Curso de LaTeX

Editores: Andrés Miniguano Trujillo y Milton Torres España
AsoiMat
Escuela Politécnica Nacional
Email: andres.miniguano@epn.edu.ec y milton.torres@epn.edu.ec
Publicación: 5 de abril de 2017
GitHub: Capítulo 4

Capítulo 4

Escritura de código

1 Código con y en LaTeX

En la carpeta encontrarás los archivos `Prueba_personalizada.cpp` y `Estimador_pi.Rnw`.

El primero es un archivo de C++, si lo abres podrás observar que consiste en un generador automático de pruebas para la materia de Cálculo en una variable. Analiza el código y mira en qué cambia respecto a la escritura de un archivo de texto sencillo.

El segundo es un archivo de `Sweave`, un paquete del lenguaje R que tiene su propio módulo en RStudio. Si lo abres, podrás observar que RStudio también lo puedes usar como un editor de LaTeX. En este caso, el documento consiste en una serie de pruebas con variables aleatorias para simular el valor de π. Si compilas varias veces el documento, obtendrás diferentes resultados sobre la estimación de π.

Los ejemplos anteriores muestran cómo se puede integrar a LaTeXcon otros lenguajes de programación y obtener documentos interactivos. El proceso inverso es bastante sencillo puesto que ya conoces a los comandos `\texttt{...}` y `\verb||`. También existe el entorno `verbatim` y te permite introducir código sin formato. Por ejemplo, las primeras líneas del programa en C++ son

```
#include <vector>
#include <sstream>
#include <string>
#include <fstream>
#include <iostream>
using namespace std;

void personalizador(int num);

int num;
int main()
{
    cout << "Generador de prueba personalizada" << endl << endl;
```

Algo que sería interesante es que este código tenga colores semejantes a los que puedes visualizar en un editor de C++, o tener la opción de añadir numerales, una descripción, etc. Ese es el objeto de la siguiente sección.

2 Introducción al paquete `listings`

El paquete `listings` te permite extender la funcionalidad del ambiente `verbatim`, incluyendo una alta gama de opciones que permiten controlar desde el tipo de lenguaje usado hasta el grosor de un bordillo alrededor de un código.

En esta lección te presentaremos cuatro formas diferentes de introducir código con el mismo paquete. Fíjate que como este mismo documento usa el paquete, su alcance no se limita a un único lenguaje de programación.

2.1 C++

Un entorno de código con `listings` tiene un conjunto de opciones `lstset` el cual puedes modificar a tu conveniencia:

```
\lstset
{
   language          = C++,
   breaklines        = true,
   basicstyle        = \tt\footnotesize,
   keywordstyle      = \color{azul},
   identifierstyle   = \color{naranja},
   commentstyle      = \color{verde},
   showstringspaces  = false,
   numbers           = left,
   numberstyle       = \tiny\color{gray},
   numbersep         = 10pt,
   frame             = single,
   rulecolor         = \color{black!20},
   tabsize           = 3,
   texcl             = true,
}
```

Ahora puedes generar un entorno `lstlistings` y, al igual que con `verbatim`, pegar tu código directamente desde un archivo. Si quieres añadir una leyenda a tu código, puedes añadir la siguiente opción:

```
\begin{lstlisting}{caption = {}}
```

Por ejemplo

Código 1: Código en C++ con listings

```
#include <vector>
#include <sstream>
#include <string>
#include <fstream>
#include <iostream>
using namespace std;

void personalizador(int num);

int num;
int main()
{
    cout << "Generador de prueba personalizada" << endl << endl;
    // Y este es un comentario adicional que tiene texto matemático: a^2
```

Fíjate que en la línea 14 tienes texto matemático, esto lo activaste con la opción `texcl = true` y necesariamente debes escribir el texto entre símbolos de dólar.

2.2 R

El siguiente bloque de código aproxima numéricamente la integral de línea compleja de una función diferenciable dentro de la región limitada por C, salvo en el punto $z = 0$:

```
\lstset
{
   language          = R,
   breaklines        = true,
   basicstyle        = \tt\footnotesize,
   keywordstyle      = \color{da},
   commentstyle      = \color{db},   stringstyle = \color{mauve},
   identifierstyle   = \color{naranja},
   showstringspaces  = false,
   numbers = left,   numberstyle = \tiny\color{gray},   numbersep = 10pt,
   rulecolor         = \color{black!20},
   tabsize           = 3,
   texcl             = true,
   morekeywords      = {*,...}
   basicstyle        = \mlttfamily,
}
```

Código 2: Aproximación numérica con R

```
# Integral de línea aproximada de f sobre C.
f <- function(s){
   z <- 2*pi*s*(0+1i)
   z <- (2*pi*(0+1i))*exp(z - exp(-z))*sin(exp(-z))
   z
}
a <- integrate( function(s) Re(f(s)), 0, 1)$value
b <- 1i * integrate( function(s) Im(f(s)), 0, 1)$value
a+b
```

2.3 Mathematica

Los códigos en Mathematica son simplificados y por lo tanto el resultado es bastante compacto. El mismo código en R para la integral de línea lo puedes reducir a una línea, en dicho caso se te puede ocurrir prescindir de los bordes laterales del recuadro de código.

```
\lstset
{
   language          = Mathematica,
   breaklines        = true,
   basicstyle        = {\sffamily\footnotesize},
   keywordstyle      = \color{da},
   commentstyle      = \color{db},   stringstyle = \color{mauve},
   identifierstyle   = \color{naranja},
   showstringspaces  = false,
   numbers = left,   numberstyle = \tiny\color{gray},   numbersep = 10pt,
   rulecolor         = \color{black!20},
   tabsize           = 3,
   texcl             = true,
   morekeywords      = {*,...}
   basicstyle        = \mlttfamily,
   frame             = {lines},
   columns           = flexible,
}
```

Nota que los únicos cambios se dieron en el lenguaje y en las últimas dos líneas.

Código 3: Aproximación numérica con Mathematica

```
*Integral exacta de $f$ sobre $C$.      <- limitaciones del texto matemático*

Integrate[ 2*Pi*I* Exp[2*Pi*I*t] * Exp[-Exp[-2*Pi*I*t]] * Sin[Exp[-2*Pi*I*t ]], {t, 0, 1}]
```

2.4 LaTeX

La configuración que se usó para los ejemplos de código en estos documentos es la siguiente:

```
\lstset
{
   language=[LaTeX]TeX,
   breaklines=true,
   basicstyle=\tt\footnotesize,
   keywordstyle = \color{azul!90!db},
   identifierstyle = \color{naranja!80!dh},
   showstringspaces = false,
   frame = single,
   rulecolor = \color{black!20},
   tabsize = 3,
}
```

Obtienes el siguiente resultado:

```
\documentclass[11pt]{report}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish, es-nolayout, es-nodecimaldot, es-tabla]{babel}
\usepackage{amsmath, amsfonts, amssymb, amsthm}

\title{Un documento}
\date{}

\begin{document}
   \maketitle

   Un espacio de Hilbert $H$ es un [...]  % 2^2 aquí también funciona el texto matemático
\end{document}
Esto de aquí ya no se imprime wiiii.
```

2.5 MATLAB

Para insertar código en MATLAB, es necesario incluir los paquetes `bigfoot` y `matlab-prettifier` con las opciones `numbered` y `framed`. Además de unas configuraciones adicionales que incluimos a continuación.

```
\let\ph\mlplaceholder
\lstMakeShortInline"
\lstset{
  style              = Matlab-editor,
  basicstyle         = \footnotesize\mlttfamily,
  escapechar         = ",
  mlshowsectionrules = true,
  identifierstyle    = \color{black},
}
```

En este caso puedes revisar el código de inicialización para la aproximación de la solución de una EDO de convección – advección.

Código 4: Diferencias finitas sin estabilización

```
% Esquema de diferencias finitas para aproximar la solución del problema
% -αv''(x)+βv'(x)=0, en x∈(0,1),
% sujeto a
% v(0)=0    y    -αv'(1)+βv(1)=-βu_0.

clear all;      clc

n = input('n = ');      alfa = input('alfa =  ');      beta = input('beta = ');
u = input('u_0 = ');
```