In [29]:
```
using LinearAlgebra
using Plots
using LatexPrint

include("Cholesky.jl")
include("Algoritmo LU.jl")
```

```
┌ Info: Precompiling LatexPrint [d2208f48-c256-5759-9089-c25ed2a93924]
└ @ Base loading.jl:1342
```

Out[29]:
```
LUPP (generic function with 1 method)
```

Unidad 4: Mínimos cuadrados lineales

U4 1

1. Dados los siguientes vectores:

$$\alpha = \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \quad \beta = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}$$

encuentre la matriz de rotación G y el reflector de Householder F tales que

$$G\alpha = \begin{pmatrix} 5 \\ 0 \end{pmatrix}, \quad F\beta = \begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix}$$

Solución:

Sabemos que

$$G = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{1}$$

y que

$$\cos\theta = \frac{\alpha_1}{||\alpha||} = \frac{4}{5} \tag{2}$$

$$\sin\theta = -\frac{\alpha_2}{||\alpha||} = -\frac{3}{5}. \tag{3}$$

Por lo tanto,

$$G = \frac{1}{5}\begin{pmatrix} 4 & 3 \\ -3 & 4 \end{pmatrix} \tag{4}$$

.

Por otro lado, para construir a F consideramos el vector

$$v = ||\beta||\overline{e_1} - \beta \tag{5}$$
$$= \begin{pmatrix} 1 \\ -1 \\ -2 \end{pmatrix}. \tag{6}$$

Entonces definimos

$$H = I - 2\frac{vv^T}{v^T v} \tag{7}$$
$$= \frac{1}{6}(6I - 2vv^T) \tag{8}$$
$$= \frac{1}{6}\begin{pmatrix} 4 & 2 & 4 \\ 2 & 4 & -4 \\ 4 & -4 & -2 \end{pmatrix}. \tag{9}$$

Gracias a la teoría sabemos que

$$H\beta = \begin{pmatrix} ||\beta|| \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

. Por lo tanto, si definimos

$$F = -H = \frac{1}{6}\begin{pmatrix} -4 & -2 & -4 \\ -2 & -4 & 4 \\ -4 & 4 & 2 \end{pmatrix}. \tag{11}$$

entonces ocurrirá que

$$F\beta = \begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix}, \tag{12}$$

como buscábamos.

U4 2

1. Dada la matriz \begin{align}
   ```
   A = \begin{pmatrix}
   4 & 10 \\
   3 & 0
   \end{pmatrix}
   ```
   \end{align} Calcule su factorización QR (a mano) utilizando los métodos de Gram-Schmidt, reflexiones de Householder y rotaciones de Givens.

Solución:

Gram-Schmidt: Consideramos los vectores columna de A: $a_1 = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$, $a_2 = \begin{pmatrix} 10 \\ 0 \end{pmatrix}$. Al aplicar el método de Gram-Schmidt obtenemos los vectores

$$q_1 = \frac{a_1}{||a_1||} = \frac{1}{5}\begin{pmatrix} 4 \\ 3 \end{pmatrix}, \tag{13}$$

$$q_2 = \frac{a_2 - (q_1^T a_2)q_1}{||a_2 - (q_1^T a_2)q_1||} = \frac{1}{5}\begin{pmatrix} 3 \\ -4 \end{pmatrix}. \tag{14}$$

De esta manera,

$$Q = (q_1|q_2) = \frac{1}{5}\begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix}. \tag{15}$$

Definiendo

$$r_{11} = ||a_1|| = 5, \tag{16}$$
$$r_{12} = q_1^T a_2 = 8, \tag{17}$$
$$r_{22} = ||a_2 - (q_1^T a_2)q_1|| = 6, \tag{18}$$

obtenemos la matriz

$$R = \begin{pmatrix} 5 & 8 \\ 0 & 6 \end{pmatrix} \tag{19}$$

Un cálculo explícito muestra que, efectivamente,

$$A = QR. \tag{20}$$

Householder: Para usar Householder, consideramos el vector

$$x = A^1 = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \tag{21}$$

y definimos

$$v = ||x||\bar{e}_1 - x. \tag{22}$$

Entonces la matriz de Householder que necesitamos es:

$$H = I - 2\frac{vv^T}{v^T v} = \frac{1}{10}\begin{pmatrix} 8 & 6 \\ 6 & -8 \end{pmatrix} \tag{23}$$

De esta manera, definimos

$$R = H * A \tag{24}$$
$$= \begin{pmatrix} 5 & 8 \\ 0 & 6 \end{pmatrix} \tag{25}$$

y

$$Q = H^T \tag{26}$$
$$= \frac{1}{10}\begin{pmatrix} 8 & 6 \\ 6 & -8 \end{pmatrix} \tag{27}$$
$$= \frac{1}{5}\begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix} \tag{28}$$

Este es el mismo resultado que el que obtuvimos mediante el método de Gram-Schmidt.

Rotaciones de Givens: Buscamos una matriz de rotación G tal que

$$GA = \begin{pmatrix} 5 & ? \\ 0 & ? \end{pmatrix}. \tag{29}$$

Usamos la matriz G que calculamos en el primer problema:

$$G = \frac{1}{5}\begin{pmatrix} 4 & 3 \\ -3 & 4 \end{pmatrix}. \tag{30}$$

Entonces

$$R = GA = \begin{pmatrix} 5 & 8 \\ 0 & -6 \end{pmatrix}. \tag{31}$$

Además, si

$$Q := G^T \tag{32}$$

entonces se verifica que Q es ortogonal, R es triangular superior y $QR = A$. Notemos que aquí obtenemos un resultado distinto al que obtuvimos mediante Gram-Schmidt y mediante Householder.

U4 3

El siguiente problema consiste en determinar la curva de crecimiento de una población de bacterias. Los datos a utilizar son: el número de individuos de una especia particular de bacterias (y_i) en el tiempo (t_i).

$$t = [0, 4, 7.5, 25, 31, 48.75, 52, 58.5, 72.7, 78, 95, 96, 108, 112, 133, 136.75, 143, 156.5, 166.7, 181]$$
$$y = [8, 6, 6, 7, 8, 10, 13, 18, 33, 38, 76, 78, 164, 175, 280, 300, 320, 405, 385, 450].$$

1. Realice un programa que calcule el polinomio de ajuste de grado n (con n dada por el usuario) mediante el método de Ecuaciones normales.

2. ¿Qué pasa cuando el valor de n crece?, ¿es mejor el ajuste?, ¿tiene sentido para el fenómeno real?

In [10]:

```
# Datos dados. y_i es el número de individuos en el tiempo t_i.
t = [0 ; 4; 7.5; 25; 31; 48.75; 52 ; 58.5; 72.7; 78; 95; 96; 108; 112; 133; 136.75; 143; 1
y = [8; 6; 6; 7; 8; 10; 13; 18; 33; 38; 76; 78; 164; 175; 280; 300; 320; 405; 385; 450]

# La siguiente función toma como argumentos un número natural n dado por el usuario y los
# Regresa como output la función polinomial de grado n  PolyModel(x) que mejor interpola
# Esta función se obtiene mediante ecuaciones normales.

function PolyModel(n,t,y)
    #Número de observaciones
    m = length(t)

    #Calculamos la matriz de diseño A. Esta es una matriz de  mx(n+1). La entrada A_{ij} =
    A = zeros(m,n+1);
    for i = 1:m
        for j = 1:n+1
            A[i,j] = t[i]^(j-1)
        end
    end


    #Buscamos la factorización de Cholesky de A^T A
```

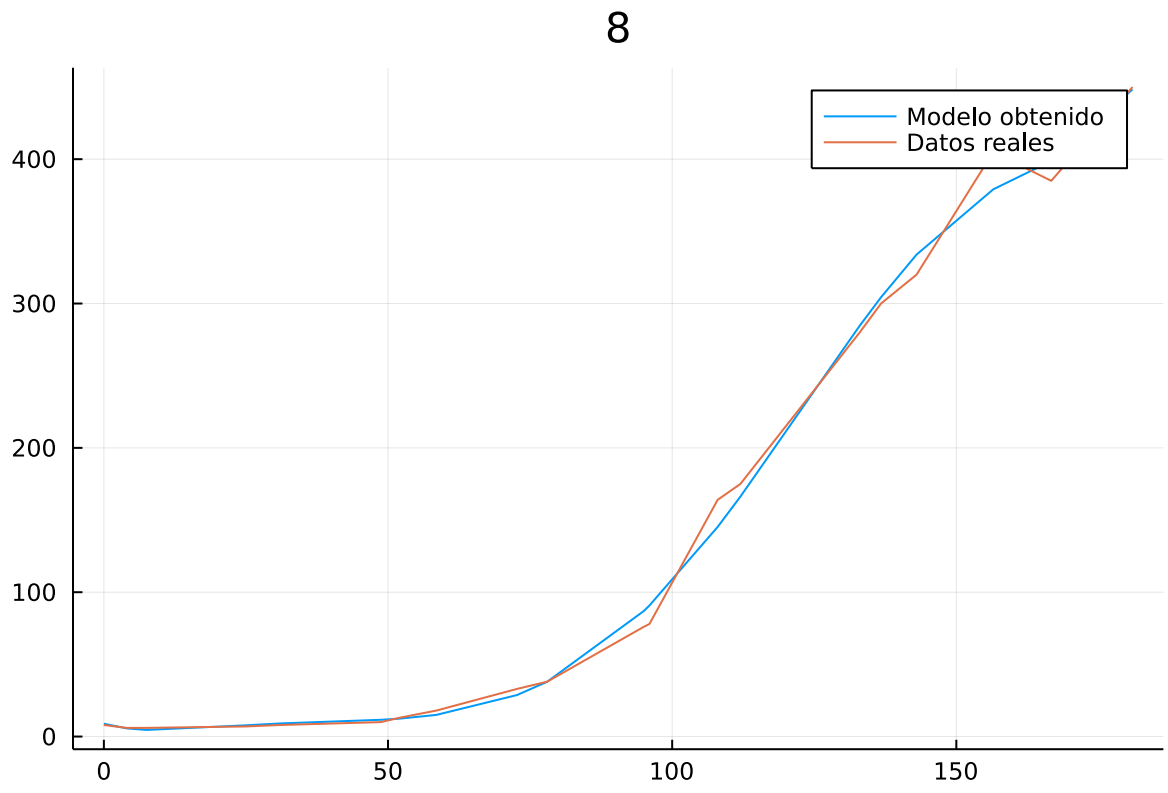
```
    (L,U) = FacChol(A'*A)

    # Encontramos una solución w_0 a la ecuación lineal  Lw = A^Ty mediante la función Sol
    #   (porque L es triangular inferior).
    w0 = SolFwd(L,A'*y)

    # Encontramos una solución c_0 a la ecuación lineal L^Tc = w_0 mediante la función Sol
    # (porque L^T es triangular superior).
    c0 = SolBwd(L',w0)

    #Aquí construimos la función polinomial PolyMod(x) = c0^T * (1,x, ... , x^n)^T
    function PolyFun(x)
        y = zeros(n+1)
        for i = 1:n+1
            y[i] = x^(i-1)
        end
        return c0'*y
    end

    return(PolyFun)
end
```

Out[10]:
```
PolyModel (generic function with 1 method)
```

A continuación veremos qué tan buen modelo es el que obtuvimos. Vamos a graficar para distintos valores de n

In [12]:
```
# Este dato lo puede cambiar el usuario. Es el grado del polinomio que
# vamos a usar para ajustar
n = 8
g = PolyModel(n,t,y)

plot(t,[g,y], title =  n, label=["Modelo obtenido" "Datos reales" ] )
```

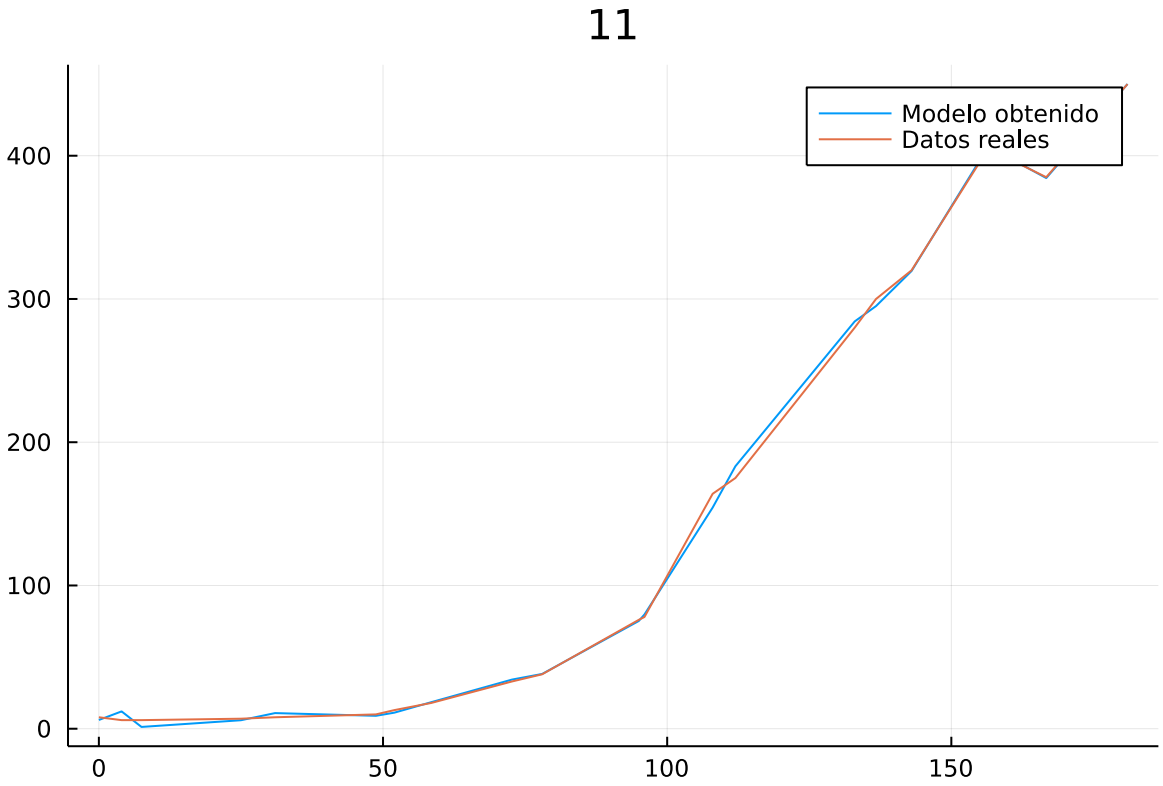
Out[12]:



In [15]:
```
n = 11
g = PolyModel(n,t,y)
```

```
plot(t,[g,y], title =  n, label=["Modelo obtenido" "Datos reales" ] )
```

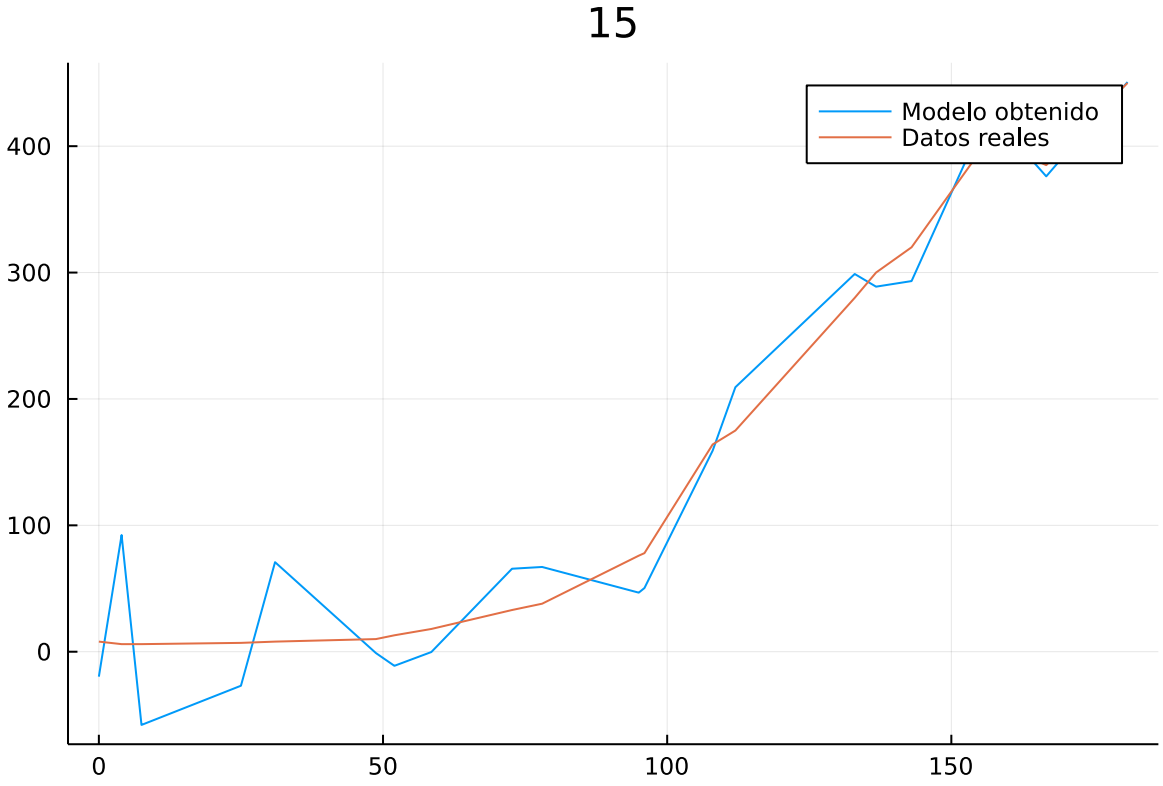
Out[15]:



In [14]:

```
n = 15
g = PolyModel(n,t,y)

plot(t,[g,y], title =  n, label=["Modelo obtenido" "Datos reales" ] )
```

Out[14]:

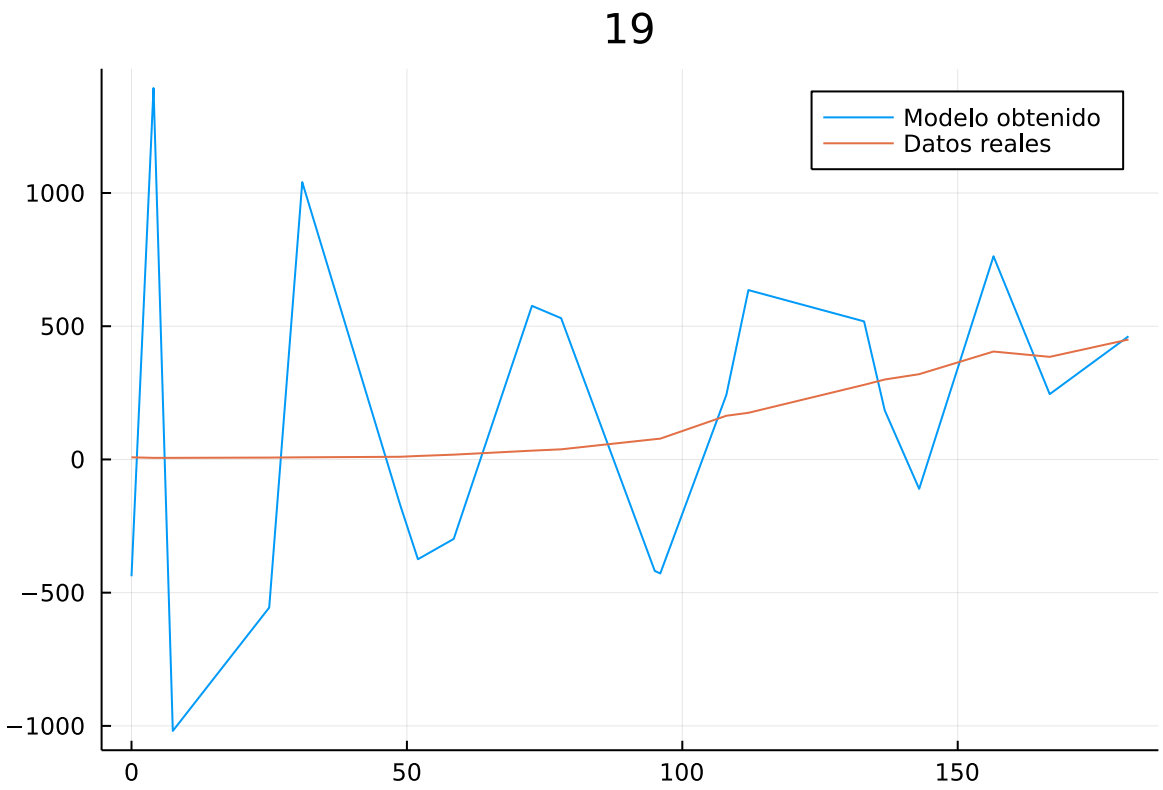


In [11]:

```
# Este dato lo puede cambiar el usuario. Es el grado del polinomio que
# vamos a usar para ajustar
n = 19
g = PolyModel(n,t,y)
```

```
plot(t,[g,y], title =  n, label=["Modelo obtenido" "Datos reales" ] )
```

Out[11]:




```
┌ Info: Precompiling GR_jll [d2c73de3-f751-5644-a686-071e5b155ba9]
└ @ Base loading.jl:1342
```

Notemos que cuando n se acerca a 11, el modelo se ajusta muy bien a los datos reales. Sin embargo, cuando $n = 19$, el modelo se aleja mucho de la realidad. De hecho, el modelo hace oscilaciones muy pronunciadas e incluso toma valores negativos muy grandes, lo cual no tiene sentido para el fenómeno real.

U4 4

Repita el proceso utilizando la factorización QR mediante el método de Householder. ¿Cómo son las gráficas ahora y los valores de los números de condición de la matriz R del sistema? Concluya.

In [17]:

```
# Primero definiremos una función QRHouseholder que tome como input una matriz A y nos reg
# obtenida mediante el método de Householder.

#Esta función toma como argumento una matriz A y regresa la matriz de Householder H tal qu
# HA es una matriz cuya primera columna tiene puros ceros debajo de la diagonal
function Householder(A)
    (m,n)=size(A)

    x = A[:,1]

    e1 = zeros(m)
    e1[1] = 1.0

    v = x+sign(A[1,1])*norm(x)*e1
    v = v/norm(v)

    H = I -2*v*v'
end

#Esta función toma como argumento una matriz A de mxn. Regresa una dupla (Q,R) donde Q es
# R es una matriz triangular superior tales que QR = A.
```

```
function QRHouseholder(A)
    (m,n) = size(A)
    Q = Matrix(1.0*I,m,m)
    R = A

    for i = 1:n
        HTild = Householder(R[i:m,i:n] )

        H = Matrix(1.0*I,m,m)
        H[i:m,i:m] = HTild

        R = H*R
        Q = H*Q
    end
    Q = Q'
    return(Q,R)
end


# La siguiente función toma como argumentos un número natural n dado por el usuario y los
# Regresa como output la función polinomial de grado n  PolyModelQR(x) que mejor interpola
# Esta función se obtiene mediante la factorización QR por el método de Householder


function PolyModelQR(n,t,y)
    #Número de observaciones
    m = length(t)

    #Calculamos la matriz de diseño A. Esta es una matriz de  mx(n+1). La entrada A_{ij} =
    A = zeros(m,n+1);
    for i = 1:m
        for j = 1:n+1
            A[i,j] = t[i]^(j-1)
        end
    end


    #Queremos resolver la ecuación  A c = y. Para esto encontramos la factorización QR de
    (Q,R) = QRHouseholder(A)

    # La ecuación se convierte en Rc= Q^Ty. Como R es triangular superior, esto lo podemos
    # hacia atrás

    #c0= SolBwd(R,Q'*y)

    #R tiene dimensiones mxn+1
    s = min(m,n+1)
    R2 = R[1:s,1:s]
    c0 = SolBwd(R2,Q'*y)

    #Aquí construimos la función polinomial PolyFun(x) = c0^T * (1,x, ... , x^n)^T
    function PolyFun(x)
        b = zeros(s)
        for i = 1:s
            b[i] = x^(i-1)
        end
        return c0'*b
    end

    return(PolyFun)
end
```

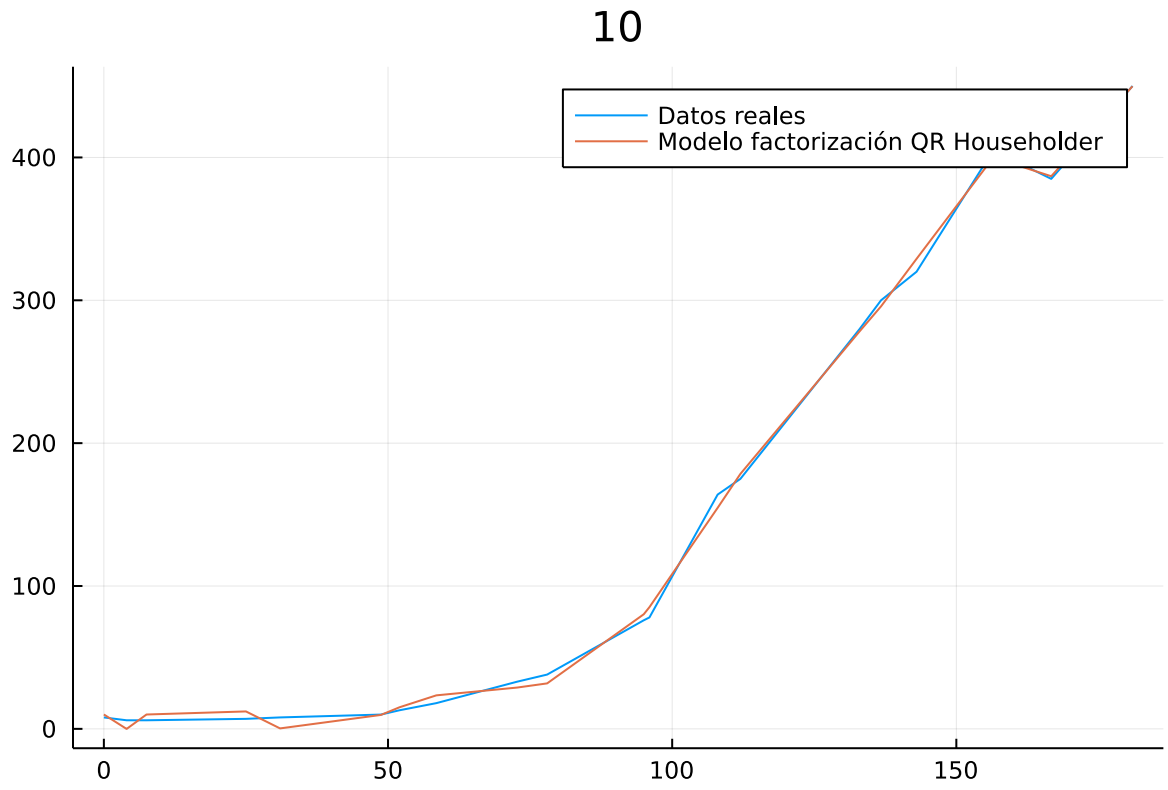
Out[17]:
```
PolyModelQR (generic function with 1 method)
```

Ahora haremos graficaremos el modelo obtenido y los datos reales para distintos valores de n.

In [20]:

```
# Este dato lo puede cambiar el usuario. Es el grado del polinomio que
# vamos a usar para ajustar. Necesitas m > n
n = 10

#Modelo a partir de algoritmo QR
h = PolyModelQR(n,t,y)


plot(t,[y,h], title =  n, label=["Datos reales" "Modelo factorización QR Householder" ] )
```

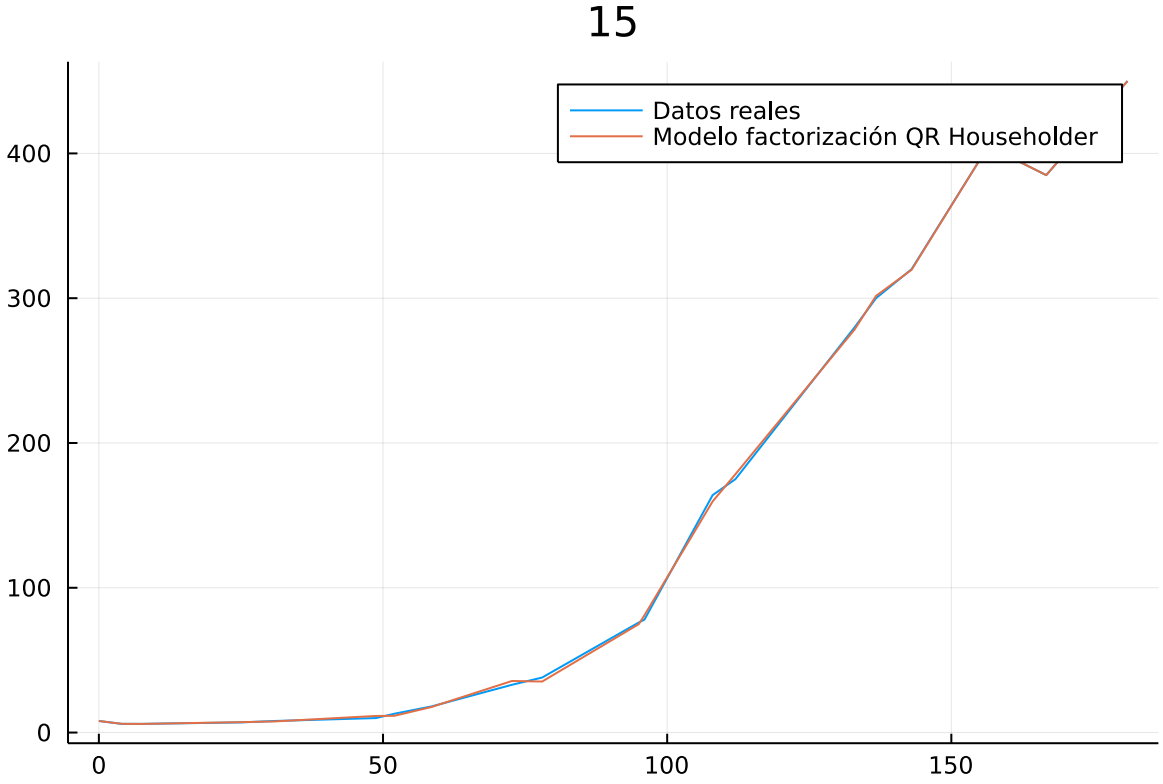
Out[20]:



In [19]:

```
# Este dato lo puede cambiar el usuario. Es el grado del polinomio que
# vamos a usar para ajustar. Necesitas m > n
n = 15

#Modelo a partir de algoritmo QR
h = PolyModelQR(n,t,y)


plot(t,[y,h], title =  n, label=["Datos reales" "Modelo factorización QR Householder" ] )
```

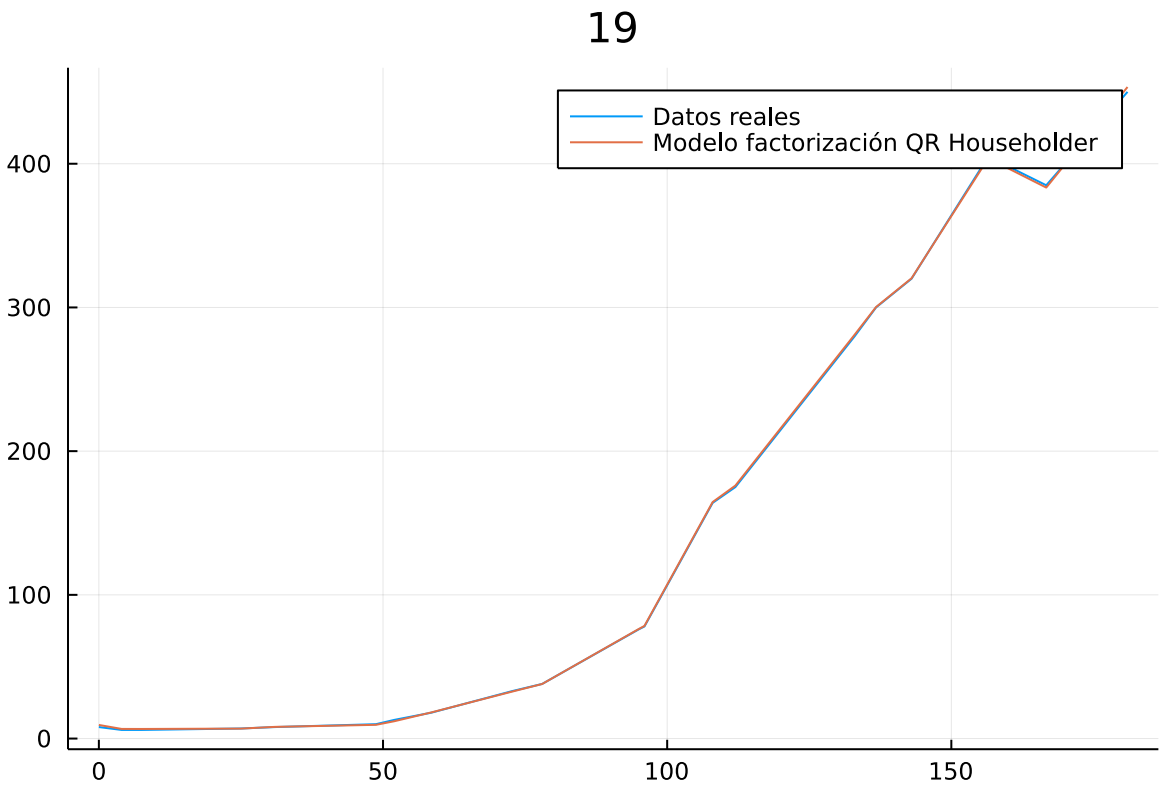
Out[19]:



In [18]:

```
# Este dato lo puede cambiar el usuario. Es el grado del polinomio que
# vamos a usar para ajustar. Necesitas m > n
n = 19

#Modelo a partir de algoritmo QR
h = PolyModelQR(n,t,y)


plot(t,[y,h], title =  n, label=["Datos reales" "Modelo factorización QR Householder" ] )
```

Out[18]:



Observemos que los datos se ajustan muy bien al modelo.

Finalmente, ¿cuál es el número de condición de la matriz R?

Evidentemente, este número depende del valor de n. Hagamos una tabla para los distintos valores de n

In [35]:

```
m = length(t)
ConditionR = zeros(19)
ConditionNormal = zeros(19)
N = zeros(19)
# Para cada n de 1 a 19, calculamos el número de condición de R y de las matrices A'*A

for n = 1:19
    N[n] = n
   A = zeros(m,n+1);
    for i = 1:m
        for j = 1:n+1
            A[i,j] = t[i]^(j-1)
        end
    end
    (Q,R) = QRHouseholder(A)
    ConditionR[n] = cond(R)
    ConditionNormal[n] = cond(A'*A)
end

Table = [N'; ConditionR'; ConditionNormal']'

lap(Table)
```

```
\left[
\begin{array}{ccc}
1.0 & 187.25029863335945 & 35062.67433828433 \\
2.0 & 34918.52335869614 & 1.2193032735740993e9 \\
3.0 & 6.261546153092644e6 & 3.92069609137215e13 \\
4.0 & 1.1261969033630772e9 & 1.2682213500233848e18 \\
5.0 & 2.0774101372219263e11 & 1.5373901344564632e23 \\
6.0 & 3.871924646072298e13 & 2.1068354153073705e26 \\
7.0 & 8.107708514361775e15 & 6.615572427374224e29 \\
8.0 & 1.471594394070287e18 & 1.4763598407746939e34 \\
9.0 & 7.762495610404213e20 & 1.8564396390630998e37 \\
10.0 & 1.1385290757354841e25 & 1.759474976951025e41 \\
11.0 & 1.0331067176441162e27 & 2.8154417452613565e45 \\
12.0 & 3.5852064747051696e28 & 1.3037370407111056e47 \\
13.0 & 1.2007225119784896e31 & 1.2227978374097504e52 \\
14.0 & 1.1743276017330145e34 & 4.6182823185956895e54 \\
15.0 & 9.097797082544422e36 & 8.396470942652052e57 \\
16.0 & 5.2900274683857616e39 & 1.4016749297356005e61 \\
17.0 & 1.8767350905081065e42 & 2.2727191837443505e64 \\
18.0 & 1.2244291370439 85e46 & 6.003779878524312e66 \\
19.0 & 3.2558873635130327e49 & 1.942622974415792e70 \\
\end{array}
\right]
```

n	$cond(R)$	$cond(A^TA)$
1.0	187.25029863335945	35062.67433828433
2.0	34918.52335869614	$1.2193032735740993e9$
3.0	$6.261546153092644e6$	$3.92069609137215e13$
4.0	$1.1261969033630772e9$	$1.2682213500233848e18$
5.0	$2.0774101372219263e11$	$1.5373901344564632e23$
6.0	$3.871924646072298e13$	$2.1068354153073705e26$
7.0	$8.107708514361775e15$	$6.615572427374224e29$
8.0	$1.471594394070287e18$	$1.4763598407746939e34$
9.0	$7.762495610404213e20$	$1.8564396390630998e37$
10.0	$1.1385290757354841e25$	$1.759474976951025e41$
11.0	$1.0331067176441162e27$	$2.8154417452613565e45$
12.0	$3.5852064747051696e28$	$1.3037370407111056e47$
13.0	$1.2007225119784896e31$	$1.2227978374097504e52$
14.0	$1.1743276017330145e34$	$4.6182823185956895e54$
15.0	$9.097797082544422e36$	$8.396470942652052e57$
16.0	$5.2900274683857616e39$	$1.4016749297356005e61$
17.0	$1.8767350905081065e42$	$2.2727191837443505e64$
18.0	$1.224429137043985e46$	$6.003779878524312e66$
19.0	$3.2558873635130327e49$	$1.942622974415792e70$

Notemos que aunque ni R ni A^TA están bien condicionadas, R está mucho mejor condicionada.

U4 5

Considere los siguientes datos, obtenidos de un experimento a intervalos de un segundo, con la primera observación en el tiempo $t = 1.0$:

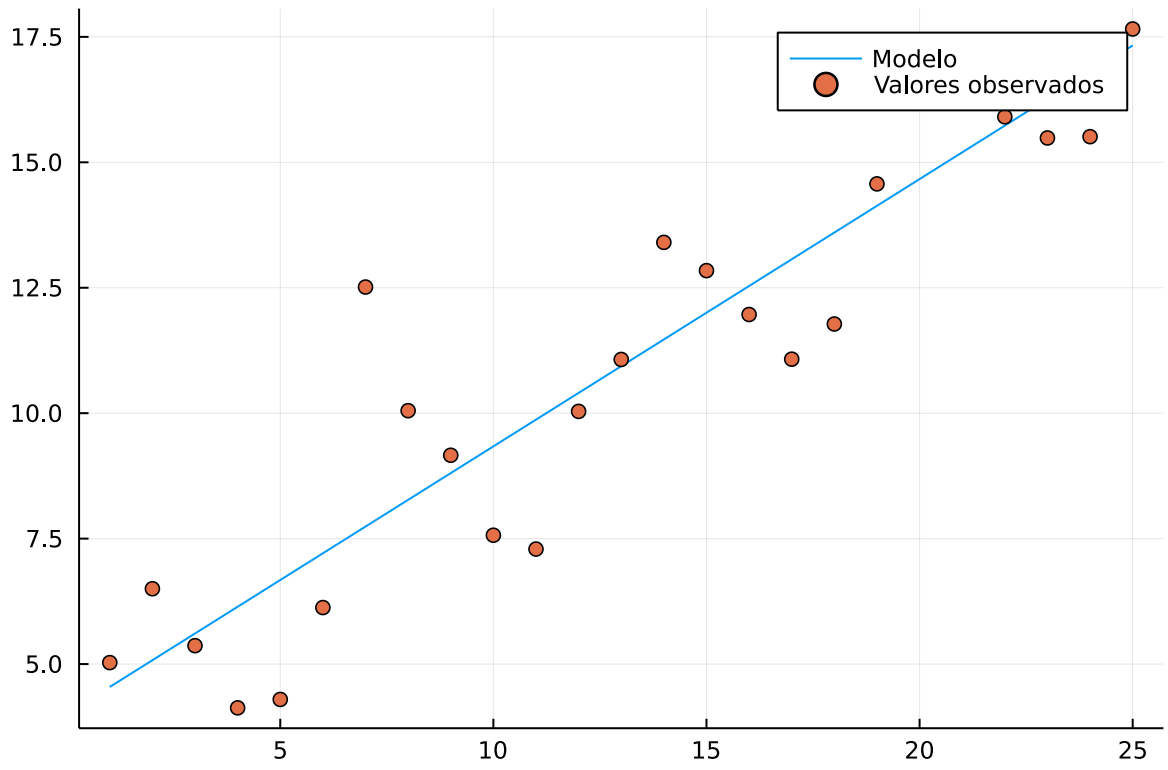
$t: 1-9$	$t: 10-18$	$t: 19-25$
5.0291	7.5677	14.5701
6.5009	7.2920	17.0440
5.3666	10.0357	17.0398
4.1272	11.0708	15.9069
4.2948	13.4045	15.4850
6.1261	12.8415	15.5112
12.5140	11.9666	17.6572
10.0502	11.0765	
9.1614	11.7774	

1. Utilizando las ecuaciones normales, ajuste los datos por una línea recta $y(t) = \beta_1 + \beta_2 t$ y grafique los residuales $y(t_k) - y_k$. Observe que uno de los datos tiene un residual mucho mayor que el resto. Sospechamos que no encaja con el resto de los datos, es decir, es un valor atípico.

In [39]:
```
# Vectores de datos
t = collect(1:25)
y = [ 5.0291; 6.5009; 5.3666; 4.1272 ; 4.2948; 6.1261; 12.5140; 10.0502; 9.1614; 7.5677; 7
    11.0708; 13.4045; 12.8415; 11.9666; 11.0765; 11.7774; 14.5701; 17.0440; 17.0398; 15.90
```

```
    17.6572 ]

#Cantidad de datos
m = length(t)

#La matriz de diseño tendrá dos columnas. Cada renglón de la forma (1 t_i)
A = ones(m,2)
A[:,2] = A[:,2].*t

# Ahora queremos resolver el sistema de ecuaciones normales A^TAc = A^Ty
# Calculamos la factorización de Cholesky de A^T A

(L,U) = FacChol(A'*A)

#Encontramos una solución w_0 a la ecuación lineal Lw = A^Ty mediante sustitución hacia ad
w0 = SolFwd(L,A'*y)

# Encontramos una solución c_0 a la ecuación lineal L^T c = w_0 mediante sustitución hacia
c0 = SolBwd(U,w0)

beta1 = c0[1]
beta2 = c0[2]

# Definimos la función modelo y(t)
function yModel(t)
   beta1+beta2*t
end

# Vector de residuales
Residuales = yModel.(t).-y

# Graficaremos las obsevaciones (t_i,y_i) y la función y(t_i)
#plot(t,[yModel,y])

plot(t, yModel, label = "Modelo ")
scatter!(t,y, label = "Valores observados")
```

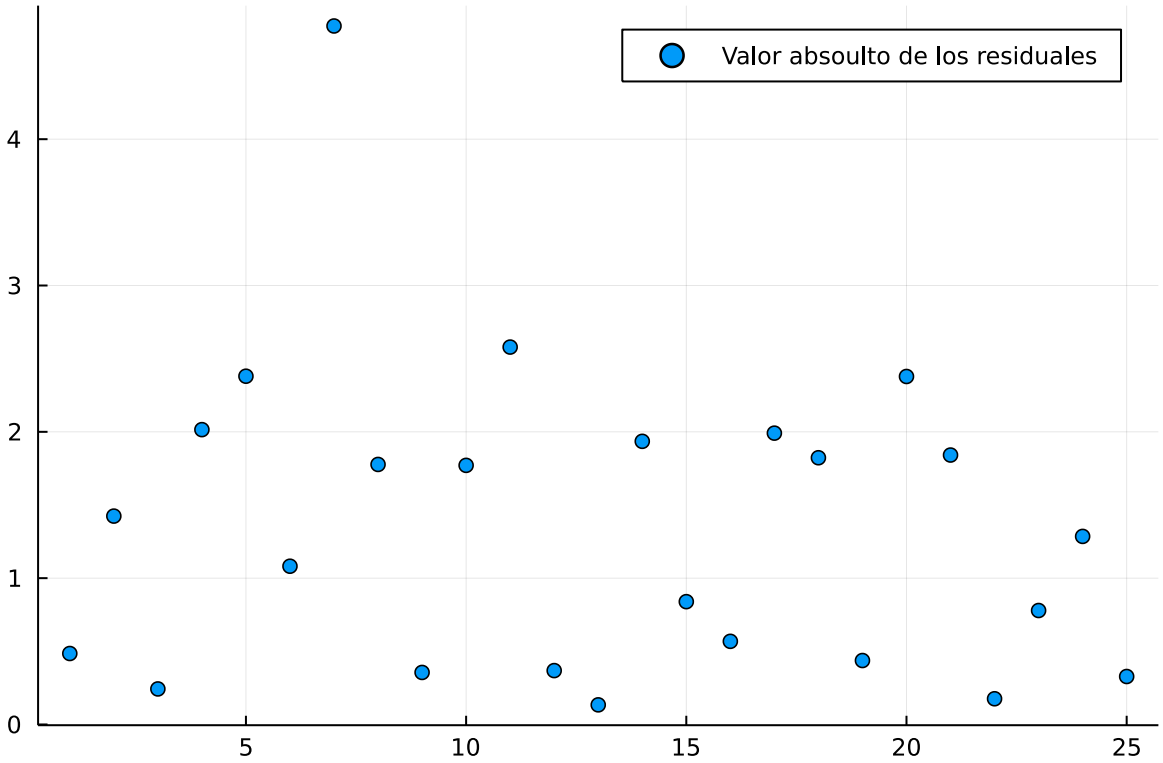
Out[39]:



In [40]:

```
#Graficaremos los residuales
scatter(t, abs.(Residuales), label = "Valor absoulto de los residuales")
```

Out[40]:



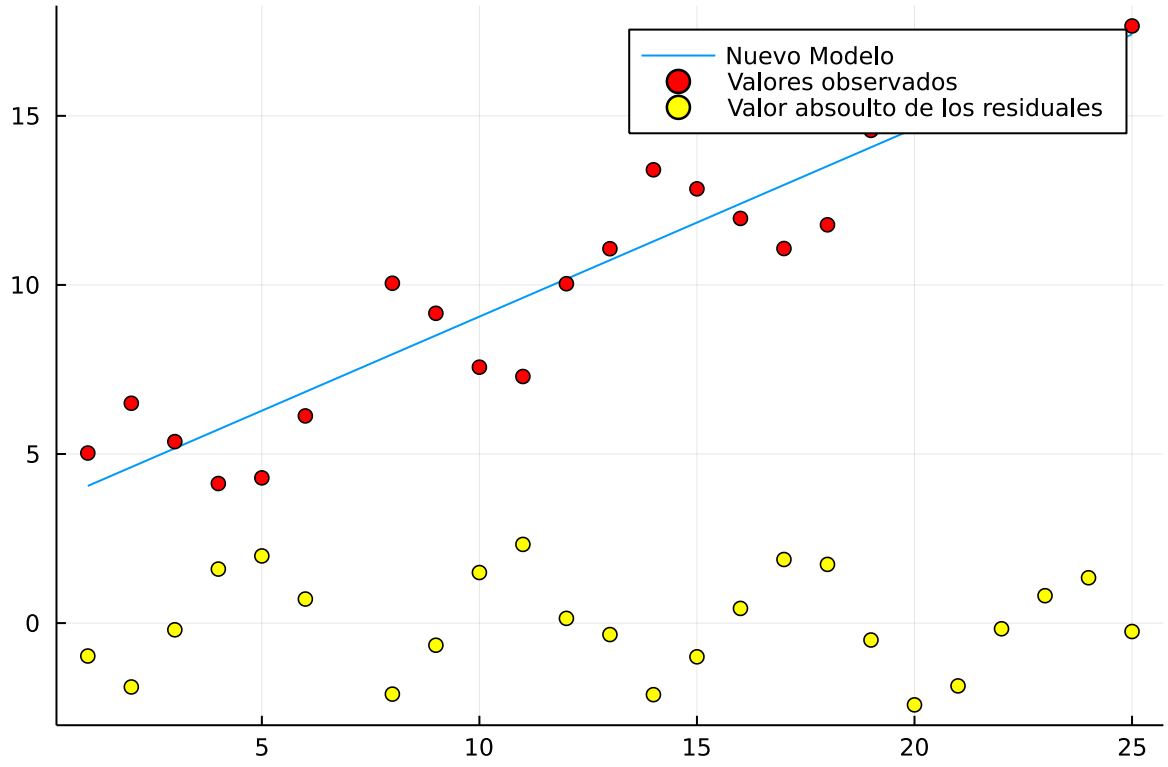
b) Desecehe el valor atípico y ajuste nuevamente los datos con una línea react. Una vez más, grafique los residuales. ¿Qué patrón se observa en la gráfica de residuales? ¿Los residuos parecen aleatorios?

In [43]:

```
# Buscamos el índice del residual con valor absoluto más grande
# y quitamos esa entrada de los vectores t,y

tNew = deleteat!(copy(t),   argmax(abs.(Residuales)) )
yNew = deleteat!(copy(y), argmax(abs.(Residuales)))

# Volvemos a encontrar un modelo pero ahora usando las observaciones de
# tNew, yNew

#Cantidad de datos
m = length(tNew)

#La matriz de diseño tendrá dos columnas. Cada renglón de la forma (1 t_i)
A = ones(m,2)
A[:,2] = A[:,2].*tNew

# Ahora queremos resolver el sistema de ecuaciones normales A^TAc = A^Ty
# Calculamos la factorización de Cholesky de A^T A

(L,U) = FacChol(A'*A)

#Encontramos una solución w_0 a la ecuación lineal Lw = A^Ty mediante sustitución hacia ad
w0 = SolFwd(L,A'*yNew)

# Encontramos una solución c_0 a la ecuación lineal L^T c = w_0 mediante sustitución hacia
c0 = SolBwd(U,w0)

beta1 = c0[1]
beta2 = c0[2]

# Definimos la función modelo y(t)
function yModelNew(t)
   beta1+beta2*t
end
```

```
# Vector de residuales
ResidualesNew = yModelNew.(tNew).-yNew

# Graficamos el modelo,los datos reales y los residuales
plot(tNew, yModelNew, label = "Nuevo Modelo ")
scatter!(tNew, yNew , label = "Valores observados", color = "red")
scatter!(tNew, ResidualesNew, label = "Valor absoulto de los residuales", color = "yellow"
```

Out[43]:



Los residuales ahora son pequeños en valor absoluto. Notemos que parecen seguir una distribución sinusoidal.

c) Para deshacerse de las tendencias de los residuos, ajustar los datos (ya sin el valor atípico) con un nuevo modelo:

$$y(t) = \beta_1 + \beta_2 t + \beta_3 \sin(t) \tag{35}$$

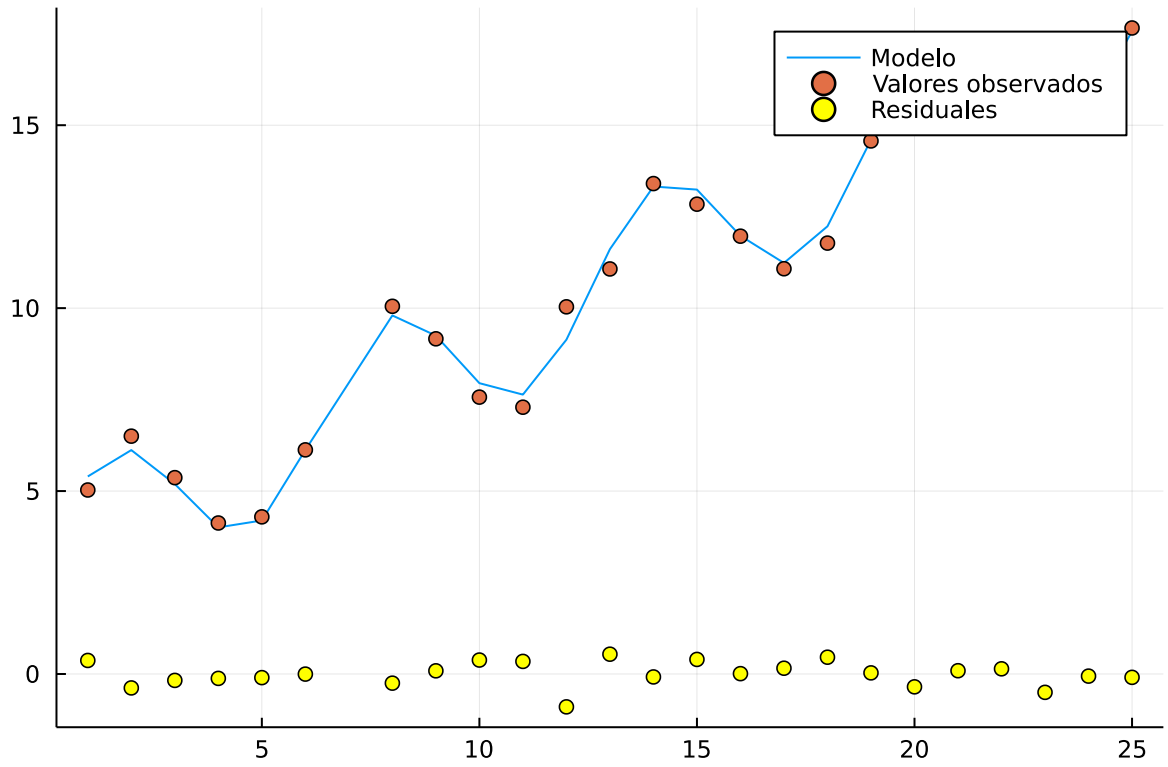
Grafica los residuales. ¿Parecen azarosos ahora?

In [45]:

```
#La matriz de diseño tendrá tres columnas. Cada renglón de la forma
# (1 t_i sin(t_i))
A = ones(m,3)
A[:,2] = A[:,2].*tNew
A[:,3] = sin.(tNew)


# Ahora queremos resolver el sistema de ecuaciones normales A^TAc = A^Ty
# Calculamos la factorización de Cholesky de A^T A

(L,U) = FacChol(A'*A)

#Encontramos una solución w_0 a la ecuación lineal Lw = A^Ty mediante sustitución hacia ad
w0 = SolFwd(L,A'*yNew)

# Encontramos una solución c_0 a la ecuación lineal L^T c = w_0 mediante sustitución hacia
c0 = SolBwd(U,w0)

beta1 = c0[1]
beta2 = c0[2]
```

```
beta3 = c0[3]

# Definimos la función modelo y(t)
function yTrig(t)
   beta1+beta2*t +beta3*sin(t)
end

# Vector de residuales
ResidualesTrig = yTrig.(tNew).-yNew

# Graficaremos las obsevaciones (t_i,y_i), la función modelo y los residuales


plot(tNew, yTrig, label = "Modelo ")
scatter!(tNew,yNew, label = "Valores observados")
scatter!(tNew,ResidualesTrig, label= "Residuales", color = "yellow")
```

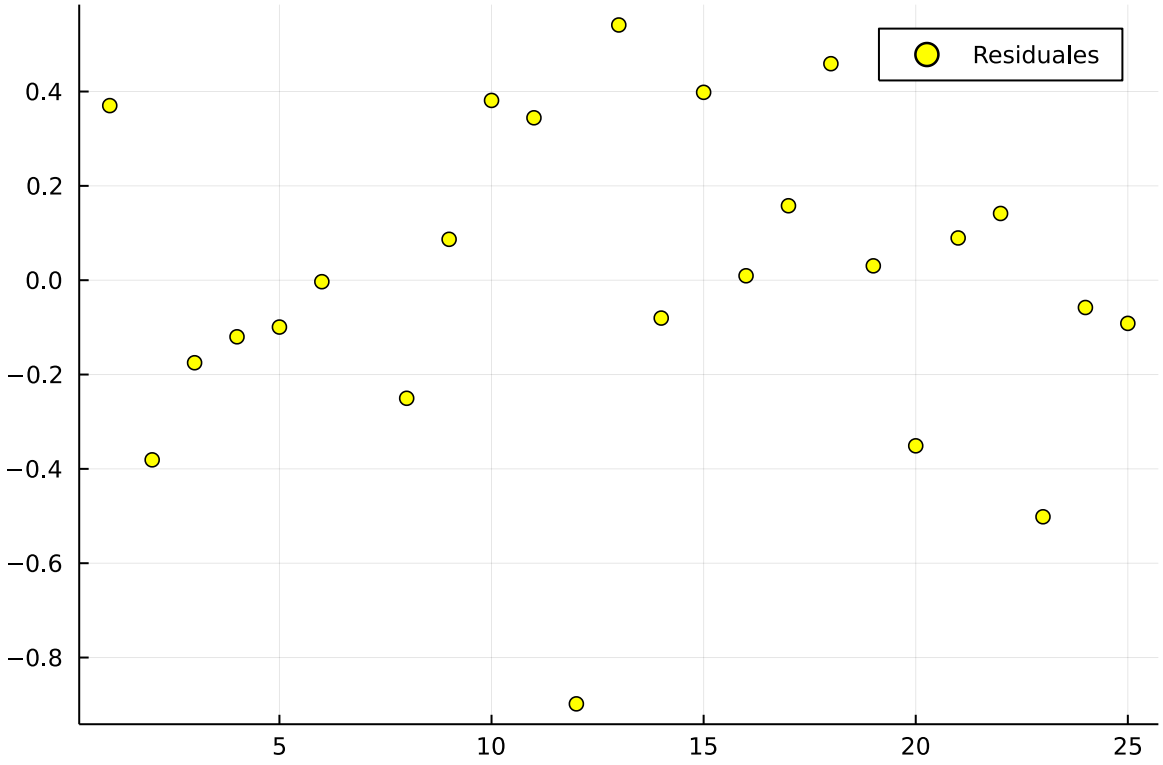
Out[45]:



In [46]:

```
scatter(tNew,ResidualesTrig, label= "Residuales", color = "yellow")
```

Out[46]:



Los residuales ahora sí parecen azarosos.

U4 6 [PENDIENTE]

Un fenómeno que puede presentarse al utilizar el método de Gram-Schmidt es la pérdida de ortogonalidad. En una A.P.F. de 5 dígitos con redondeo, considere la siguiente matriz:

$$A = \begin{bmatrix} 0.70001 & 0.70711 \\ 0.70002 & 0.70711 \end{bmatrix} \tag{36}$$

Observe que la matriz es cercana a una de rango deficiente.

a) Calcule el primer paso de Gram-Schmidt (es decir, $j = 1$) para obtener r_{11} y q_1.

b) Calcule el segundo paso y obtenga r_{12}, r_{22} y q_2.

c) Utilizando los incisos anteriores dé la forma de las matrices Q y R de la factorización QR de A.

d) ¿Las columnas de ! son ortogonales? ¿Cómo afecta el rango deficiente de A a las columnas que se obtuvieron de Q ? Concluya

U4 7

Para matrices que son de rango deficiente es natural observar una pérdida de ortogonalidad en la matriz Q de la factorización QR. Po ejemplo, para la matriz

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{37}$$

Un experimento interesante consiste en perturbar ligeramente la matriz de tal forma que

$$A = \begin{pmatrix} 1 & 1 & 1+\epsilon \\ 1+\epsilon & 1 & 1 \\ 1 & 1+\epsilon & 1 \end{pmatrix} \tag{38}$$

con ϵ pequeño. Tomando esta matriz perturbada, realice lo siguiente para valores de $\epsilon = 10^{-n}$ con $n = 0, 1, \ldots, 15$.

a) La factorización QR de la matriz por el méotodo de Gram-Schmidt y verifique la ortogonalidad de la matriz Q resultante calculando la norma $||Q^T Q - I||$.

b) El mismo proceso pero mediante el método de reflexiones de Householder.

c) El mismo proceso pero mediante el método de rotaciones de Givens.

Elabore una tabla donde ilustre y pueda comparar los resultados. ¿Qué puede concluir acerca de los métodos de factorización QR para matrices cercanas a una de rango deficiente?, ¿cuál preserva mejor la ortogonalidad?

In [57]:

```
############ Gram-Schmidt

# Aquí vamos a guardar los resultados del cálculo
NormasGS = zeros(16)

for n = 0:15
    # Matriz perturbada
    epsilon = 10.0^(-n)
    A = [1 1 1+epsilon; 1+epsilon 1 1; 1 1+epsilon 1]

    # Vamos a encontrar la factorización QR mediante el método de
    # Gram-Schmidt

    #Aquí vamos hay ir guardando los vectores q y las v
    QMatrix =  Matrix(1.0*I,3,3)
    VMatrix = Matrix(1.0*I,3,3)

    VMatrix[:,1] = A[:,1]
    QMatrix[:,1] = VMatrix[:,1]/norm(VMatrix[:,1])

    VMatrix[:,2] = A[:,2]-(QMatrix[:,1]'*A[:,2])*QMatrix[:,1]
    QMatrix[:,2] = VMatrix[:,2]/(norm(VMatrix))

    VMatrix[:,3] = A[:,3]-(QMatrix[:,1]'*A[:,3])*QMatrix[:,1]-(QMatrix[:,2]'*A[:,3])*QMatr
    QMatrix[:,3] = VMatrix[:,3]/norm(VMatrix[:,3])

    NormasGS[n+1] = norm(QMatrix'*QMatrix-I)
end

############## Householder

#Aquí vamos a guardar los resultados del cálculo:
NormasHouse = zeros(16)

for n = 0:15
    # Matriz perturbada
    epsilon = 10.0^(-n)
    A = [1 1 1+epsilon; 1+epsilon 1 1; 1 1+epsilon 1]

    #Factorización QR de A
    (Q,R) = QRHouseholder(A)

    NormasHouse[n+1] = norm(Q'*Q-I)
```

```
end

############### Rotaciones de Givens

### Primero definiremos funciones que nos permiten calcular la factorización QR
# mediante rotaciones de Givens.


# Esta función toma un vector x de nxn y dos índices i,j \leq n
# Regresa una matriz de rotación G tal que y = Gx tiene su j-ésima
# entrada es igual a 0
function Givens(i,j,x)
    n = length(x)
    G = Matrix(1.0*I,n,n)
    G[i,i] = x[i]/(sqrt(x[i]^2+x[j]^2))
    G[j,j] = G[i,i]
    G[j,i] = -x[j]/(sqrt(x[i]^2+x[j]^2))
    G[i,j] = -G[j,i]
    return(G)
end

# Esta función calcula la factorización QR de una matriz A mediante rotaciones de Givens
function QRGivens(A)
    (m,n) = size(A)
    Q = Matrix(1.0*I,m,m)
    B = A
    for j = 1:n-1
        for i = m:-1:j+1
            x = B[:,j]
            T = Givens(j,i,x)  ### Estamos parados en la j-ésima columna y queremos hacer
            B = T*B
            Q = T*Q
        end
    end
    Q = Q'
    R = B
    return(Q,R)
end

#Aquí vamos a guardar los resultados del cálculo
NormasGivens = zeros(16)

for n = 0:15
    # Matriz perturbada
    epsilon = 10.0^(-n)
    A = [1 1 1+epsilon; 1+epsilon 1 1; 1 1+epsilon 1]

    #Factorización QR de A mediante Givens
    (Q,R) = QRGivens(A)

    NormasGivens[n+1] = norm(Q'*Q-I)
end




# Estos comandos son para obtener el código LaTex de la tabla
epsilon = zeros(16)
Table = [epsilon'; NormasGS'; NormasHouse'; NormasGivens']'
lap(Table)
```

```
\left[
\begin{array}{cccc}
0.0 & 0.8284465094562853 & 2.7476618026966064e-16 & 2.5514002453611344e-16 \\
0.0 & 0.9964522285631954 & 9.297082117745284e-16 & 4.2998752849492583e-16 \\
```

```
0.0 & 0.9999626891707288 & 1.5700924586837752e-16 & 3.8459253727671276e-16 \\
0.0 & 0.999999625187669 & 5.564975606931872e-16 & 3.6821932062951477e-16 \\
0.0 & 0.9999999962501875 & 5.338891568193822e-16 & 2.254873622441467e-16 \\
0.0 & 0.9999999999625001 & 5.978733960281817e-16 & 4.577566798522237e-16 \\
0.0 & 0.999999999999625 & 2.9634845277824204e-16 & 4.1725779438208954e-16 \\
0.0 & 0.9999999999999962 & 8.785856370954422e-16 & 3.6821932062951477e-16 \\
0.0 & 1.0000000000000004 & 5.382005793715205e-16 & 3.597533769998862e-16 \\
0.0 & 1.0 & 6.181460191301304e-16 & 3.3537189618722464e-16 \\
0.0 & 1.0 & 7.162874682589104e-16 & 2.603703785810335e-16 \\
0.0 & 1.0 & 5.661048867003676e-16 & 1.7554167342883506e-16 \\
0.0 & 1.000000032870657 & 7.938288718631387e-16 & 3.040470972244059e-16 \\
0.0 & 1.0 & 7.977012308035777e-16 & 2.9634845277824204e-16 \\
0.0 & 1.0 & 2.8576114088871287e-16 & 3.260541871072589e-16 \\
0.0 & 1.0312575156051293 & 5.715222817774257e-16 & 2.8576114088871287e-16 \\
\end{array}
\right]
```

ϵ	$Gram-Schmidt$	$Householder$	$Givens$
1	0.8284465094562853	$2.7476618026966064e-16$	$2.5514002453611344e-16$
10^{-1}	0.9964522285631954	$9.297082117745284e-16$	$4.2998752849492583e-16$
10^{-2}	0.9999626891707288	$1.5700924586837752e-16$	$3.8459253727671276e-16$
10^{-3}	0.999999625187669	$5.564975606931872e-16$	$3.6821932062951477e-16$
10^{-4}	0.9999999962501875	$5.338891568193822e-16$	$2.254873622441467e-16$
10^{-5}	0.9999999999625001	$5.978733960281817e-16$	$4.577566798522237e-16$
10^{-6}	0.999999999999625	$2.9634845277824204e-16$	$4.1725779438208954e-16$
10^{-7}	0.9999999999999962	$8.785856370954422e-16$	$3.6821932062951477e-16$
10^{-8}	1.0000000000000004	$5.382005793715205e-16$	$3.597533769998862e-16$
10^{-9}	1.0	$6.181460191301304e-16$	$3.3537189618722464e-16$
10^{-10}	1.0	$7.162874682589104e-16$	$2.603703785810335e-16$
10^{-11}	1.0	$5.661048867003676e-16$	$1.7554167342883506e-16$
10^{-12}	1.000000032870657	$7.938288718631387e-16$	$3.040470972244059e-16$
10^{-13}	1.0	$7.977012308035777e-16$	$2.9634845277824204e-16$
10^{-14}	1.0	$2.8576114088871287e-16$	$3.260541871072589e-16$
10^{-15}	1.0312575156051293	$5.715222817774257e-16$	$2.8576114088871287e-16$

Lo que podemos concluir de esta tabla es que tanto el método de Givens como el de Householder preservan muy bien la ortogonalidad de la matriz Q, mientras que el método de Gram-Schmidt es muy pobre en este sentido.

En general, el método de Givens parece dar los mejores resultados, pero tanto los resultados que arroja este método como los que arroja el de Householder son satisfactorios.

U4 8

Los datos que sigue a trayectoria de un nuevo planeta detectado por la Agencia Espacial Mexicana (AEM) son:

$$\begin{bmatrix} x & 1.02 & 0.95 & 0.87 & 0.77 & 0.67 & 0.56 & 0.44 & 0.30 & 0.16 & 0.01 \\ y & 0.39 & 0.32 & 0.27 & 0.22 & 0.18 & 0.15 & 0.13 & 0.12 & 0.13 & 0.15 \end{bmatrix}$$

Con el fin de predecir la ubicación del planeta en determinado tiempo es necesario encontrar una buena aproximación a su órbita. Por ello, la AEM recurre a usted con el fin de encontrar tal órbita. Tomando la

ecuación:

$$ay^2 + bxy + cx + dy + e = x^2 \tag{39}$$

a) Encuentra la órbita elíptica que mejor se ajuste utilizando el método de ecuaciones normales para encontrar los coeficientes de la cuadrática y grafique la órbita calculada junto con las observaciones. Calcule el valor residual para este ajuste.

Solución: Buscamos que

$$ay_i^2 + bx_iy_i + c_ix + dy_i + e = x^2, \tag{40}$$

para todos los datos (x_i, y_i), lo cual es equivalente a

$$\begin{pmatrix} y_i^2 & x_iy_i & x_i & y_i & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = x_i^2 \tag{41}$$

para todos los datos (x_i, y_i). Por lo tanto, nuestra matriz de diseño será la matriz de 10×6:

$$A = \begin{pmatrix} y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{10}^2 & x_{10}y_{10} & x_{10} & y_{10} & 1 \end{pmatrix} \tag{42}$$

Queremos resolver el problema

$$A\beta = \begin{pmatrix} x_1^2 \\ \vdots \\ x_{10}^2 \end{pmatrix}, \tag{43}$$

donde

$$\beta = \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} \tag{44}$$

Las ecuaciones normales de este problema son

$$A^TA\beta = A^T \begin{pmatrix} x_1^2 \\ \vdots \\ x_{10}^2 \end{pmatrix} \tag{45}$$

A continuación escribimos la implementación.

In [61]:

```
# Vectores de datos
```

```
x = [1.02; 0.95 ; 0.87 ; 0.77 ; 0.67 ; 0.56 ; 0.44 ; 0.30 ; 0.16 ; 0.01]
y = [0.39 ; 0.32 ; 0.27 ; 0.22 ; 0.18 ; 0.15 ; 0.13 ; 0.12 ; 0.13 ; 0.15]

# Cantidad de datos
m = length(x)
# número de parámetros por estimar
n = 5

# Construimos la matriz de diseño
A = Matrix(1.0*I,m,n)

# Los elementos de la primera columna son y_j^2
A[:,1] = y .* y
# Los elementos de la segunda columna son x_j*y_j
A[:,2] = x .* y
# Los elementos de la tercera columna son x_j
A[:,3] = x
# Los elementos de la cuarta columna son y_j
A[:,4] = y
# Los elementos de la quinta columna son todos unos
A[:,5]  = ones(m)

# Factorización de Cholesky de A^T A: L L^T = A^T A
(L,LT) = FacChol(A'*A)

# Resolvemos el sistema  Lw = A^T * (x_j^2)_j por sustitución hacia adelante
w0 = SolFwd(L,A'*(x .* x))

#Resolvemos el sistema LT \beta = w_0 por sustitución hacia atrás
betaNormal = SolBwd(LT,w0)
```

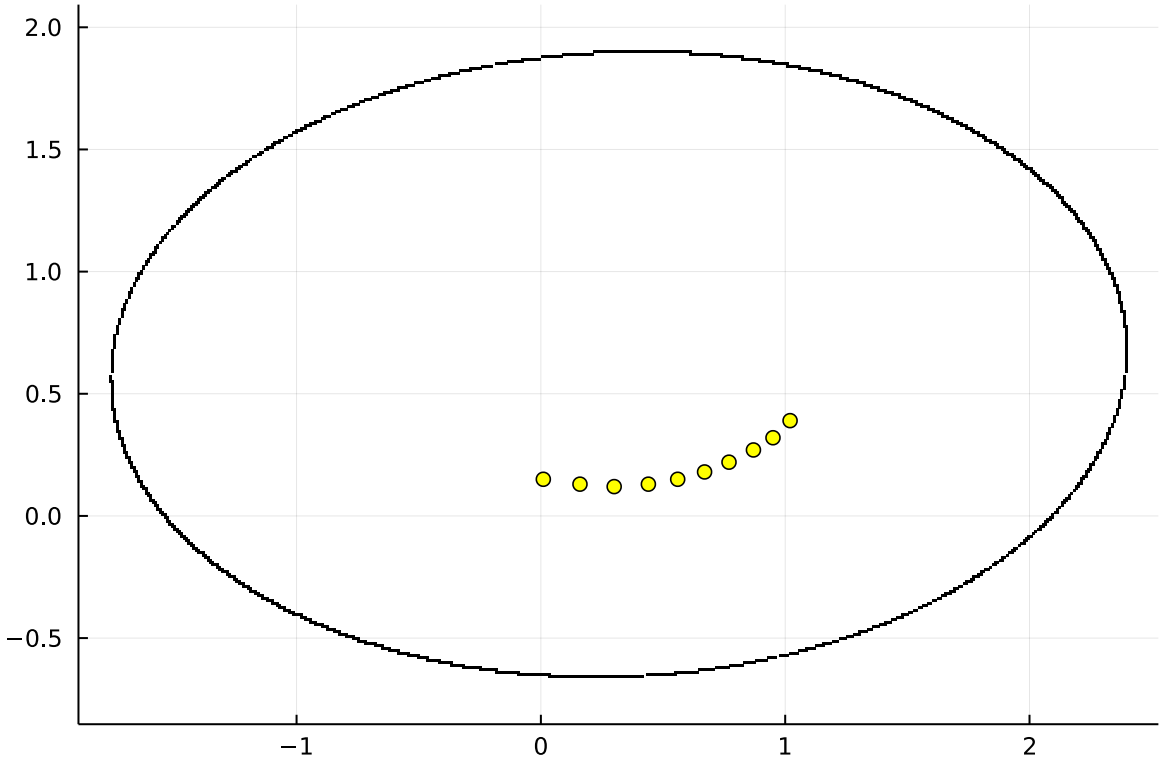
Out[61]:
```
5-element Vector{Float64}:
 -2.635625483707927
  0.14364618259687348
  0.5514469631406753
  3.2229403381056914
 -0.43289427026451116
```

Por lo tanto, la ecuación cuadrática obtenida es

$$x^2 = -2.635625483707927y^2 + 0.14364618259687348xy + 0.5514469631406753x + 3.2229403381056914y - 0.43289427026451116 \tag{46}$$

In [62]:
```
# Ahora queremos graficar los datos (x_i,y_i) juntos con la ecuación cuadrática definida p

fNormal(x,y) = -x^2 + betaNormal[1]*y^2 + betaNormal[2]*x*y + betaNormal[3]*x + betaNormal
rNormal = Eq(fNormal,0)

plot(rNormal, aspect_ratio  = :equal, label = "Modelo Ecuaciones Normales", color = "blue"
scatter!(x,y, label = "Datos reales", color = "yellow")
```

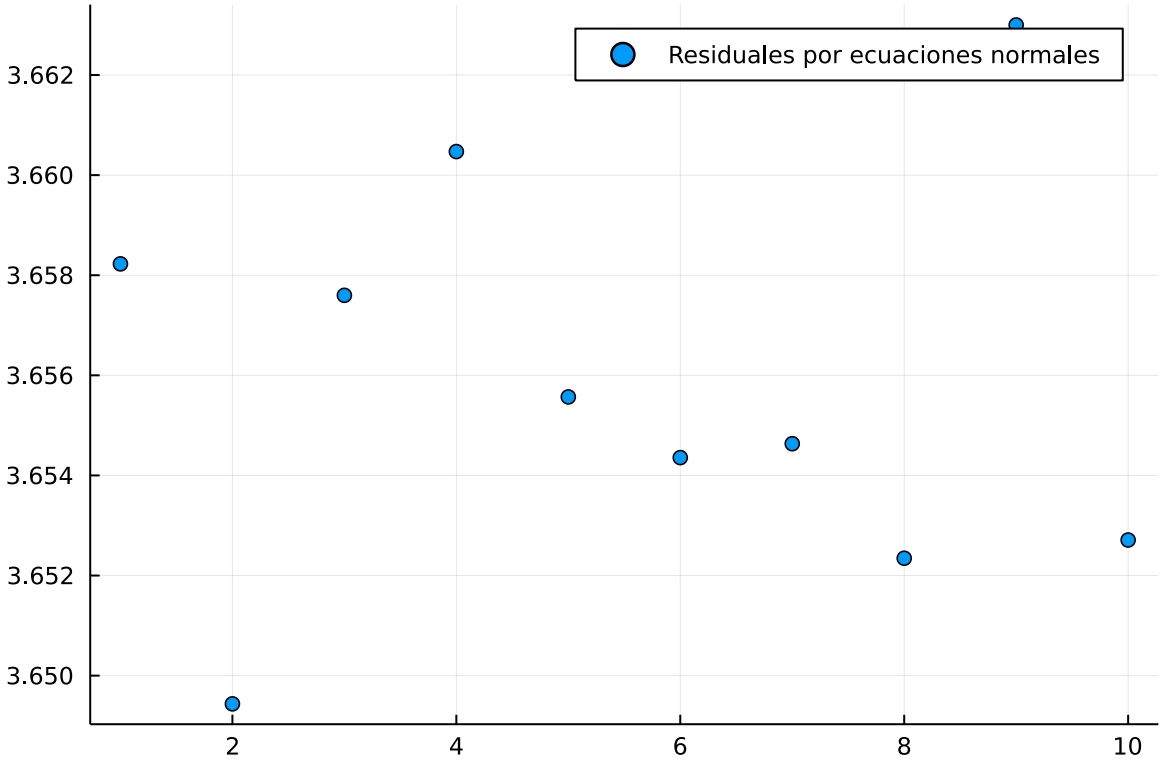
Out[62]:



In [63]:

```
# Ahora calcularemos y graficaremos los residuales
fNormalResiduales = zeros(m)
t = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]

for i = 1:m
    fNormalResiduales[i] = fNormal(x[i],y[i])
end

scatter(t,fNormalResiduales, label = "Residuales por ecuaciones normales")
```

Out[63]:



In [ ]: