Tarea 1 - ACP-149 - Estadística

Daniela Citlalli Tuz Lopez

2/10/2021

El lenguaje R, vectores

El lenguaje R, como se vió en sesiones anteriores tiene como un tipo de datos importante a los vectores. Los vectores atómicos se pueden generar de diversas formas, por ejemplo usando el comando `vector()` el cual toma dos parámetros, el tipo de vector atómico (`character`, `double`, etc) y el número de valores de ese tipo. Por ejemplo:

```
# Con el comando vector generamos vectores de la siguiente manera
vector("numeric", 5) # vector(tipo_elemento, no_de_elementos)
```

```
## [1] 0 0 0 0 0
```

Además de `numeric` también podemos generar vectores del tipo `character`, `double`, `integer` y `logical`.

Actividad

- Genere un vector de 10 valores tipo `integer`.

```
vector<-sample(1:100, 10, replace = TRUE)
vector
```

```
##  [1]  98  97   8  42 100  63  74  65  37  38
```

- Genere un vector de 15 valores tipo `character`

```
vector<-rep("character", 15)
vector
```

```
##  [1] "character" "character" "character" "character" "character" "char
acter"
##  [7] "character" "character" "character" "character" "character" "char
acter"
## [13] "character" "character" "character"
```

Generación de vectores atómicos

Note que el comando `vector`, solo genera vectores con valores por defecto, por ejemplo para los del tipo `numeric` genera puros 0s. Para generar vectores más interesantes, éstow los podemos generar mediante el comando `c()` que es una abreviación de `combine` o combinar. Por ejemplo, para generar un vector con los nombres de 5 personas que viven en Cancún, lo podemos hacer de la siguiente forma:

```r
# Código para generar un vector con los nombres de 5 personas
personas <- c("Eduardo", "Maria", "Mercedes", "Iván", "Eugenio")
print(personas) # Con este comando imprimimos el vector personas.
```

```
## [1] "Eduardo"  "Maria"    "Mercedes" "Iván"     "Eugenio"
```

Actividad

Genere los siguientes vectores

1. Un vector con diez números enteros e imprímalos

```r
enteros <- c(42,18 ,67 ,8 ,1 ,5 ,89 ,99 ,45 ,10)
print(enteros)
```

```
##  [1] 42 18 67  8  1  5 89 99 45 10
```

2. Un vector con 5 valores lógicos (TRUE o FALSE)

```r
logicos<- sample(c(TRUE, FALSE), 5, replace = TRUE)
print(logicos)
```

```
## [1] FALSE  TRUE FALSE  TRUE  TRUE
```

3. Un vector con 10 nombres de compañeros de la clase.

```r
compClase <- c("Gibran","Leydi","Armando","Raquel","Paola","Karol", "jorg
e", "Gadiel", "Andres", "Gael")
print(compClase)
```

```
##  [1] "Gibran"  "Leydi"   "Armando" "Raquel"  "Paola"   "Karol"   "jorg
e"
##  [8] "Gadiel"  "Andres"  "Gael"
```

R como calculadora

R, como se vió en sesiones previas, permite realizar cálculos matemáticos tal cual fuese una calculadora. R permite realizar casí cualquier tipo de cálculo desde aritmética básica hasta el cálculo de la derivada de una función. Por ejemplo:

```r
# R como calculadora
sin(pi)   # Cálculo del seno de pi
```

```
## [1] 1.224606e-16
```

```r
sqrt(4)   # Cáculo de la raíz cuadrada de 4
```

```
## [1] 2
```

```r
min(c(4,9)) # Valor mínimo entre 4 y 9
```

```
## [1] 4
```

Actividades

Hallar las siguientes cantidades usando R

- $\cos(2\pi)$

```
calculadora<-cos(2*pi)
calculadora
```

```
## [1] 1
```

- $\sqrt{2\pi}/\max(3,2,4,1,8)$

```
calculadora<-sqrt(2 * pi) / max(c(3, 2, 4, 1, 8))
calculadora
```

```
## [1] 0.3133285
```

- $\log_{10}(10000)$

```
calculadora<-log(10000, base=10)
calculadora
```

```
## [1] 4
```

Generando secuencias en R

R permite generar secuencias de valores con el comando `seq`, el comando `seq` genera secuencias entre un valor mínimo y máximo y puede generar secuencias de acuerdo a cierto incremento (con el parámetro `by=`) o una longitud de valores entre mínimo y máximo (con el parámetro `length=`). Por ejemplo:

```
# Uso del comando seq()
seq(1,6)  # Genera valores entre 1 y 6 en intervalos de 1 (por defecto)
```

```
## [1] 1 2 3 4 5 6
```

```
seq(1,2, by=0.1) # Genera valores entre 1 y 2 en incrementos de 0.1
```

```
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

```
seq(1,20, length=30) # Genera 30 valores entre 1 y 20
```

```
##  [1]  1.000000  1.655172  2.310345  2.965517  3.620690  4.275862  4.93
1034
##  [8]  5.586207  6.241379  6.896552  7.551724  8.206897  8.862069  9.51
7241
## [15] 10.172414 10.827586 11.482759 12.137931 12.793103 13.448276 14.10
3448
## [22] 14.758621 15.413793 16.068966 16.724138 17.379310 18.034483 18.68
9655
## [29] 19.344828 20.000000
```

Actividades

Generar las siguientes secuencias:

1. Una secuencia entre 0 y π con 10 valores.

```
ans<- seq(0,pi, length=10)
ans
```

```
##  [1] 0.0000000 0.3490659 0.6981317 1.0471976 1.3962634 1.7453293 2.094
3951
##  [8] 2.4434610 2.7925268 3.1415927
```

2. Una secuencia de 10 a 8, con incrementos de 0.33

```
ans<- seq(10, 8, by = -0.33)
ans
```

```
## [1] 10.00  9.67  9.34  9.01  8.68  8.35  8.02
```

3. Una secuencia con valores entre 0 y $\log_2(8)$ con 20 valores.

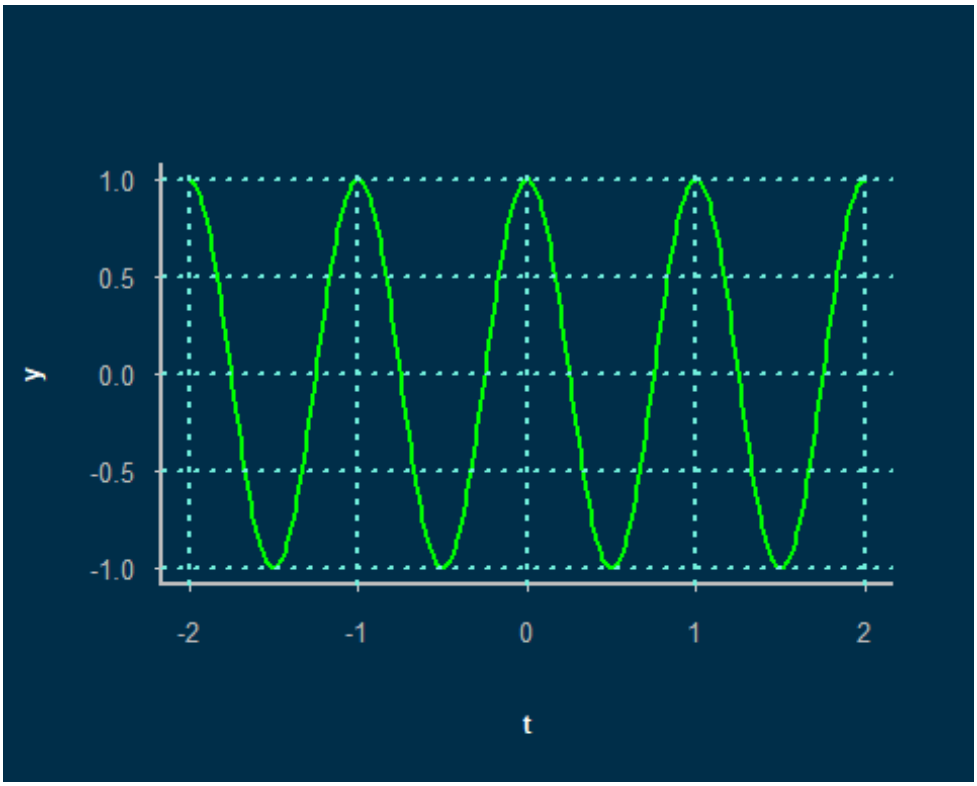
```
# Inserten su código abajo
ans<- seq(0, log2(8), length = 20)
ans
```

```
##  [1] 0.0000000 0.1578947 0.3157895 0.4736842 0.6315789 0.7894737 0.947
3684
##  [8] 1.1052632 1.2631579 1.4210526 1.5789474 1.7368421 1.8947368 2.052
6316
## [15] 2.2105263 2.3684211 2.5263158 2.6842105 2.8421053 3.0000000
```

Graficando funciones con R

Una característica muy potente de Res en la realización de gráficos de funciones. Para lo anterior consideraremos que una función puede graficarse a partir de su relación $y = f(t)$ o $y = f(x)$ y los valores de x los podemos generar mediante `c()` o `seq()` y los valores de y aplicando una función a la variable x. Por ejemplo, graficaremos $y = \cos(2\pi t)$ de -2 a 2.

```
library(basetheme)  # para utilizar basetheme, primero instalarlo
basetheme("deepblue") # install.packages("basetheme")
t <- seq(-2,2, length=200)  # Generamos los valores del eje horiz.
y <- cos(2*pi*t)           # Calculamos la función.
plot(t,y, type="l", col="green")
grid(col="#72efdd")      # Le agregamos lineas punteadas con color.
```

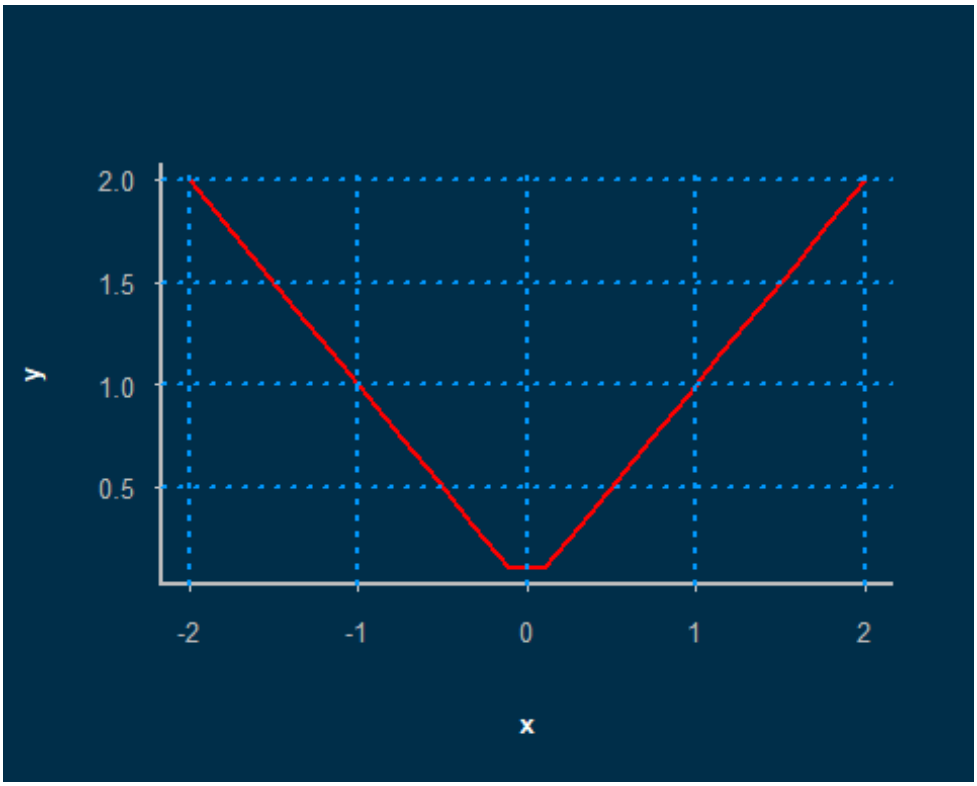


Actividades

Graficar las siguientes funciones:

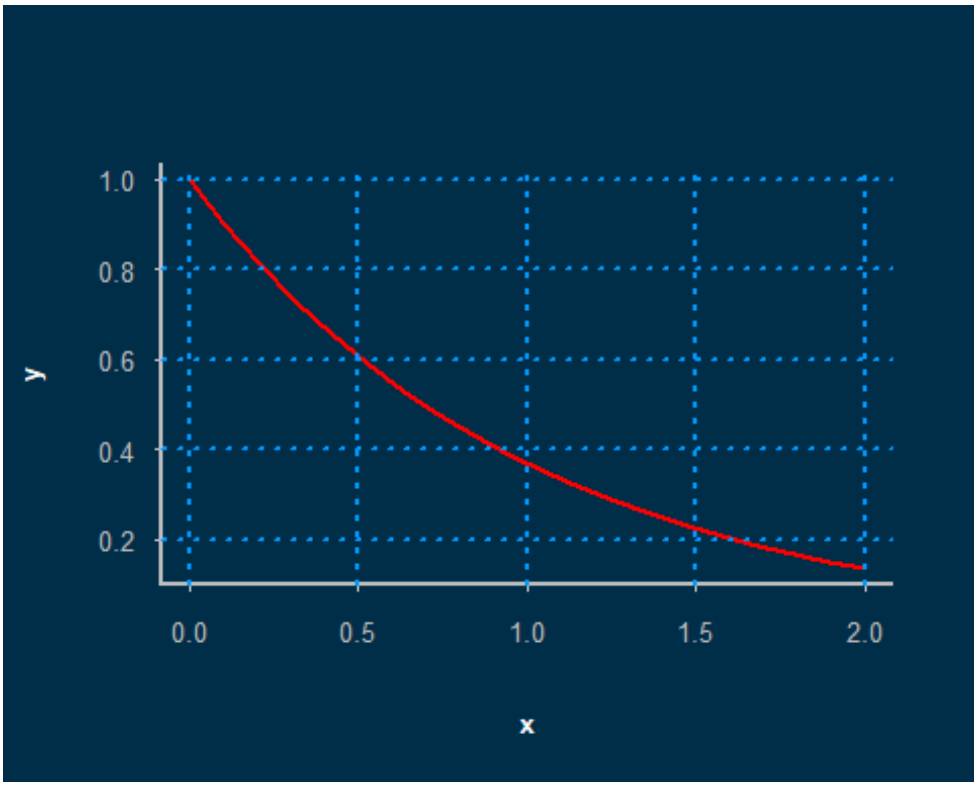
1. $y = |x|$ (valor absoluto de x) de -2 a 2.

```
library(basetheme)  # para utilizar basetheme, primero instalarlo
basetheme("deepblue") # install.packages("basetheme")
x <- seq(-2, 2, length = 20)
y <- abs(x)
plot(x,y, type="l", col="red")
grid(col="#0099ff")
```



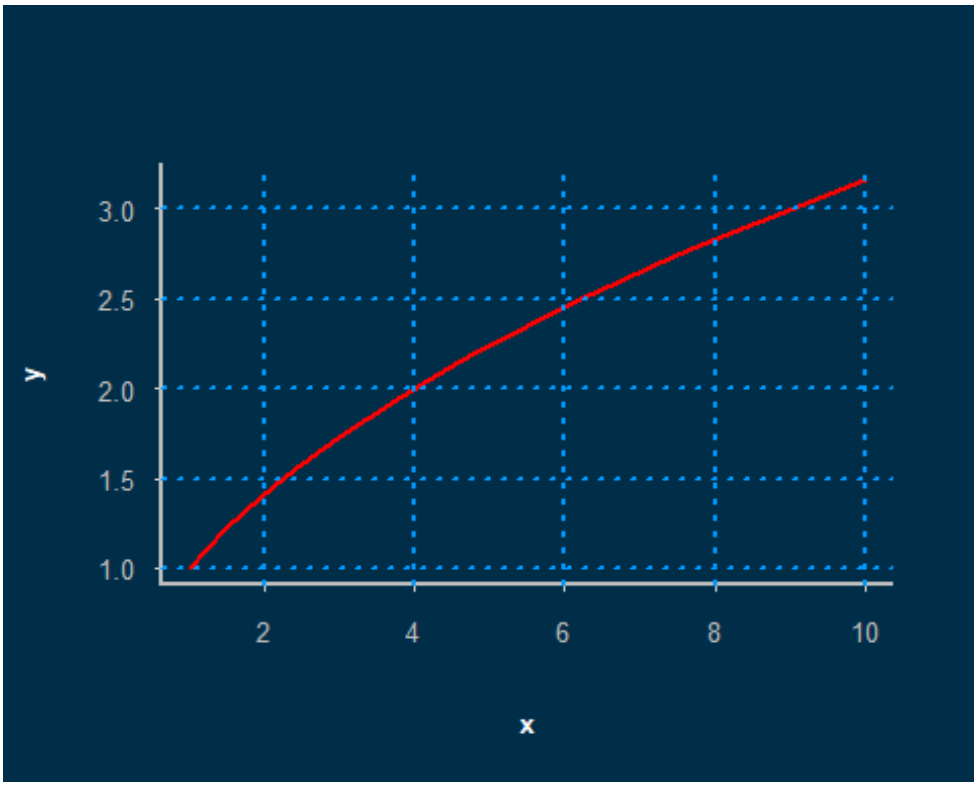
2. $y = e^{-x}$ de 0 a 2 (función exponencial)

```
library(basetheme)  # para utilizar basetheme, primero instalarlo
basetheme("deepblue") # install.packages("basetheme")
x <- seq(0, 2, length = 20)
y <- exp(-x)
plot(x,y, type="l", col="red")
grid(col="#0099ff")
```



3. Graficar la función $y = \sqrt{x}$ de 1 a 10

```
library(basetheme)  # para utilizar basetheme, primero instalarlo
basetheme("deepblue") # install.packages("basetheme")
x <- seq(1, 10, length = 20)
y <- sqrt(x)
plot(x,y, type="l", col="red")
grid(col="#0099ff")
```


3.0
2.5
2.0
1.5
1.0
y
2
4
6
8
10
x