# Emacs para ciencias del dato

José A. Alonso Jiménez

4 de Abril de 2018

# Contents

**1 Emacs como entorno de programación** **2**
1.1 Emacs como entorno de programación para Haskell . . . . . 2
1.2 Emacs como entorno de programación para Python . . . . . 2
1.3 Emacs como entorno de programación para R . . . . . 2

**2 Procesamiento de texto con el modo Org** **2**
2.1 Cabecera del documento . . . . . 2
2.2 Tabla de contenidos . . . . . 3
2.3 Estructura del documento: secciones y subsecciones . . . . . 4
2.4 Párrafos, líneas rotas y citas . . . . . 4
2.5 Líneas horizontales . . . . . 5
2.6 Tipos de letra . . . . . 5
2.7 Hiperenlaces . . . . . 6
2.8 Notas al pie de página . . . . . 6
2.9 Listas . . . . . 7
2.10 Tablas . . . . . 8
2.11 Imágenes . . . . . 11
2.12 Expresiones matemáticas (como en LaTeX) . . . . . 11
2.13 Exportaciones . . . . . 13
2.14 Exportación a HTML . . . . . 13
2.15 Bloques literales y de código . . . . . 14
2.16 Inclusión de archivos . . . . . 16

**3 Programación literaria con el modo Org** **17**
3.1 Activación de lenguajes . . . . . 17
3.2 Código en línea . . . . . 17
3.3 Bloques de código . . . . . 17
3.4 Resultados gráficos . . . . . 20

**4 Metaprogramación con Babel** **23**

5 Bibliografía 23

# 1 Emacs como entorno de programación

## 1.1 Emacs como entorno de programación para Haskell

- Ejemplo de edición y evaluación de código Haskell.
  - Se usará el programa Takeuchi.hs.
  - El modo utilizado es Haskell Mode for Emacs.
  - Se evalúa con la pestaña `Haskell / Load file` o `C-c C-l`

## 1.2 Emacs como entorno de programación para Python

- Ejemplo de edición y evaluación de código Python.
  - Se usará el programa Takeuchi.py.
  - El modo utilizado es Elpy, the Emacs Lisp Python Environment.
  - Se evalúa con la pestaña `Elpy / Interactive Python / Send buffer to Python` o `C-c C-c`

## 1.3 Emacs como entorno de programación para R

- Ejemplo de edición y evaluación de código R.
  - Se usará el programa Takeuchi.r.
  - El modo utilizado es ESS (Emacs Speaks Statistics).
  - Se evalúa con la pestaña `ESS / Load file` o `C-c C-l`

# 2 Procesamiento de texto con el modo Org

## 2.1 Cabecera del documento

### 2.1.1 Información del documento

- La parte informativa de este documento es

```
#+TITLE:  Emacs para ciencias del dato
#+AUTHOR: José A. Alonso Jiménez
#+DATE:   4 de Abril de 2018
```

- Para no poner el nombre del autor, se deja vacío el valor del campo `#+AUTHOR:`.
- Para no poner la fecha, se deja vacío el valor del campo `#+DATE:`.

### 2.1.2 Opciones generales para las exportaciones

- En el campo `#+LENGUAJE:` se elige el lenguaje para la exportación, como por ejemplo `es` (español).

- En el campo `#+OPTIONS:` se pueden colocar diferentes comandos, como por ejemplo `toc:nil`, que sirve para que al crear el documento, este no tenga tabla de contenidos.

### 2.1.3 Opciones para las exportaciones a LaTeX

- En el campo `#+LATEX_CLASS_OPTIONS:` se escriben opciones para el procesamiento por LaTeX para exportar a PDF. En este caso las opciones elegidas son

  - `a4paper` (para usar las dimensiones de hoja A4),
  - `12pt` (para el tamaño de la letra) y
  - `twoside` (para escribir en las dos caras).

- En el campo `#+LATEX_HEADER:` se escriben opciones para la cabecera del fichero LaTeX que se generará. En este caso, se ha elegido

  - `\usepackage{a4wide}` (para usar el paquete de las márgenes anchos de las hojas A4).

### 2.1.4 Opciones para las exportaciones a HTML

- En el campo `#+HTML_HEAD:` se escriben opciones para la cabecera del fichero HTML que se generará. En este caso, se ha elegido

  - `<link rel="stylesheet" type="text/css" href="./estilo.css" />` para usar como estilo `estilo.css`.

- En el campo `#+HTML_MATHJAX:` se escriben opciones para el procesamiento de los símbolos matemáticos.

## 2.2 Tabla de contenidos

- Se puede configurar. Por ejemplo,

```
#+OPTIONS: toc:2          (solo a dos niveles en TOC)
#+OPTIONS: toc:nil        (no TOC en todo)
```

## 2.3 Estructura del documento: secciones y subsecciones

- Las cabeceras definen la estructura del documento.
- Las cabeceras comienzan con una o más estrellas en el margen izquierdo. Por ejemplo,

```
* Cabecera de nivel superior
** Segundo nivel
*** Tercer nivel
    cualquier texto
*** Tercer nivel
    más texto

* Otro título de nivel superior
```

- La estructura del documento es un árbol con secciones/subsecciones/...
- Se puede mostrar/ocultar niveles del árbol con `TAB`

## 2.4 Párrafos, líneas rotas y citas

- Los párrafos están separados por al menos una línea vacía. Por ejemplo,

```
Esto es un párrafo
que sigue aquí.

Y esto es otro párrafo
```

produce

> Esto es un párrafo que sigue aquí.
>
> Y esto es otro párrafo

- Para forzar el fin de línea se usa \\ al final de la línea. Por ejemplo,

```
Esta es una línea \\
y esta es otra.
```

produce

> Esta es una línea
> y esta es otra.

- Se puede usar el formato de versos. Por ejemplo,

```
#+BEGIN_VERSE
Caminante, no hay camino
se hace camino al andar.

Antonio Machado
#+END_VERSE
```

produce

> Caminante, no hay camino
> se hace camino al andar.
>
> Antonio Machado

- Se puede incluir citas. Por ejemplo,

```
#+BEGIN_QUOTE
Todo necio confunde valor y precio. ~ Antonio Machado
#+END_QUOTE
```

produce

> Todo necio confunde valor y precio. ~ Antonio Machado

## 2.5 Líneas horizontales

- Se puede insertar líneas horizontales. Por ejemplo,

```
-----
```

produce

---

## 2.6 Tipos de letra

Se puede

- resaltar una palabra escribiéndola entre `*` (como `*ejemplo*` que produce **ejemplo**),
- subrayar una palabra escribiéndola entre `_` (como `_ejemplo_` que produce ejemplo)
- escribir una palabra en cursiva escribiéndola entre `/` (como `/ejemplo/` que produce *ejemplo*).
- escribir una palabra como máquina de escribir escribiéndola entre `~` (como `~ejemplo~` que produce `ejemplo`).
- escribir código escribiéndolo entre `=` (como `=ejemplo=` que produce `ejemplo`).

## 2.7 Hiperenlaces

### 2.7.1 Escritura de enlaces

- Enlace a sitio web:

```
Enlace al [[http://www.cs.us.es][Dpto. de Ciencia de la Computación e I.A.]]
```

produce Enlace al Dpto. de Ciencia de la Computación e I.A.

- Enlace a un fichero:

```
[[./Takeuchi.py][Takeuchy.py]] contiene la definición de la función de Takeuchi en Py
```

produce Takeuchy.py contiene la definición de la función de Takeuchi en Python.

- Enlace a sitio web:

```
http://www.us.es es la página de la Universidad de Sevilla
```

produce `http://www.us.es` es la página de la Universidad de Sevilla

- El formato de enlace general es

```
[[enlace]] o [[enlace][descripción]]
```

- Se pueden gestionar con `Org / Hyperlinks`

### 2.7.2 Manejo de enlaces

- Se crean con `C-c C-l`
- Se abren con `C-c C-o`
- Se gestionan con `Org / Hyperlinks`

## 2.8 Notas al pie de página

Se puede insertar notas al pie de página usando `C-c C-x f`. Por ejemplo, la nota [1]

[1] Este es un ejemplo de nota a pie de página.

## 2.9 Listas

### 2.9.1 Listas no ordenadas

El siguiente código

```
+ punto
+ punto
  - punto2 1
  - punto2 2
+ punto
+ punto
```

produce

- punto
- punto
  - punto2 1
  - punto2 2
- punto
- punto

### 2.9.2 Listas ordenadas

El siguiente código

```
1. Punto 1
2. Punto 2
  1) Punto 2.1
  2) Punto 2.2
     1) Punto 2.2.1
3) Punto 3
```

produce

1. Punto 1
2. Punto 2
   (a) Punto 2.1
   (b) Punto 2.2
       i. Punto 2.2.1
3. Punto 3

### 2.9.3 Listas descriptivas

El siguiente código

```
Definiciones del DRAE:
+ Ciencia :: Conjunto de conocimientos obtenidos mediante la observación
     y el razonamiento, sistemáticamente estructurados y de los que se
     deducen principios y leyes generales con capacidad predictiva y
     comprobables experimentalmente.
+ Dato :: Información dispuesta de manera adecuada para su tratamiento
     por una computadora.
+ Estadística :: Rama de la matemática que utiliza grandes conjuntos de
     datos numéricos para obtener inferencias basadas en el cálculo de
     probabilidades.
+ Matemática :: Ciencia deductiva que estudia las propiedades de los
     entes abstractos, como números, figuras geométricas o símbolos, y
     sus relaciones.
+ Informática :: Conjunto de conocimientos científicos y técnicas que
     hacen posible el tratamiento automático de la información por medio
     de computadoras.
```

produce

Definiciones del DRAE:

**Ciencia** Conjunto de conocimientos obtenidos mediante la observación y el razonamiento, sistemáticamente estructurados y de los que se deducen principios y leyes generales con capacidad predictiva y comprobables experimentalmente.

**Dato** Información dispuesta de manera adecuada para su tratamiento por una computadora.

**Estadística** Rama de la matemática que utiliza grandes conjuntos de datos numéricos para obtener inferencias basadas en el cálculo de probabilidades.

**Matemática** Ciencia deductiva que estudia las propiedades de los entes abstractos, como números, figuras geométricas o símbolos, y sus relaciones.

**Informática** Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras.

## 2.10 Tablas

### 2.10.1 Creación de tablas

Para poder crear tablas fácilmente en modo Org, se sigue los siguientes pasos: se escribe la cabecera de la tabla

```
| Alumno | Materia | Nota |
|-
```

y a continuación se le da a la tecla `TAB`, quedando lo anterior de esta forma

```
| Alumno | Materia | Nota |
|--------+---------+------|
|        |         |      |
```

Se escribe un nombre (por ejemplo, Pedro)

```
| Alumno | Materia | Nota |
|--------+---------+------|
| Pedro  |         |      |
```

Al pulsar el tabulador, el cursor se pone en la siguiente columna, salvo cuando se encuentra en la última columna que entonces crea una nueva fila y se pone en su primera columna.

Con este procedimiento escribimos la tabla

```
|--------+-------------+------|
| Alumno | Materia     | Nota |
|--------+-------------+------|
| Pedro  | Matemáticas |    6 |
| Luis   | Física      |    9 |
|--------+-------------+------|
```

que, al procesar el documento produce

| Alumno | Materia | Nota |
|---|---|---|
| Pedro | Matemáticas | 6 |
| Luis | Física | 9 |

### 2.10.2 Movimiento en las tablas

- `TAB` mueve al siguiente campo y realinea la tabla
- `RET` mueve a la siguiente fila (o la crea si no hay) realinea la tabla
- `C-c C-c` realinea la tabla sin mover el cursor.
- Más opciones en la pestaña `Tbl`

### 2.10.3 Generación de tablas a partir de CSV:

- Seleccionando la siguiente región y pulsando `C-c |`

```
Nombre,Teléfono,Edad
Pedro,12345678,17
Ana,87654321,25
```

produce

```
| Nombre | Teléfono | Edad |
| Pedro  | 12345678 |   17 |
| Ana    | 87654321 |   25 |
```

### 2.10.4 Leyendas y etiquetas

- En las tablas,
  - Se pueden poner leyendas con `#+CAPTION: <valor>`
  - Se pueden poner etiquetas con `#+LABEL: <valor>` y
- Ejemplo

```
#+CAPTION: Tabla de teléfonos y edades
#+LABEL: tab:telefonos
| Nombre | Teléfono | Edad |
| Pedro  | 12345678 |   17 |
| Ana    | 87654321 |   25 |
```

produce

Table 1: Teléfonos y edades

| Nombre | Teléfono | Edad |
|---|---|---|
| Pedro | 12345678 | 17 |
| Ana | 87654321 | 25 |

- Se puede hacer referencia a una tabla. Por ejemplo.

```
Los números de teléfonos están en la [[tab:telefonos][tabla de teléfonos]].
```

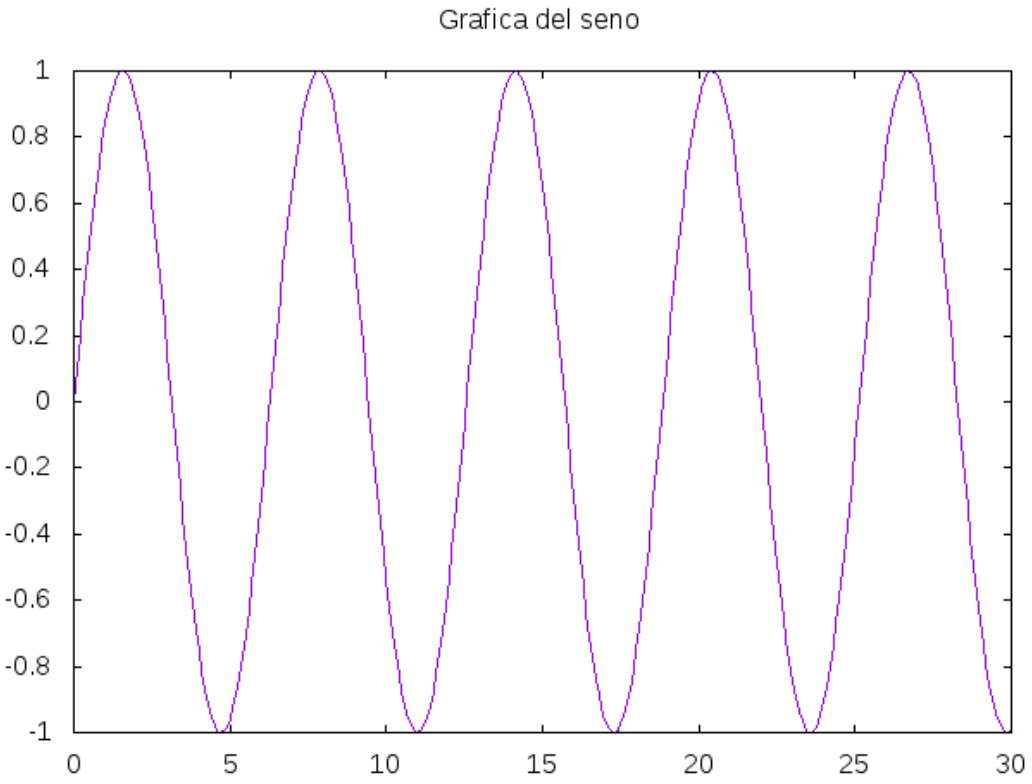
produce Los números de teléfonos están en la tabla de teléfonos.

## 2.11 Imágenes

Para incluir una imagen basta escribir el nombre del fichero que la contiene entre dobles corchetes. Por ejemplo,

```
[[./Grafica_del_seno.png]]
```

produce




## 2.12 Expresiones matemáticas (como en LaTeX)

### 2.12.1 Letras griegas

- Se pueden escribir letras griegas. Por ejemplo, `\alpha, \beta` produce $\alpha$, $\beta$.

### 2.12.2 Subíndices y superíndices

- Se pueden escribir subíndices y superíndices. Por ejemplo, `x_1 = 2^10` produce $x_1 = 2^{10}$

- Se gestiona con `Org / LaTeX`

### 2.12.3 Ecuaciones

El modo Org es muy útil para realizar documentos científicos pues su aprendizaje es rápido y cómodo, y luego tiene fácil exportación a pdf.

Para poder escribir ecuaciones en una línea y de seguido con el resto del texto, se usan los comandos \ y paréntesis de la siguiente manera:

- La expresión `\(x^2+1=0\)` produce $x^2 + 1 = 0$
- Si quisiéramos poner una ecuación en su propia línea en el centro, pondríamos en la siguiente línea el comando siguiente. Por ejemplo, la expresión `\[a^2= b^2 + c^2\]` produce

$$a^2 = b^2 + c^2$$

- Por otro lado, si quisiéramos nombrar la ecuación para, posteriormente hacer referencia a ella en el documento utilizaríamos el comando `#+name:`. Por ejemplo, el texto

```
Sean a, b y c las longitudes de los lados de un triángulo
rectángulo, con \(a > b\) y \(a > c\). Entonces:

#+name: pitagoras
\begin{equation}
a^2 = b^2 +c^2
\end{equation}
```

produce

Sean a, b y c las longitudes de los lados de un triángulo rectángulo, con $a > b$ y $a > c$. Entonces:

$$a^2 = b^2 + c^2 \tag{1}$$

Ahora podemos hacer referencia escribiendo su nombre; por ejemplo, 1 nos hace referencia a la ecuación anterior.

### 2.12.4 Matrices

Se pueden escribir matrices. Por ejemplo,

```
\[M = \left(\begin{matrix}
             a & b \\
             c & d \\
            \end{matrix}
      \right) \]
```

produce

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

## 2.13 Exportaciones

### 2.13.1 Formatos para exportar

- Se puede exportar a

| | |
|---|---|
| c | iCalendar |
| h | HTML |
| l | LaTeX / PDF |
| o | ODT |
| t | Texto plan (ASCII, Latin-1, UTF-8) |

- Se pueden instalar más formatos de exportación.

### 2.13.2 Procedomiento de exportación

- Para exportar, se usa los comandos `Ctrl-c-Ctrl-e` (o `Org / Export/Publish`) y posteriormente se elige el formato deseado. Por ejemplo,
  - `h o` para exportar a HTML y verlo en el navegador
  - `l o` para exportar a PDF y verlo en el visor.

## 2.14 Exportación a HTML

- Se puede exportar código literal para HTML con

```
#+HTML: Código Literal HTML para exportar
```

o

```
#+BEGIN_HTML
Todas las líneas entre estos marcadores son exportados
literalmente
#+END_HTML
```

- Se puede especificar la forma de las tablas. Por ejemplo,

```
#+ATTR_HTML: :border 2 :rules all :frame border
| Nombre | Teléfono | Edad |
|--------+----------+------|
| Pedro  | 12345678 |   17 |
| Ana    | 87654321 |   25 |
```

produce

| Nombre | Teléfono | Edad |
|---|---|---|
| Pedro | 12345678 | 17 |
| Ana | 87654321 | 25 |

### 2.14.1 Exportación a LaTeX/PDF

- Se puede usar `#+LaTeX:` y la construcción `#+BEGIN_LaTeX ... #+END_LaTeX` para añadir código LaTeX literal.
- Se puede poner atributos para modificar la forma de las tablas

exportadas a LaTeX. Por ejemplo,

```
#+ATTR_LaTeX: :center nil :align |l|l|c|
|--------+-------------+------|
| Alumno | Materia     | Nota |
|--------+-------------+------|
| Pedro  | Matemáticas |    6 |
|--------+-------------+------|
| Luis   | Física      |    9 |
|--------+-------------+------|
```

produce

| Alumno | Materia | Nota |
|---|---|---|
| Pedro | Matemáticas | 6 |
| Luis | Física | 9 |

## 2.15 Bloques literales y de código

### 2.15.1 Bloques literales

- Ejemplo de bloque literal pequeño

```
: Prelude> product [1..30]
: 265252859812191058636308480000000
```

produce

```
Prelude> product [1..30]
265252859812191058636308480000000
```

- Ejemplo de bloque literal

```
#+BEGIN_EXAMPLE
Prelude> product [1..30]
265252859812191058636308480000000
#+END_EXAMPLE
```

produce

```
Prelude> product [1..30]
265252859812191058636308480000000
```

- Para crear un bloque de ejemplo, se escribe `<e TAB`

### 2.15.2 Bloques de código

- Ejemplo de bloque de código Python

```
#+BEGIN_SRC python
def tak(x,y,z):
    if x <= y:
        return y
    else:
        return tak(tak(x - 1, y, z),
                   tak(y - 1, z, x),
                   tak(z - 1, x, y))
#+END_SRC
```

produce

```
def tak(x,y,z):
    if x <= y:
        return y
    else:
        return tak(tak(x - 1, y, z),
                   tak(y - 1, z, x),
                   tak(z - 1, x, y))
```

- Ejemplo de bloque de código R

```
#+BEGIN_SRC
tak <- function(x,y,z) {
    if (x <= y) return (y)
                return (tak(tak(x - 1, y, z),
                            tak(y - 1, z, x),
                            tak(z - 1, x, y)))
}
#+END_SRC
```

produce

```
tak <- function(x,y,z) {
    if (x <= y) return (y)
                return (tak(tak(x - 1, y, z),
                            tak(y - 1, z, x),
                            tak(z - 1, x, y)))
}
```

- Para crear un bloque de ejemplo, se escribe `<s TAB`
- Para editar un bloque de código, usar ~C-c '~

## 2.16 Inclusión de archivos

- Se puede incluir ficheros en la exportación. Por ejemplo, con

```
#+INCLUDE: "./Takeuchi.hs" src haskell
```

produce

```
-- La función de Takeuchi en Haskell

tak :: Integer -> Integer -> Integer -> Integer
tak x y z
    | x <= y    = y
    | otherwise = tak (tak (x-1) y z)
                      (tak (y-1) z x)
                      (tak (z-1) x y)

-- Eficiencia
--    > :set +s
--    > tak 16 6 1
--    16
--    (0.01 secs, 154,992 bytes)
--
--    > tak 900 6 1
--    900
--    (0.03 secs, 3,011,208 bytes)
```

- Con ~C-c '~ se visita el fichero incluido.

# 3 Programación literaria con el modo Org

## 3.1 Activación de lenguajes

- Para activar la evaluación en R y en Python hay que añadir a la configuración de emacs

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . t)
   (python     . t)
   (R          . t)
   ))
```

## 3.2 Código en línea

- El código

```
Dos más tres es src_python{return(2+3)}.
```

produce

Dos más tres es 5.

## 3.3 Bloques de código

### 3.3.1 Gestión de bloques de código

- Gestión de bloque se código
  - Se crean conjunto con `<s TAB` y el nombre del lenguaje.
  - Se edita con ~C-c '~
  - Se evalúa con `C-c C-c`

### 3.3.2 Bloques de cálculo de expresiones

- Bloque de R: La evaluación de

```
#+BEGIN_SRC R
a <- 3
2 * a
#+END_SRC
```

produce

```
#+RESULTS:
: 6
```

- Bloque de Python: La evaluación de

```
#+BEGIN_SRC python
b = 5
return (3 * b)
#+END_SRC
```

produce

```
#+RESULTS:
: 15
```

### 3.3.3 Bloque con definiciones de funciones

- Programa Python: La evaluación de

```
#+BEGIN_SRC python
def g(x):
  if x > 0:
    return "Positivo"
  else:
    return "Negativo"

return g(5)
#+END_SRC
```

produce

```
#+RESULTS:
: Positivo
```

### 3.3.4 Bloque con tablas como resultado

- Ejemplo con tabla como resultado: La evaluación de

```
#+BEGIN_SRC R
  cuadrado <- function(x)
  {
    x * x
  }

  cuadrado(1:10)
#+END_SRC
```

produce

```
#+RESULTS:
|   1 |
|   4 |
|   9 |
|  16 |
|  25 |
|  36 |
|  49 |
|  64 |
|  81 |
| 100 |
```

### 3.3.5 Bloques con vectores como resultado

- Ejemplo con salida vectorial: La evaluación de

```
#+BEGIN_SRC R :results output
  cuadrado <- function(x)
  {
    x * x
  }

  cuadrado(1:10)
#+END_SRC
```

produce

```
#+RESULTS:
:  [1]   1   4   9  16  25  36  49  64  81 100
```

### 3.3.6 Bloques con comentarios

- Ejemplo de programa con comentarios. La evaluación de

```
#+BEGIN_SRC R :results output
  x <- 1:10
  x
  cuadrado <- function(x) {
    cat("Función para calcular cuadrados.\n")
    x * x
  }

  cuadrado(1:10)
#+END_SRC
```

produce

```
#+RESULTS:
:  [1]  1  2  3  4  5  6  7  8  9 10
: Función para calcular cuadrados.
:  [1]   1   4   9  16  25  36  49  64  81 100
```

### 3.3.7 Trabajo con sesiones

- La evaluación de

```
#+BEGIN_SRC R :session *R1* :results none
  cuadrado <- function(x) {
    x * x
  }
  x <- 1:10
#+END_SRC

#+BEGIN_SRC R :session *R1* :results output
  cuadrado(x)
#+END_SRC
```

produce

```
#+RESULTS:
:  [1]   1   4   9  16  25  36  49  64  81 100
```

## 3.4 Resultados gráficos

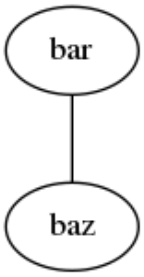
### 3.4.1 Grafos with Graphviz

- Graphviz es un programa de visualización de grafos
- Para activarlo en el modo Org hay que añadir `(dot . t)` a los valores de `org-babel-load-languages`
- Ejemplo de grafo no dirigido: La evaluación de

```
#+BEGIN_SRC dot :file imagenes/grafo.png :cmdline -Tpng
graph foo {
        "bar" -- "baz";
}
#+END_SRC
```

produce

```
#+RESULTS:
[[file:imagenes/grafo.png]]
```

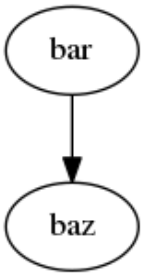



- Ejemplo de grafo dirigido: La evaluación de

```
#+BEGIN_SRC dot :file imagenes/grafoDirigido.png :cmdline -Tpng
digraph foo {
        "bar" -> "baz";
}
#+END_SRC
```

produce

```
#+RESULTS:
[[: file:imagenes/grafoDirigido.png]]
```




### 3.4.2 Gráficos con R

- Tabla de datos

```
#+NAME: temp
| Mes   | Grados  |
|-------+---------|
|    01 |     3.8 |
|    02 |     4.1 |
|    03 |     6.3 |
|    04 |     9.0 |
|    05 |    11.9 |
```

```
|    06 |    15.1 |
|    07 |    17.1 |
|    08 |    17.4 |
|    09 |    15.7 |
|    10 |    11.8 |
|    11 |     7.7 |
|    12 |     4.8 |
```

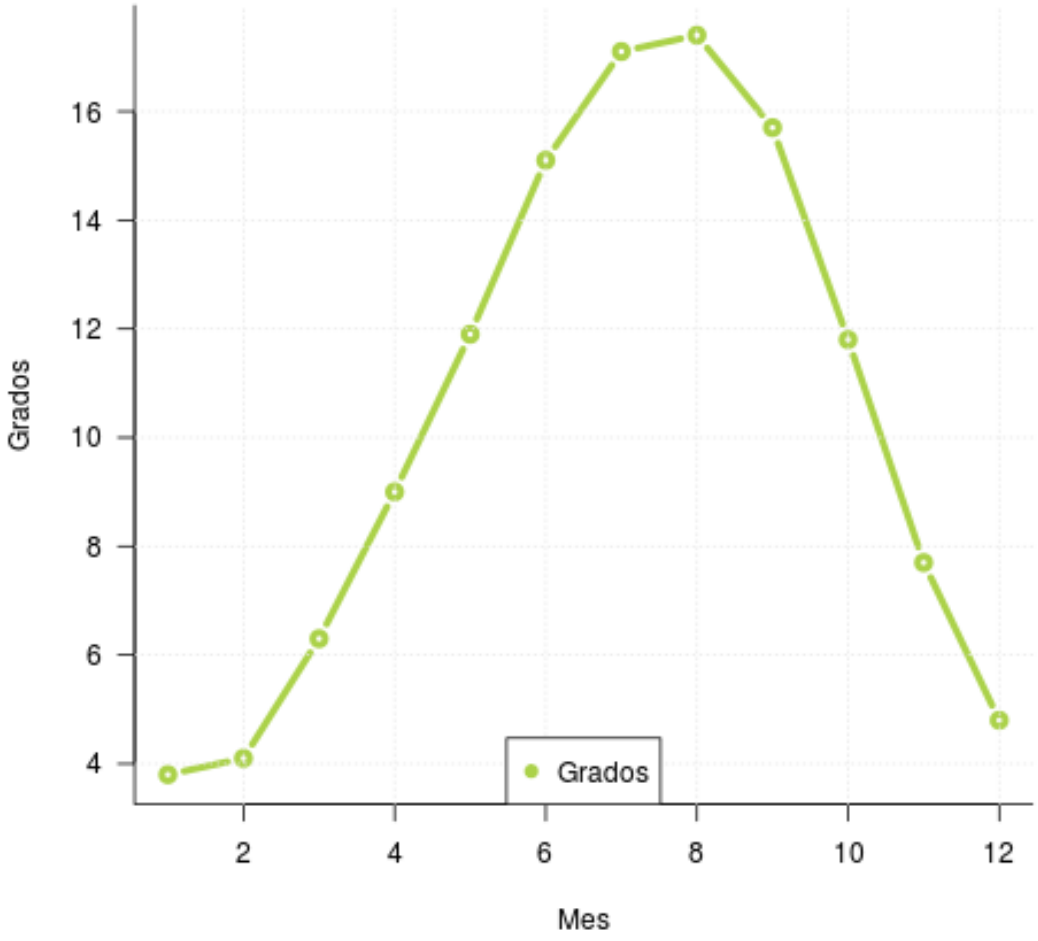
produce

| Mes | Grados |
|---|---|
| 01 | 3.8 |
| 02 | 4.1 |
| 03 | 6.3 |
| 04 | 9.0 |
| 05 | 11.9 |
| 06 | 15.1 |
| 07 | 17.1 |
| 08 | 17.4 |
| 09 | 15.7 |
| 10 | 11.8 |
| 11 | 7.7 |
| 12 | 4.8 |

- La evaluación de

```
#+NAME: dibujo-R
#+BEGIN_SRC R :var data=temp :results graphics :file imagenes/dibujo.png :exports res
plot(data, type="b", bty="l", col=c("#ABD249"), las=1, lwd=4)
grid(nx=NULL, ny=NULL, col=c("#E8E8E8"), lwd=1)
legend("bottom", legend=c("Grados"), col=c("#ABD249"), pch=c(19))
#+END_SRC
```

produce




# 4 Metaprogramación con Babel

- Se puede hacer metaprogramción con Babel como se muestra en este fichero.

# 5 Bibliografía


- Getting started with Emacs text editor. ~ Sachin Patil
- How to create LaTeX documents with Emacs. ~ Sachin Patil
- A multi-language computing environment for literate programming and reproducible research. ~ E. Schulte, D. Davison, T. Dye, C. Dominik

- Using Emacs, Org-mode and R for research writing in social Sciences. ~ Vikas Rawal
- Org-mode and R: An introduction. ~ Erik Iverson
- Emacs for Data Science. ~ Robert Vesco
- Emacs for Data Science. ~ Ahsan Ijaz