Knitr Motores de lenguaje / Language Engines - Programación de aplicaciones para Internet y la Nube.

Andres Julian Moreno M. Cód. 20152195015

June 1, 2016

El presente documento dinámico es generado para el informe de programación literaria para informes de experimentos computacionales en la nube.

Es una herramienta la cual tiene por objetivo la generación de documentos dinámicos, esta herramienta funciona en el lenguaje R, y permite trabajar desde la interfaz Rstudio. Permitiendo integrar las funcionalidades de R en documentos de tipo científico y académicos LATEX, Markdown y también en documentos o publicaciones sobre HTML, a través de la programación literaria busca fortalecer la reproducibilidad de la investigación.

Knitr permite integrar R con otros lenguajes de programación como $python, Perl, C, C++, shell, Awk, SAS, Scala, Haskell, Graphviz, TikZyCoffeescript$ en un unico documento dinámico.

Por ejemplo vamos a ejecutar un trozo de R, sobre este documento dinámico de tipo R Markdown

```
library(knitr)
set.seed(1234)
rnorm(5)

## [1] -1.2070657  0.2774292  1.0844412 -2.3456977  0.4291247
```

Con el siguiente comando observamos cuales son todos los lenguajes soportados por Knitr:

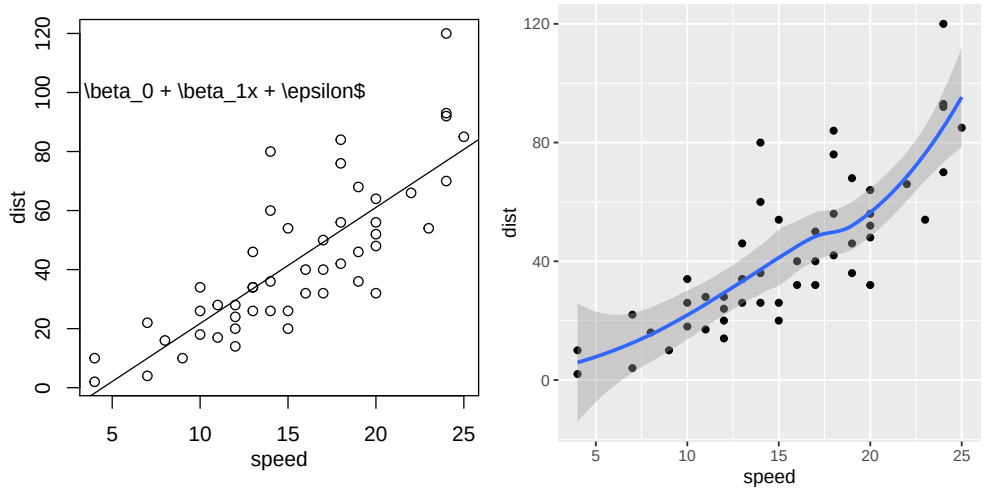
```
names(knit_engines$get())

##  [1] "awk"       "bash"      "coffee"    "gawk"      "groovy"    "haskell"   "lein"
##  [8] "mysql"     "node"      "perl"      "psql"      "python"    "Rscript"   "ruby"
## [15] "sas"       "scala"     "sed"       "sh"        "stata"     "zsh"       "highlight"
## [22] "Rcpp"      "tikz"      "dot"       "c"         "fortran"   "asy"       "cat"
## [29] "asis"      "stan"      "block"
```

Ahora un Ejemplo de R un poco complejo incluyendo un par de gráficas, integrando el paquete de graficos para R ggplot2

```
## tomado de http://yihui.name/knitr/
fit <- lm(dist ~ speed, data = cars)
par(mar = c(4, 4, 1, 0.1), mgp = c(2, 1, 0))
```

```r
with(cars, plot(speed, dist, panel.last = abline(fit)))
text(10, 100, "$Y = \\beta_0 + \\beta_1x + \\epsilon$")
library(ggplot2)
qplot(speed, dist, data = cars) + geom_smooth()
```



Soporte

Como los *chunkhooks* (trozos de código), los motores o lenguajes están soportados por *Knitr* y *R*.

Las funciones chunk se encargan de ejecutar el chunk (trozo de código) en el lenguaje determinado por el usuario y ejecutar el código en R, obteniendo resultados y mostrando su salida, para la mayoría de los casos, el código se comunica con los programas externos mediante la funciona del sistema, *system*().

Como Funciona Knitr

Todos los lenguajes que soporta Knitr son guardados en la función $knit_engines$, la cual cuenta con métodos *get*() y *set*(), como la ejecución de funciones chunk hooks y chunk options, entonces si por ejemplo queremos generar un documento en Python se debe seleccionar la función 'engine' y asignarle la opción de python, como por ejemplo $engine =' python'$ o $engine =' awk'$ si queremos ejecutar otro lenguaje. El objeto $knit_engines$ almacena una serie de funciones con nombre del lenguaje soportado.

Ejemplo en *Python*

A continuación se muestran varios ejemplos de programas sencillos hechos en *Python* y ejecutados en *R*, gracias a *Knitr* y los *chunkengine*.

Programa 1: Mostrar numeros Naturales

```r
n = 1
while n <= 30:
    print n,
## Imprime los primeros 30 numeros naturales
    n += 1
```

```
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

Programa 2: Sumas

```
## Imprime la suma de 1+2+3+4+....+50
h = range(1, 51)
print sum(h)

## 1275
```

Programa 3: Hola Mundo

```
## Imprime Hola Mundo
x= "Hola Mundo"
print x

## Hola Mundo
```

Programa 4: Importando Librerias y el $seno(\pi)$

```
import math
## calcular el seno de pi
y= math.sin(math.pi)
print y

## 1.22464679915e-16
```

Programa 5: Numeros Primos en una sola linea

```
#Lista de numeros primos entre 1 y 100 en una sola linea
c = [i for i in xrange(2,101) if (i%2!=0 or i==2) and (i%3!=0 or i==3)
  and (i%5!=0 or i==5) and (i%7!=0 or i==7)]
print c
#Tomado de https://gist.github.com/developingo/2772442

## [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

En los anteriores 5 ejemplos observamos como funciona Language Engines con Python, en el primer segmento muestra el código que se ejecuta en python, y en el siguiente recuadro muestra el resultado de los comandos ejecutados en el lenguaje seleccionado (##*resultado*).

Hola Mundo en diferentes Lenguajes con Knitr

Perl

```
$test = "Hola Mundo";
$test =~ s/j/h/;
print $test

## Hola Mundo
```

Awk

```
BEGIN { print "Hola mundo"; exit }

## Hola mundo
```

Ruby

```
x = 'Hola Mundo!'
print x

## Hola Mundo!
```

Bash

```
echo Hola Mundo!!!

## Hola Mundo!!!
```

Python

```
x = 'Hola, Hola Mundo, Hola Internet!'
print(x)
print(x.split(' '))

## Hola, Hola Mundo, Hola Internet!
## ['Hola,', 'Hola', 'Mundo,', 'Hola', 'Internet!']
```

En los anteriores ejemplos se ha ejecutado la función básica de imprimir una cadena de caracteres, se ha realizado el famoso ejemplo *HolaMundo*, realizado cuando uno se inicia en un nuevo lenguaje de programación.

Los ejemplos están basados en `http://yihui.name/knitr/demo/engines/`

A continuación se muestra un ejemplo tratando una base de datos

Knitr + Python + R + ggplot2

En este Ejemplo se presenta la interacción entre Knitr, Python, R y ggplot2.

Definiendo X en python:

```
def f(x):
  return x + 2
f(2)
```

Este ejemplo esta basado en [ggplot2] `http://github.com/hadley/ggplot2` y [notedown]`https://github.com/aaren/notedown`.

Se carga la libreria ggoplot en R, con el siguiente chunk

```
library(ggplot2)
```

Cargando una base de datos ejemplo

Usaremos la base de datos de flores; [iris]`http://stat.ethz.ch/R-manual/R-patched/library/datasets/html/iris.html`.

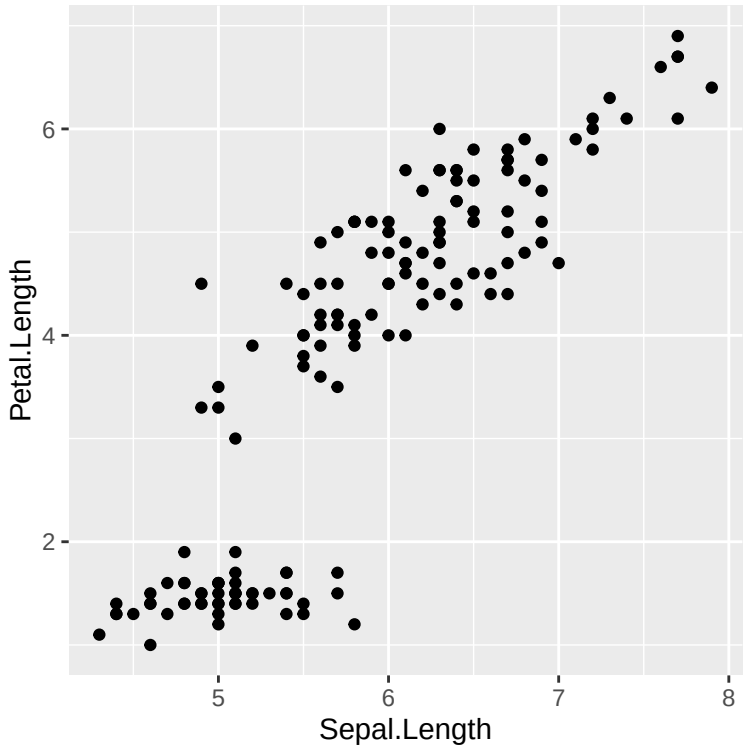
```
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Gráficando

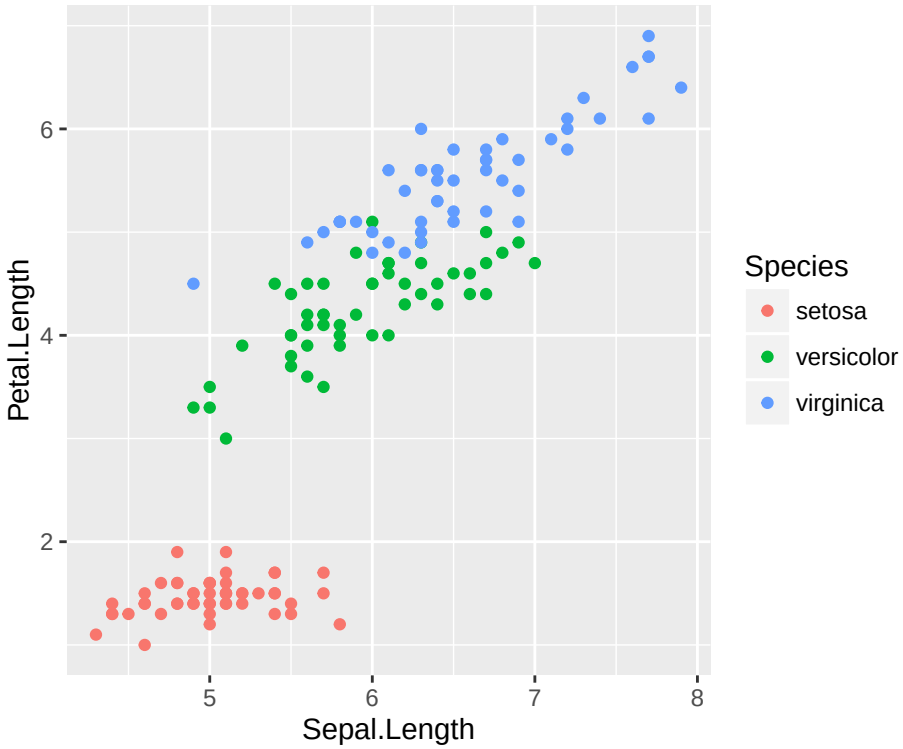
Se crea una gráfica: scatterplot of *Sepal.Length* with *Petal.Length*.

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point()
```



Adicionando color al gráfico

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(aes(color = Species))
```



Realizado en la Universidad Distrital Francisco José de Caldas, Maestría en Ciencias de la Información y las Comunicaciones.

Se compilaron en total 18 chunks.