGS)

En vez de invertir el resultado de la actualización, este método surge de imponer la ecuación secante a la inversa de B_n. Esta recursión está dada por:

$$H_{n+1} = \left(I - \frac{1}{y_n^T s_n} s_n y_n^T\right) H_n \left(I - \frac{1}{y_n^T s_n} y_n s_n^T\right) + \frac{s_n s_n^T}{y_n^T s_n}$$

Nuevamente, la matriz resultante se emplea en la recursión para obtener x_{n+1} a partir de x_n.

Para seleccionar H_0 dos posibles alternativas son aproximar la hessiana en x_0 e invertirla (debería tomar menos iteraciones al usar una mejor aproximación) o utilizar la matriz opuesta a la identidad (para asegurar que las sucesivas aproximaciones sean definidas negativas).

4. Analisis descriptivo de los datos

La base cuenta con observaciones de 526 personas y con las siguientes 22 variables: Salario (promedio por hora, medido en dólares), Años de Educación, Años de Experiencia, Antigüedad, Raza (variable indicadora, 1 si la persona no es de raza blanca), Sexo, Estado Civil (vale 1 si la persona está casada), Número de Dependientes, Región Metropolitana (vale 1 si la persona vive en dicha región), Región (dividida en tres variales indicadoras: Norte, Sur y Oeste), Rama de Actividad (dividida en tres variables indicadoras: Construcción, Comercio y Servicios), Ocupación (tres variables indicadoras: Profesional, Administrativos y Servicios), el logaritmo de la variable Salario y los cuadrados de las variables Experiencia y Antigüedad. Se crea una variable que vale 1 cuando el salario de la persona está en el nivel promedio (5,896103) o por encima.

Se analizar como se distribuyen la población de acuerdo a las variables categóricas (Región, Rama de actividad, Ocupación, Raza, Sexo y Estado civil). Para las primeras cuatro variables buscamos saber cuantos individuos corresponden a cada rama, y aplicar medidas de resumen de la variable salario para cada categoría de las otras tres variables. Inicialmente se parte de ver la proporción de personas cuyo nivel está por debajo del nivel medio de salario y por encima.

4.1. Salario medio

Como antes se mencionó, el nivel medio del salario es 5,896103.

	sal_med	Proporción
1	Bajo el promedio	0.62
2	Sobre promedio	0.38

Cuadro 1: Proporción de personas con salarios por debajo y sobre el promedio

Más del 50 % de las personas tienen un salario por debajo del promedio, lo que es de esperarse dada la distribución asimétrica a rama izquierda conocida de la variable.

4.2. Región

Nos indica en qué región se ubican los individuos. Las categorías de esta variable son Norte, Sur, Este y Oeste. Tenemos 3 variables indicadoras para esta región, Norte-Centro, Sur y Oeste. Este es la categoría de referencia.

	Región	Cantidad	Mínimo	Media	Máximo
1	Norte	132	1.50	5.71	21.86
2	Sur	187	1.50	5.39	20.00
3	Este	118	1.43	6.37	24.98
4	Oeste	89	0.53	6.61	22.20
5	Todas	526	0.53	5.90	24.98

Cuadro 2: Cantidad de observaciones y salario medio, mínimo y máximo para las distintas regiones.

Se observa que la región Oeste tiene pocos individuos en comparación con las otras, sobre todo con la zona Sur que es la que contiene más individuos. En cuanto al salario, la media en las zonas Norte y Sur es menor que la media total de las observaciones. En cuanto a los máximos, se ve todos rondan entre 20 y 25 aproximadamente. Los mínimos también se asimilan entre sí, excepto la zona Oeste que tiene un mínimo de casi la tercera parte de los demás.

4.3. Rama de actividad

	Rama de Actividad	Cantidad	Mínimo	Media	Máximo
1	Construcción	24	3.00	5.96	17.71
2	Servicios	53	0.53	4.34	12.50
3	Comercio	151	1.43	4.79	21.86
4	Otros	298	1.50	6.73	24.98
5	Todas	526	0.53	5.90	24.98

Cuadro 3: Cantidad de observaciones y salario medio, mínimo y máximo para las distintas ramas de actividad.

Se observa una distribución muy desigual entre las cuatro opciones de respueta, pero en "Otros" es donde más observaciones hay, incluso más de la mitad. En cuanto al salario, observamos que Construcción tiene el mayor mínimo de las 4 opciones, con una media similar a la media total, pero un máximo un poco bajo. En el caso de Servicios, vemos que tiene el mínimo, la media y el máximo menor de todas. Comercio tiene una media similar a la de Servicios aunque su máximo y su mínimo es mayor.

4.4. Ocupación

	Ocupación	Cantidad	Mínimo	Media	Máximo
1	Servicio	74	0.53	3.59	7.81
2	Administrativos	88	2.65	4.74	12.50
3	Profesional	193	2.23	8.04	24.98
4	Otros	171	1.43	5.07	15.00
5	Todas	526	0.53	5.90	24.98

Cuadro 4: Cantidad de observaciones y salario medio, mínimo y máximo para las distintas ocupaciones.

Se observa que la media y el máximo de Profesionales es la mayor de todas, algo de esperarse, que una persona que se desempeña como profesional tenga un salario mayor. Para Administrativos se ve que tiene el mayor mínimo y el menor máximo, por lo que podríamos considerar que los salarios no varían tanto. Para los trabajadores del área de Servicios, vemos que tienen un mínimo, media y máximo menor, por lo que podemos decir que en promedio el salario es menor.

4.5. Otras variables categóricas

Con el resto de las variables econtramos que las proporciones entre las categorías son:

- Para la variable Sexo hay un 47.91 % de mujeres y 52.09 % de hombres. La proporción de la mitad cada sexo.
- Para la variable Estado Civil hay un 60.84 % de casados y 39.16 % de no casados.
- Para la variable Área Metropolitana hay un 72.24 % de personas que habitan en la misma y 27.76 % que no. Predomina bastante la población metropolitana.
- Para la variable Raza hay un 10.27 % de personas no blancas y 89.73 % blancas. El resultado es de esperarse dado que las personas no blancas son una minoría en los Estados Unidos.

5. Resultados

5.1. Estimando mediante las funciones base

Inicialmente para tener como referencia y para comparar, estimaremos el modelo mediante la función de R *glm()* del paquete base *stats*, la cual hace uso del método de estimación por mínimos cuadrados iterativamente re-ponderados (IWLS) (R Core Team, 2020). Estos resultados se visualizarán a la hora de comparar con los otros métodos. Cuando se busque analizar cuál método es más veloz se empleará el número de iteraciones que necesita para llegar a la convergencia, ya que el tiempo máquina depende de más factores que el propio algoritmo (la computadora que se está usando, los programas corriendo de fondo, etc.). Evaluando los coeficientes resultantes en la función de verosimilud el valor del máximo obtenido es -232,3441.

5.2. Estimación mediante los métodos vistos en clase

Retomando el planteo de nuestro problema, debemos buscar el máximo de nuestra función de verosimilitud, es decir:

$$\max_{\beta} \; log\, L(\beta|Y) = \max_{\beta} \; \left[\sum_{i=1}^{n} Y_i(x_i'\beta) - \sum_{i=1}^{n} log(1 + exp(x_i'\beta))\right]$$

Para emplear los métodos numéricos de optimización antes mencionados haremos uso de las funciones para aplicarlos vistas en el curso, las que fue necesario modificar para que fuera posible emplear aproximaciones numéricas del gradiente y la hessiana de la log-verosimilitud en vez de ser estas calculadas manualmente.

5.2.1. Ascenso más rápido

Para comenzar con este método y verificar que las funciones se desempeñan correctamente, probamos suministrándole como valores iniciales los estimados mediante la función *glm()*. A continuación, se presenta un cuadro comparando los valores de ambas estimaciones:

	glm()	Ascenso
(Intercept)	-4.442367437	-4.442367437
female	-1.194755469	-1.194755469
educ	0.277912495	0.277912308
exper	0.007568890	0.007568376
tenure	0.083587709	0.083587627
northcen	-0.012727340	-0.012727340
south	0.035935951	0.035935951
west	0.688820844	0.688820844
reg.metro	0.745975971	0.745975971
construc	-0.363769611	-0.363769611
services	-0.493415175	-0.493415175
trade	-1.429567604	-1.429567604
profocc	0.854540015	0.854540015
clerocc	-0.585495085	-0.585495085
servocc	-1.209387935	-1.209387935

Cuadro 5: Comparación de las estimaciones de glm() y el método del ascenso más rápido

Se puede apreciar, como era de esperarse, la similitud entre ambas estimaciones, difiriendo a lo sumo recién a partir del sexto valor después de la coma (se fijó una tolerancia de 10^{-5}). Toma solo una iteración tener

este resultado. Reduciendo la tolerancia la diferencia entre las estimaciones por los dos métodos se hace menor, pero toma más iteraciones y es necesario permitir más iteraciones de la búsqueda lineal, además de que se necesita hacer *forward-tracking* en varios pasos. Esto último quiere decir que fue necesario agrandar la longitud del paso de la búsqueda para que se cumpliera la condición de la curvatura. El valor del máximo obtenido es -232,3441, lo que lo hace equivalente al de la función de control.

Debe tenerse en cuenta que este y el resto de los métodos presentados cuando convergen lo hacen a *un* máximo, lo que no significa que converjan al máximo global. Esto es una limitación ya que usualmente no tendremos la certeza de que no haya un vector que haga aun mayor el valor de la función a maximizar. En el caso que se pueda, una manera subsanar esto es probando distintos puntos de arranque de una grilla y ver a donde convergen, si lo hacen a un mismo punto es evidencia a favor de que se está en presencia de un máximo. El problema con este camino es de caracter práctico; incluso considerando un número reducido de valores para la grilla (por ejemplo: -2, -1, 0, 1 y 2) el espacio que ocuparía en la memoria es demasiado grande debido al número de coeficientes a estimar; sumándose a esto el elevado tiempo que implicaría evaluar suficientes valores iniciales distintos para encontrar los que hagan que el método del ascenso más rápido (o cualquier otro) converja. Reduciendo aún más la grilla (los valores posibles son -1, 0 y 1) el primero problema se soluciona, pero sigue presente el segundo.

Probemos ahora con un vector de valores iniciales (x_0) compuesto solo de ceros. Esto nos lleva a que el algoritmo no converja a una solución, dadas las 1000 asignadas y presente varios fallos en la búsqueda lineal (el número de iteraciones -10- resultaba muy reducido para encontrar una longitud del paso del algoritmo apropiada). Incluso asignando 5000 iteraciones el resultado continúa siendo el mismo. Esto nos lleva a pensar que se necesita un mejor punto de inicio para que el algoritmo converja más velozmente, es decir, uno más cercano al verdadero valor que máximiza la log-verosimilitud.

Se puede intentar llegar a una solución utilizando estimaciones iniciales dadas por el método de mínimos cuadrados ordinarios (MCO) y considerando el salario en vez de la indicadora utilizada. Esto último puede ser de utilidad ya que las estimaciones pueden dar una idea aproximada de la relación entre nuestra variable dependiente y sus regresores. Pero utilizar las estimaciones por MCO no nos lleva a que se alcance la convergencia del algoritmo, incluso permitiéndole al algoritmo utilizar 3000 iteraciones. Aunque puede que se encuentre la solución con un mayor número de iteraciones, conviene encontrar un punto de arranque que permita llegar a la convergencia en un número menor de iteraciones y por lo tanto en menor tiempo.

A excepción del caso donde se elige como punto de partida el propio vector objetivo, este método no encuentra una rápida convergencia para el problema planteado.

5.2.2. Método de Newton

Dado los problemas de la velocidad de convergencia y con el vector inicial del método anterior, parece una buena opción intentar utilizar el método de Newton. Como se mencionó previamente, este algoritmo acelera la convergencia del método del ascenso más rápido al incorporar información de la hessiana de la función en el algoritmo.

Para que en efecto el método de Newton converja a un máximo y no a un mínimo, tenemos que tener una hessiana definida negativa, así que por lo menos en el primer paso debemos que comprobar que la hessiana evaluada en el vector inicial cumpla esta condición. A modo de comprobar esto, utilizamos una función vista en el curso que calcula los vectores propios de la hessiana inicial, si todos ellos resultan negativos la matriz es definida negativa.

Al igual que con el método anterior, comencemos con el punto inicial de "control", el vector de estimaciones dadas por la función *glm()*. Primero hay que comprobar que la hessiana de la log-verosimilitud evaluada en dicho vector es definida negativa con la función antes mencionada, lo que nos da como resultado que en efecto lo es. Entonces nos disponemos a utilizar el algoritmo de Newton.

Análogamente a lo que ocurrió con el método del ascenso más rápido, el de Newton converge en una iteración a un vector de coeficientes muy similares y un máximo equivalente a los obtenidos con la función *glm()*. Cabe destacar que la búsqueda lineal necesito hacer *forward-tracking* y se da un fallo en la búsqueda lineal.

Ahora intentemos utilizar como vectores iniciales el compuesto por ceros y el de las estimaciones MCO. La hessiana de la log-verosimilitud resulta definida negativa al evaluarla en ambos vectores, por lo que en un principio pueden ser candidatos.

Variable	glm()	Newton (Vector de 0's)	Newton (Vector de $\hat{\beta}_{MCO}$)
(Intercept)	-4.442367437	-4.442363082	-4.442361732
female	-1.194755469	-1.194755231	-1.194755497
educ	0.277912495	0.277912242	0.277912116
exper	0.007568890	0.007568853	0.007568843
tenure	0.083587709	0.083587710	0.083587707
northcen	-0.012727340	-0.012727760	-0.012727487
south	0.035935951	0.035935444	0.035935708
west	0.688820844	0.688820173	0.688820410
reg.metro	0.745975971	0.745975951	0.745975895
construc	-0.363769611	-0.363771043	-0.363770666
services	-0.493415175	-0.493415066	-0.493415144
trade	-1.429567604	-1.429567877	-1.429567922
profocc	0.854540015	0.854540092	0.854540730
clerocc	-0.585495085	-0.585495421	-0.585494736
servocc	-1.209387935	-1.209388297	-1.209387806

Cuadro 6: Comparación de las estimaciones de glm() y el método de newton

Podemos ver que este método convergió rápidamente, en tan solo cuatro iteraciones en el caso del vector de ceros y en siete usando las estimaciones MCO, necesitando hacer *forward-tracking* solo una vez en ambos casos. En este caso el vector nulo parece ser una buena opción de punto de arranque si no se cuenta con información sobre los coeficientes a estimar, incluso mejor que utilizar las estimaciones MCO ya que nos ahorramos iteraciones.

5.2.3. Método de Broyden-Fletcher-Goldfarg-Shanno (BFGS)

Este método está comprendido dentro de la categoría de los de Cuasi-Newton. En vez de requerir la evaluación de la hessiana de la función a maximizar, en estos métodos se la aproxima utilizando la información de lo cambios en el gradiente. Al ahorrarse la evaluación de la hessiana, estos métodos pueden resultar más veloces que el de Newton, aunque esto también dependerá del vector inicial empleado; el de Newton tenderá a tomar menos iteraciones si se comienza cerca de la solución. De todas maneras resulta necesario una evaluación inicial de la hessiana, que se puede elegir como el opuesto de la matriz identidad cuya dimensión es la cantidad de variables de la función a optimizar, o emplear una estimaciones de la hessiana en x_0. Los distintos métodos de Cuasi-Newton se diferencian en la manera en que se aproxima y actualiza la hessiana, en particular el BFGS no implica la inversión de la aproximación actualizada, sino actualizar la inversa como se explicó anteriormente.

Nuevamente comenzamos tomando como vector inicial el de las estimaciones de la función *glm()*, obteniéndose un resultado análogo a los casos anteriores. Se quiso también hacer una comparación con el resultado obtenido con el algoritmo bfgs de la función *optim()* de la biblioteca *stats*, pero tuvo problemas al aproximar la hessiana. Continuamos presentando las estimaciones resultantes de utilizar como vector inicial el nulo y el de las estimaciones MCO.

Variable	glm()	BFGS (vector de 0's)	BFGS ($\hat{\beta}_{MCO}$)
(Intercept)	-4.442367437	-4.441287082	-4.442571372
female	-1.194755469	-1.194356801	-1.194609027
educ	0.277912495	0.277846624	0.277942647
exper	0.007568890	0.007565239	0.007576988
tenure	0.083587709	0.083585123	0.083582161
northcen	-0.012727340	-0.012964024	-0.012858416
south	0.035935951	0.035914771	0.035839134
west	0.688820844	0.688620025	0.688680786
reg.metro	0.745975971	0.745850209	0.745871069
construc	-0.363769611	-0.363854741	-0.363864731
services	-0.493415175	-0.493346501	-0.493359194
trade	-1.429567604	-1.429542202	-1.429527372
profocc	0.854540015	0.854472462	0.854280808
clerocc	-0.585495085	-0.585483998	-0.585675743
servocc	-1.209387935	-1.209298455	-1.209619458

Cuadro 7: Comparación de las estimaciones de glm() y el método BFGS

Si bien se llega al mismo máximo que en el método anterior, el desempeño de este método resultó peor que el de Newton en cuanto a la cantidad de iteraciones que toma para converger y la precisión de las estimaciones respecto a *glm()*. En cuanto a lo primero, se puede deber a que se partió de un vector lo suficientemente similar al que maximiza la log-verosimilitud para que el método de Newton fuera más veloz que el BFGS. Lo segundo puede ser consecuencia de que al aproximarse y no evaluarse la hessiana se pierde precisión, esto se puede solucionar reduciendo la tolerancia, aunque la consecuencia de esto es un aumento en las iteraciones necesarias para la convergencia.

6. Conclusiones y futuros pasos

A continuación, se presenta una tabla que resume los resultados principales

	Método	Iteraciones	f(x_max)	Precisión respecto a glm()
1	Ascenso más rápido	No converge	No converge	No converge
2	Newton (vector 0's)	4	-232.344056522223	5 cifras desp. coma
3	Newton (beta_mco)	7	-232.34405652223	5 cifras desp. coma
4	BFGS (vector 0's)	35	-232.344059820262	2 cifras desp. coma
5	BFGS (beta_mco)	24	-232.344057462874	2 cifras desp. coma

Cuadro 8: Resumen de resultados de los distintos métodos.

En el contexto tratado, el método del ascenso más rápido no llegó a converger en un número de iteraciones práctico comparable a los otros métodos. Esto puede tener como origen la selección de los vectores de valores iniciales, no lo suficientemente cercanos al vector solución para que el método convergiera rápidamente. Los otros dos métodos, el de Newton y el Broyden-Fletcher-Goldfarg-Shanno convergieron rápidamente, aunque el primero se destacó por ser más veloz y tomando como referencia las estimaciones por el método IWLS empleado por la función *glm()* resultó más preciso en cuanto a los coeficientes estimados (para el mismo nivel de tolerancia). El método de Newton es más veloz al emplear como vector inicial el de ceros, mientras que el BFGS lo es cuando se le suministran las estimaciones por el método de mínimos cuadrados ordinarios. Los máximos alcanzados son muy semejantes en los métodos de Newton y el BFGS, a la tolerancia de 10^{-5} empleada se puede decir que son equivalentes.

A futuro se podría profundizar en métodos para la elección de vectores iniciales, como las mencionadas grillas de valores, ya que si bien en este caso se tenía una idea de donde partir para enfrentar el problema esto no

va a ocurrir en la mayoría de aplicaciones. La comparación con otros métodos de optimización adicionales también sería otro punto para expandir sobre lo trabajado.

7. Bibliografía

- Álvarez-Vaz, R. y Massa, F (2020). Optimización (diapostivas del curso).
- David B. Dahl, David Scott, Charles Roosen, Arni Magnusson and Jonathan Swinton (2019). xtable: Export Tables to LaTeX or HTML. R package version 1.8-4. https://CRAN.R-project.org/package=xtable
- Sam Firke (2020). janitor: Simple Tools for Examining and Cleaning Dirty Data. R package version 2.0.1. https://CRAN.R-project.org/package=janitor
- Jones, O., Maillardet R. y Robinson A. (2014). Introduction to scientific programming and simulation using R. CRC press. Florida.
- Nalbarte, L. (2020). Modelos lineales generalizados (diapositivas del curso).
- R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
- Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2020). dplyr: A Grammar of Data Manipulation. R package version 1.0.2. https://CRAN.R-project.org/package=dplyr
- Hadley Wickham and Jim Hester (2020). readr: Read Rectangular Text Data. R package version 1.4.0. https://CRAN.R-project.org/package=readr
- Hadley Wickham and Dana Seidel (2020). scales: Scale Functions for Visualization. R package version 1.1.1. https://CRAN.R-project.org/package=scales
- Wooldridge, J.M. (2009). Introducción a la Econometría, Un enfoque moderno, 4a edición. México, D.F. Cengage Learning.

8. Apéndice: Código utilizado

8.1. Librerías empleadas

```
library(dplyr)
library(readr)
library(janitor)
library(xtable)
library(scales)
```

8.2. Carga y limpieza de los datos

```
#Arreglo inicial de los datos

datos <- read_delim("wage1.txt", delim = "\t") %>%
  as.data.frame %>%
  clean_names() %>%
  rename(salario = "wage", reg.metro = "smsa", log.salario= "lwage")

as.numeric(sapply(datos, is.numeric))

names(subset(datos, select=!as.numeric(sapply(datos, is.numeric))))

sapply(subset(datos, select=!as.numeric(sapply(datos, is.numeric))), class)

head(subset(datos, select=!as.numeric(sapply(datos, is.numeric))))

datos <- datos %>%
  mutate(salario = gsub(",", ".", salario), log.salario = gsub(",", ".", log.salario) ) %>%
  mutate_at(c("salario", "log.salario"), as.numeric)



#datos con variables indicadoras categoricas y hacemos categorica la
#variable salario (usamos las categorias menor a 15 y mayor a 15 dolares por hora)
datos <- datos %>%
  mutate(
    region = as.factor(case_when(
                northcen == 1 ~ "northcen",
                south == 1 ~ "south",
                west == 1 ~ "west",
                northcen + south + west == 0 ~ "east"
                )),
    rama_act = as.factor(case_when(
                  construc == 1 ~ "construc",
                  trade == 1 ~ "trade",
                  services == 1 ~ "services",
                  construc + trade + services == 0 ~ "otros"
                )),
    ocupacion = as.factor(case_when(
```

```
                profocc == 1 ~ "profocc",
                clerocc == 1 ~ "clerocc",
                servocc == 1 ~ "servocc",
                profocc + clerocc + servocc == 0 ~ "otros",
    )),
    sal_med = ifelse(salario >= 5.896103 , 1, 0),
    # sal_min = ifelse(salario >= 2.3 , 1, 0),

    educsq = educ^2
  ) %>%
  dplyr::select(-profserv)
```

8.3. Creación de tablas de resumen

```
resumen_tot1 <- datos %>%
                summarise(region = "Todas",
                          Cantidad=n(),
                          `Mínimo`= min(salario),
                          Media=mean(salario),
                          `Máximo` = max(salario)) %>%
                 rename(`Región` = "region")

resumen_tot2 <- datos %>%
                summarise(rama_act = "Todas",
                          Cantidad=n(),
                          `Mínimo`= min(salario),
                          Media=mean(salario),
                          `Máximo` = max(salario)) %>%
                rename(`Rama de Actividad` = "rama_act")

resumen_tot3 <- datos %>%
                summarise(ocupacion = "Todas",
                          Cantidad=n(),
                          `Mínimo`= min(salario),
                          Media=mean(salario),
                          `Máximo` = max(salario)) %>%
                rename(`Ocupación` = "ocupacion")

resumen_reg <- datos %>%
               group_by(region) %>%
               summarise(Cantidad=n(),
                         `Mínimo`= min(salario),
                         Media=mean(salario),
                         `Máximo` = max(salario)) %>%
               arrange(forcats::fct_relevel(region,
                                    "northcen",
                                    "south",
                                    "east",
                                    "west")) %>%
               mutate(region = forcats::fct_recode(region,
                                        "Norte"="northcen",
```

```
                                        "Sur"="south",
                                        "Este"="east",
                                        "Oeste"="west")) %>%
                rename(`Región`="region")

resumen_rama_act <- datos %>%
                     group_by(rama_act) %>%
                     summarise(Cantidad=n(),
                               `Mínimo`= min(salario),
                               Media=mean(salario),
                               `Máximo` = max(salario)) %>%
                     arrange(forcats::fct_relevel(rama_act,
                                        "construc",
                                        "services",
                                        "trade",
                                        "otros")) %>%
                     mutate(rama_act = forcats::fct_recode(rama_act,
                                        "Construcción"="construc",
                                        "Servicios"="services",
                                        "Comercio"="trade",
                                        "Otros"="otros")) %>%
                     rename(`Rama de Actividad` = "rama_act")

resumen_ocupacion <- datos %>%
                      group_by(ocupacion) %>%
                      summarise(Cantidad=n(),
                                `Mínimo`= min(salario),
                                Media=mean(salario),
                                `Máximo` = max(salario)) %>%
                      arrange(forcats::fct_relevel(ocupacion,
                                        "servocc",
                                        "clerocc",
                                        "profocc",
                                        "otros")) %>%
                      mutate(ocupacion = forcats::fct_recode(ocupacion,
                                        "Servicio"="servocc",
                                        "Administrativos"="clerocc",
                                        "Profesional"="profocc",
                                        "Otros"="otros")) %>%
                      rename(`Ocupación` = "ocupacion")




propsal <- datos %>%
  group_by(sal_med) %>%
  summarise(`Proporción` = n()/dim(datos)[1]) %>%
  mutate(sal_med = ifelse(sal_med == 1, "Sobre promedio", "Bajo el promedio"))

propfem <- table(as.factor(datos$female))/dim(datos)[1]
propwhite <- table(as.factor(datos$nonwhite))/dim(datos)[1]
propmarr <- table(as.factor(datos$married))/dim(datos)[1]
```

```r
propmetro <- table(as.factor(datos$reg.metro))/dim(datos)[1]

sal_tabla <-xtable(
  propsal,
  caption = "Proporción de personas con salarios
  por debajo y sobre el promedio"
  )

rbind(resumen_reg, resumen_tot1) %>%
       xtable(
         caption = "Cantidad de observaciones y salario medio, mínimo
         y máximo para las distintas regiones.")

 rbind(resumen_rama_act, resumen_tot2) %>%
       xtable(
         caption = "Cantidad de observaciones y salario medio,
         mínimo y máximo para las distintas ramas de actividad.")

rbind(resumen_ocupacion, resumen_tot3) %>%
       xtable(caption = "Cantidad de observaciones y salario medio,
              mínimo y máximo para las distintas ocupaciones.")
```

8.4. Estimación del modelo mediante glm()

```r
#modelo logit

modelo_ctrl <- glm(sal_med ~ female + educ + exper + tenure + northcen +
                     south + west + reg.metro + construc + services +
                     trade + profocc + clerocc + servocc,
                   family = binomial (link = "logit") ,data = datos)


# cuadrovars <- xtable(as.data.frame(summary(modelo_ctrl)$coefficients[,1]), digits = 8)
```

8.5. Función de log-verosimiltud a maximizar

```r
#funcion a maximizar

X <- dplyr::select(datos,female, educ, exper, tenure,
                   northcen, south, west, reg.metro, construc,
                   services, trade, profocc, clerocc, servocc)

X <- cbind(rep(1, dim(X)[1]), X)

log_ver <- function(b){
  #b tiene que ser un vector k+1 = 14 + 1 dimensional

  lv <- sum((as.numeric(datos$sal_med))*(as.matrix(X)%*%b)) - sum(log(1+exp(as.matrix(X)%*%b)))
```

```
  return(lv)
}
```

8.6. Valor máximo de la log-verosimilitud

```
max_ctrl <- log_ver(summary(modelo_ctrl)$coefficients[,1])
```

8.7. Funciones auxiliares de los métodos

```
#(Álvarez y Massa, 2020)

#para calcular numericamente el gradiente
gradiente <- function(FUN, x0){

  f0 <- FUN(x0)
  h <- sqrt(.Machine$double.eps)
  k <- length(x0)
  dfx <- rep(0, k)

  for(i in 1:k){
    xi <- x0
    xi[i] <- xi[i] + h
    dfx[i] <- (FUN(xi)-f0)/h
  }

  return(dfx)
}

#hessiana
hessiana <- function(FUN, x0){

  f0 <- FUN(x0)
  h <- (.Machine$double.eps)^(1/3)
  k <- length(x0)
  Hfx <- matrix(0,k,k)

  for(i in 1:k){
    for(j in 1:k){
    xi <- x0
    xj <- x0
    xij <- x0

    xi[i] <- xi[i] + h
    xj[j] <- xj[j] + h

    xij[i] <- xij[i] + h
    xij[j] <- xij[j] + h

    Hfx[i,j] <- (FUN(xij)-FUN(xi)-FUN(xj)+f0)/(h^2)
```

```
    }
  }

  return(Hfx)
}




#busqueda lineal

blineal<-function(FUN,xn,pn,a_max=1,tau=0.5,maxiter_bt=10,
                  c1=1e-4,c2=0.9,tau2=0.8,mostrar_bl=TRUE){
    phi<-function(x,a,p) FUN(x+a*p)
    phi0<-phi(xn,0,pn)
    alfa<-a_max
    phia<-phi(xn,alfa,pn)

    # condicion de ascenso suficiente
    iter_bt<-1
    armijo<-phi0+c1*alfa*sum(pn^2)
    while(phia<=armijo & iter_bt<maxiter_bt){
        alfa<-alfa*tau
        armijo<-phi0+alfa*c1*sum(pn^2)
        phia<-phi(xn,alfa,pn)
        iter_bt<-iter_bt+1
    }

  #condicion de curvatura
    iter_ft<-0
    while(sum(gradiente(FUN, xn+alfa*pn)*pn) > c2*sum(pn^2)){
        if(iter_ft==0 & mostrar_bl==TRUE) cat('Realizando forward-tracking','\n')
        iter_ft<-iter_ft+1
        alfa_old<-alfa/tau
        dif<-alfa_old - alfa
        alfa<-alfa + dif*(1-tau^iter_ft)
    }

    if(iter_bt==maxiter_bt & mostrar_bl==TRUE) cat('Fallo la busqueda lineal','\n')
    salida<-list(alfa=alfa,iter_bt=iter_bt,iter_ft=iter_ft)
    return(salida)
}
```

8.8. Código del algoritmo del ascenso más rápido

```
#metodo ascenso mas rapido (Álvarez y Massa, 2020)


ascenso <- function(FUN,x0,tol=1e-5,maxiter=1000,mostrar=TRUE,
                    a_max=1,tau=0.5,maxiter_bt=10,c1=1e-4,c2=0.9,tau2=0.8,trace_bl=mostrar){
    pn <- gradiente(FUN, x0)
```

```
    alfa<-blineal(FUN,x0,pn,a_max=a_max,tau=tau,
                  maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
    an<-alfa$alfa
    x1<-x0+an*pn
    iter<-1
    cambio<-sum(pn^2)

    while(cambio>tol & iter<maxiter){
        x0<-x1
        pn<-gradiente(FUN, x0)
        alfa<-blineal(FUN,x0,pn,a_max=a_max,tau=tau,
                      maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
        an<-alfa$alfa
        x1<-x0+an*pn
        if(mostrar==TRUE) cat(paste('x',iter,sep=''),x1,'\n')
        iter<-iter+1
        cambio<-sum(pn^2)
    }
    if(iter==maxiter & mostrar==TRUE) cat('No se alcanzo convergencia','\n')

    salida<-list(x_max=x1,fx_max=FUN(x1),iter=iter)
    return(salida)
}
```

8.9. Estimación tomando como base el vector de estimaciones MCO

```
pars_ctrl <- c(summary(modelo_ctrl)$coefficients[,1])

asce1 <- ascenso(FUN = log_ver, x0 = pars_ctrl)


xtable(cbind(pars_ctrl, as.numeric(asce1$x_max)),
       digits = 9,
       caption = "Comparación de las estimaciones de glm() y el método del ascenso más rápido")

#max(pars_ctrl- as.numeric(asce1$x_max))
```

8.10. Grillas para Ascenso más rápido

```
#  vg <- seq(-2,2,1)
# vg2 <- seq(-1,1,1)

#grilla1 <- expand.grid(vg,vg,vg,vg,vg,vg, vg,vg,vg,vg,vg,vg,vg,vg,vg)

#grilla2 <- expand.grid(vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2,vg2)
```

8.11. Estimación mediante el método del ascenso más rápido, vector inicial de 0's

```
#CUIDADO, tarda, se comentan para que la compilacion del rmd tarde menos
#asce2 <- ascenso(log_ver, rep(0, 15))

#asce3 <- ascenso(log_ver, rep(0, 15), maxiter = 5000, maxiter_bt  = 20)
```

8.12. Estimaciones por mínimos cuadrados ordinarios

```
mod_mco <- lm(
  salario ~ female + educ + exper + tenure +
    northcen + south + west + reg.metro + construc +
    services + trade + profocc + clerocc + servocc, data = datos
  )

summMCO <- summary(mod_mco)$coefficients[,1]

#print.xtable(xtable(t(summMCO)), include.rownames=FALSE)
```

8.13. Estimación mediante el método del ascenso más rápido, vector inicial de estimaciones MCO

```
#asce4 <- ascenso(log_ver, summary(mod_mco)$coefficients[,1])

#asce5 <- ascenso(log_ver, summary(mod_mco)$coefficients[,1], maxiterbt = 20)

#asce6 <- ascenso(log_ver, summary(mod_mco)$coefficients[,1], maxiter = 2000)

#asce7 <- ascenso(log_ver, summary(mod_mco)$coefficients[,1], maxiter = 5000, maxiter_bt = 20)
```

8.14. Método de newton

```
#metodo de newton (Álvarez y Massa, 2020)

newton<-function(FUN, x0,tol=1e-5,maxiter=1000,mostrar=TRUE,a_max=1,tau=0.5,
                 maxiter_bt=10,c1=1e-4,c2=0.9,tau2=0.8,trace_bl=mostrar){
    pn <- -solve(hessiana(FUN, x0))%*%gradiente(FUN, x0)
    alfa<-blineal(FUN, x0,pn,a_max=a_max,tau=tau,
                  maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
    an<-alfa$alfa
    x1<-x0+an*pn
    iter<-1
    cambio<-sum(pn^2)

    while(cambio>tol & iter<maxiter){
```

```
        x0<-x1
        pn<- -solve(hessiana(FUN, x0))%*%gradiente(FUN, x0)
        alfa<-blineal(FUN,x0,pn,a_max=a_max,tau=tau,
                      maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
        an<-alfa$alfa
        x1<-x0+an*pn
        if(mostrar==TRUE) cat(paste('x',iter,sep=''),x1,'\n')
        iter<-iter+1
        cambio<-sum(pn^2)
    }
    if(iter==maxiter & mostrar==TRUE) cat('No se alcanzo convergencia','\n')

    salida<-list(x_max=x1,fx_max=FUN(x1),iter=iter)
    return(salida)
}


chequeo<-function(M){
    lam<-eigen(M)$values
    if(all(lam>0)) cat('M es definida positiva','\n')
    if(all(lam<0)) cat('M es definida negativa','\n')
    if(any(lam>0) & any(lam<0)) cat('M es indefinida','\n')
    return(lam)
}
```

8.15. Aplicación del Método de Newton

```
check1 <- chequeo(hessiana(log_ver, pars_ctrl))

newton1 <-newton(log_ver, pars_ctrl)

check2 <- chequeo(hessiana(log_ver, rep(0,15)))
check3 <- chequeo(hessiana(log_ver, summMCO))

newton2 <-newton(log_ver, rep(0,15))

newton3 <- newton(log_ver, summMCO)

xtable(cbind(pars_ctrl, as.numeric(newton2$x_max), as.numeric(newton3$x_max)),
       digits = 9,
       caption = "Comparación de las estimaciones de glm() y el método de newton")
```

8.16. Método BFGS

```
#metodo bfgs

bfgs<-function(FUN,x0,tol=1e-5,maxiter=1000,mostrar=TRUE,a_max=1,
               tau=0.5,maxiter_bt=10,c1=1e-4,c2=0.9,tau2=0.8,trace_bl=mostrar){
    H<- -diag(length(x0))
```

```
    pn <- df0 <- -H%*%gradiente(FUN, x0)
    alfa<-blineal(FUN,x0,pn,a_max=a_max,tau=tau,
                  maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
    an<-alfa$alfa
    x1<-x0+an*pn
    df1<-gradiente(FUN, x1)
    iter<-1
    cambio<-sum(pn^2)
    Id<--H

    while(cambio>tol & iter<maxiter){
        sn<-x1-x0
        yn<-df1-df0
        rn<-1/sum(yn*sn)
        H<-(Id-rn*sn%*%t(yn))%*%H%*%(Id-rn*yn%*%t(sn))+rn*sn%*%t(sn)
        x0<-x1
        df0<-df1
        pn<--H%*%df1
        alfa<-blineal(FUN,x0,pn,a_max=a_max,tau=tau,
                      maxiter_bt=maxiter_bt,c1=c1,c2=c2,tau2=tau2,mostrar_bl=trace_bl)
        an<-alfa$alfa
        x1<-x0+an*pn
        df1<-gradiente(FUN, x1)
        if(mostrar==TRUE) cat(paste('x',iter,sep=''),x1,'\n')
        iter<-iter+1
        cambio<-sum(pn^2)
    }
    if(iter==maxiter & mostrar==TRUE) cat('No se alcanzo convergencia','\n')

    salida<-list(x_max=x1,fx_max=FUN(x1),iter=iter)
    return(salida)
}
```

8.17. Aplicación de BFGS

```
bfgs1 <- bfgs(FUN = log_ver,x0 = pars_ctrl)

# log_ver_op <- function(b){
#   #b tiene que ser un vector k+1 = 14 + 1 dimensional
#
#   lv <- -sum((as.numeric(datos$sal_med))*(as.matrix(X)%*%b)) - sum(log(1+exp(as.matrix(X)%*%b)))
#
#   return(lv)
# }

# optim no puede encontrar el maximo
# bfgsr <- optim(par = rep(0,15), fn = log_ver_op, method = "BFGS", control = list(trace = TRUE))

bfgs2 <- bfgs(FUN = log_ver,x0 = rep(0,15))

bfgs3 <- bfgs(FUN = log_ver,x0 = summMCO, maxiter_bt = 20)
```

```
bfgstab <- xtable(cbind(pars_ctrl, as.numeric(bfgs2$x_max), as.numeric(bfgs3$x_max)),
            digits = 9,
             caption = "Comparación de las estimaciones de glm() y el método BFGS")
```

8.18. Resumen de resultados

```
resumen_fin <- tibble(
  metodo = c(
    "Ascenso más rápido",
    "Newton (vector 0's)",
    "Newton (beta_mco)",
    "BFGS (vector 0's)",
    "BFGS (beta_mco)"),
  Iteraciones = c(
    "No converge",
    newton2$iter,
    newton3$iter,
    bfgs2$iter,
    bfgs2$iter),
  f_max = c(
    "No converge",
    newton2$fx_max,
    newton3$fx_max,
    bfgs2$fx_max,
    bfgs3$fx_max),
  `Precisión respecto a glm()` = c(
    "No converge",
    "5 cifras desp. coma",
    "5 cifras desp. coma",
    "2 cifras desp. coma",
    "2 cifras desp. coma")
  ) %>%

  rename(
    `Método` = metodo,
    `f(x_max)` = f_max
    ) %>%
  xtable(caption = "Resumen de resultados de los distintos métodos.")
```