Progrmación en R-Introducción

Santiago Enrique Lozano González

Universidad Pilot de Colombia Seccional Alto Magdalena

11 de enero de 2020

La Máquina Universal

- ¿Qué es exactamente una computadora?¿Cómo puede un dispositivo realizar gran cantidad de diferentes tareas?
- Un computador moderno puede definirse como “una máquina que almacena y manipula información bajo el control de programas cambiables”

La Máquina Universal

- Esta definición puede ser vista desde dos elementos claves.
- Las computadoras son dispositivos para manipular información (poner información y transformarla en nueva)

La Máquina Universal

- Las computadoras operan bajo el control de un programa cambiable
- Programa Computacional: Es un detallado, paso a paso conjunto de intrucciones que dicen a un computador exactamente que hacer, si se cambia el programa, se cambia el conjunto de acciones y se realiza una tarea diferente, la máquina es la misma pero el programa controla los cambios en la máquina

La Máquina Universal

Todo computador en sí es una máquina para ejecutar y realizar programas

Potencia de los Programas

- El software controla el Hardware, pues determina lo que cualquier computador puede hacer.

-El proceso de crear software se denomina programación, es una habilidad que requiere destreza de manera integral y desarrolar aptitudes que no son para todo el mundo, sin embargo, se puede aprender en como programar computadores

¿Qué es la ciencia de la computación (Computer Science)?

- Las ciencias de la Computación no son el estudio de los computadores
- El computador es a la Ciencia de la Computación lo que el Telescopio es a la Astronomía
- El computador es una herramienta importante en la ciencia de la computación, pues es es el que lleva a cab los procesos que podemos describir, pero el no es el objjeto de estudio

¿Qué es la ciencia de la computación (Computer Science)?

- La pregunta fundamental es ¿Qué proceso podemos describir? o mejor ¿Qué puede ser calculado?, existen numerosas técnica para resolver esta pregunta
- las 3 principales técnicas son el diseño, el análisis y la experimentación

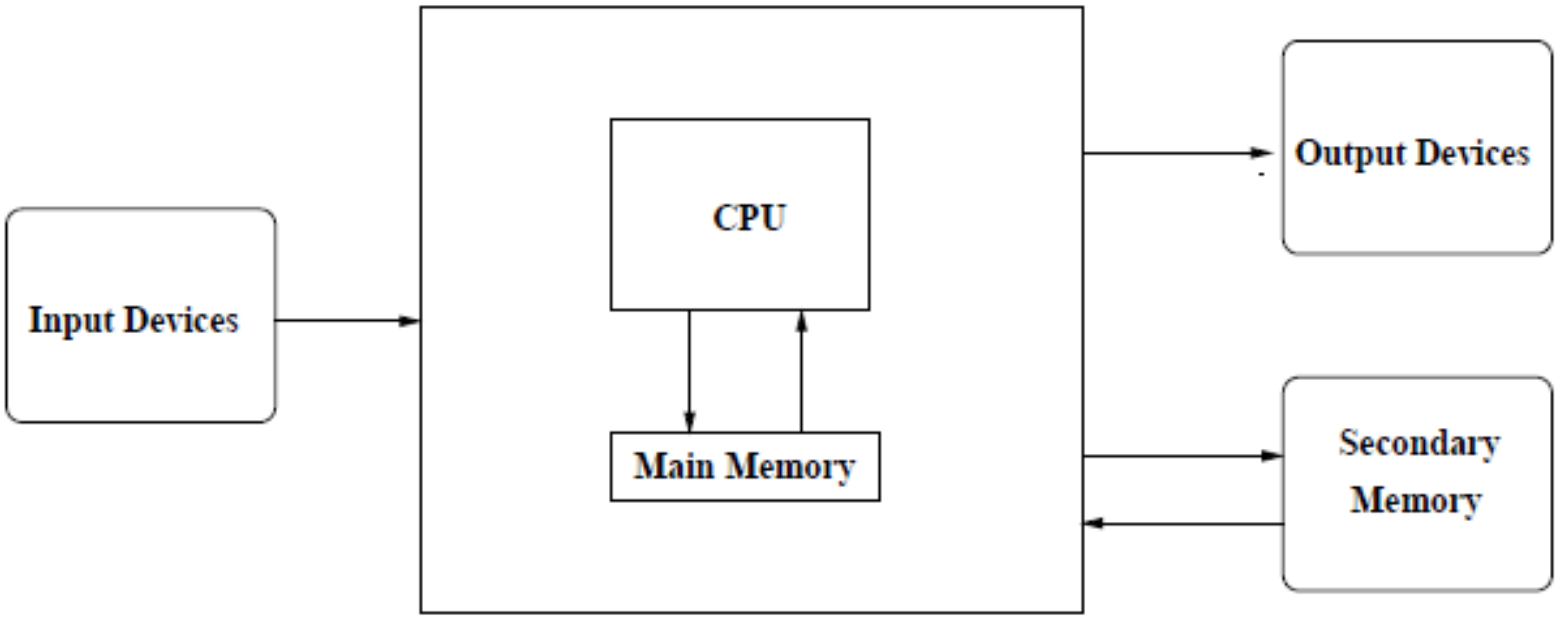
¿Qué es la ciencia de la computación (Computer Science)?

- Esto desde una perspectiva académica, pero hoy en día las ciencias de la computación envuelven mucho aspectos como redes, interación humano.computador, inteligencia artificial, bases de datos, ingeniería de software, etc

Cosas Básicas sobre el Hardware

- Aprender detalles sobre como funciona un computador nos harán mejores programadores, no es algo necesario, pero, ayuda a dominar el paso que seguimos para poner nuestros programas en acción, es como un auto, saber un poco sobre la parte mecánica ayuda a mejorar el rendimiento de este a la hora de conducir, aunque se puede aprender sin este paso
- Primeramente mencionamos la CPU (Central processing unit) la cual es el "cerebro de la máquina", es donde todas las operaciones básicas del computador se llevan a cabo.

Cosas Básicas sobre el Hardware



Cosas Básicas sobre el Hardware

- La memoria almacena programas y datos. La CPU puede solo acceder directamente a información que está almacenada en la memoria principal (Memoria RAM (Random Access Memory)), esta memoria es rápida pero es también volátil, es decir, si el computador se llega a apagar la información en la memoria se pierde
- Con esto debe existir una memoria secundaria que dé almacenamiento permanente, en un computador moderno, es usual encontrar algún tipo de medios magnéticos como disco duro (hard drive), medios óptico como CD (Compact disc) y DVD (Digital Versatile Disc) y memorias flash como memoria "USB" (stick memory)

Cosas Básicas sobre el Hardware

- los humanos interactuan con el computador a través de dispositivos de entrada y salida, dispositivos como el mouse, teclado son dispositivos de entrada.
- Tecnicamente la CPU sigue un proceso llamado ciclo Búsqueda-ejecución (fetch-execute cycle). La primera instrucción es recuperada de la memoria, descifrada para descubrir lo que representa y la acción apropiada es llevada a cabo. Entonces la siguiente instrucción es traída, decodificada y ejecutada. El ciclo continua instrucción tras instrucción. Esto es realmente lo que hace un computador desde que se enciende hasta que se apaga, para algo trivial pero la velocidad y la cantidad de millones de instrucciones que mueve la memoria cada segundo es impresionanante

Lenguajes de Programación

- Recordando que un programa es solo una secuencia de instrucciones que le dicen a un computador que hacer. Obviamente, necesitamos proveer esas instrucciones en un lenguaje que el computador puede entender. Sería genial si se pudiera indicarle a un computador usando el lenguaje nativo pero a pesar de los esfuerzo, diseñar un computador que pueda entender el lenguaje humano es una tarea sin solución, pues el lenguaje natural es tenso con ambiguedades e imprecisiones
- Las científicos en computación han conseguido alrededor de este problema diseñar notaciones para experesar cálculos en una vía exacta y sin ambiguedades. Estas notaciones especiales son llamadas lenguajes de programación

Lenguajes de Programación

- Toda estructura en un leguaje de programación tiene una forma precisa (sintaxis) y un significado preciso (semántica)
- Los programadores se refieren a menudo a sus programam como un código de computadora
- y el proceso de escribir un algoritmo en un leguaje de progrmación se denomina codificar

Lenguajes de Programación

- R es un ejemplo de un lenguaje de programación y el que veremos durante todo el curso
- Existen otro tipo de lenguaje como Python, Java, C++, Fortran, Basic. Los cuales poseen diferente características pero comparten la característica de estar bien definidos, sintaxis y semántica no ambiguas
- los lenguajes mencionados anteriormente son ejemplos de lenguajes de alto nivel, los cuales, además de ser precisos, son diseñados para ser usados y entendido por humanos

Lenguajes de Programación

- El hardware computacional puede solo entender un muy lenguaje de bajo de nivel conocido como lenguaje de máquina
- Suponga que queremos que el computador sume dos números. La instrucción que la CPU lleva a cabo realmente puede ser algo como

Lenguajes de Programación

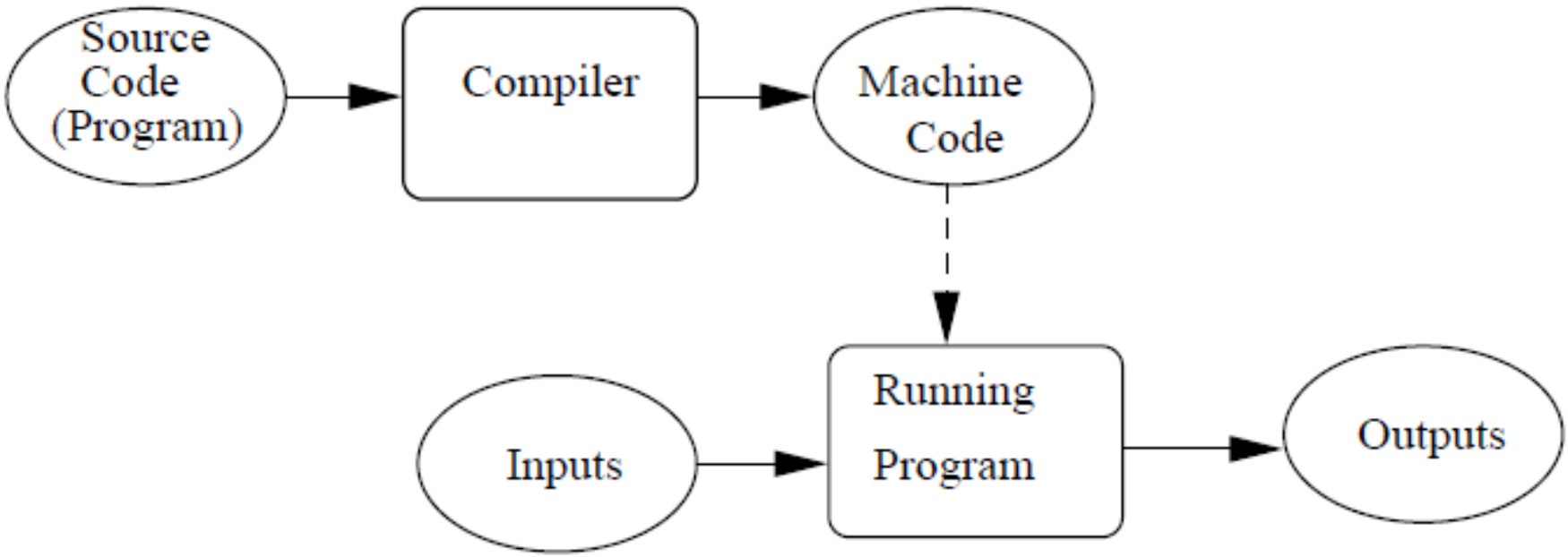
- cargue el número desde la ubicación 2001 en memoria a la CPU
- cargue el número desde la ubicación 2002 en memoria a la CPU
- sume los dos números en la CPU
- Almacene los dos números en la ubicación 2003

Parece ser mucho trbajo sumar dos número, es más, puede ser más complicado sabiendo que la instrucción y los números son representados en notación binaria

Lenguajes de Programación

- En un lenguaje de alto nivel la suma de dos números se puede hacer de una simple manera c = a + b que es más fácil de entender, pero necesitamos transladar esto a un lenguaje de máquina que el computador pueda ejecutar, para esto disponemos de dos maneras compilar o interpretar
- Un compilador es un programa de computadora complejo que toma otro programa escrito en alto nivel y lo traduce a un equivalente programa en lenguaje de máquina de algún computador

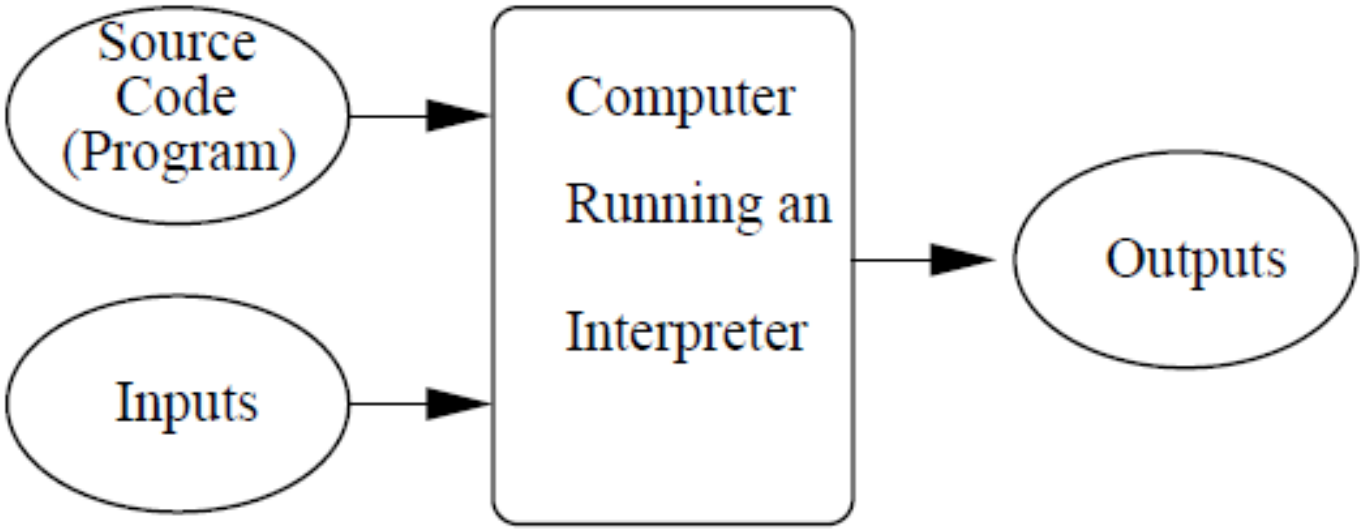
Lenguajes de Programación



Lenguajes de Programación

- Un interpetador es un programa que simula una computadora que entiende un leguaje de alto nivel. En vez de traducir, el programa fuente a lenguaje de máquina equivalente, el interpretador analiza y ejecuta la instrucción de código fuente instrucción por instrucción cuante veces sea necesario
- La diferencia entre interpretador y compilador es que un compilador es una traducción de un tiro, una vez el programa es compilado, este puede ser corrido una y otra vez sin más trabajo del compilador o el código fuente. En el caso de un interpretador, la interpretación y el código fuente son necesarias cada vez que se corre el programa

Lenguajes de Programación



Lenguajes de Programación

- los programas compilados tienden a ser más rápidos, pero los lenguajes interpretados tienden a si mismos a tener un ambiente de programamción más flexible, como programa que pueden ser desarrollados y corridos interactivamente
- otra ventaja en los programas de alto nivel es su portabilidad y es que estos puede correr en cualquier CPU, contrario a los lenguajes de máquina los cuales pueden ser corrido para CPU de sus respectivos fabricantes

Introducción a R (Definición)

- De esta manera, R es un leguaje de alto nivel y un ambiente de trabajo para el análisis de datos y gráficos, este posee un intérprete el cual ejecuta el código que le demos, a diferencia de lenguajes como Java, C o C++ los cuales deben compilar primero su código, aunque en algunos Códigos exigentes son escritos en C o C++.

Introducción a R (Origen)

- R es un software de libre uso y distribución bajo Licencia Pública General de GNU, para programar análisis estadístico y gráfico. R fue creado en 1993 por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland-Nueva Zelanda y desde 1997 se desarrolla con aportes de diversas partes del mundo, bajo la coordinación del equipo principal de desarrollo de R (R Core Team Development) (R Project).
- Es un programa clonado del program S-Plus

¿Por qué usar R?

- Primero que todo R es Gratis y es de código abierto, es decir, es un proyecto colaborativo y abierto donde los usuarios pueden definir sus propias funciones, aparte de numerosas biblioteca preconstruidas que tiene, también se pueden publicar paquetes que puede extender su configuración básica
- Es posible que algoritmos computacionalmente exigentes pueden ser desarrollados en C, C++ o Fortran y vincularlos directamente con R
- Al ser un lenguaje que contiene interpretador posee variedad de ambientes de trabajo como R Studio, R Commander, Tinn R y el ambiente predeterminado que permiten una amigabilidad a la hora de manejar códigos y datos

¿Por qué usar R?

- R funciona con paquetes de programación, los cuales están disponibles en una Red Comprehensiva de Archivos R (Comprehensive R Archive Network, CRAN) en sitios web llamados MIRROR- sitios que contienen réplicas exactas de R- desde los cuales los usuarios finales pueden descargarlos. Actualmente están disponibles 92 CRAN-MIRROR, en 45 países de los cinco continentes, 14 de las cuales se encuentran en instituciones - principalmente universidades- de siete países de América Latina: Argentina, Brasil, Chile, Colombia, Ecuador, México y Venezuela

Propiedades de R (Funcional)

- Las funciones en R se pueden manipular igual que los vectores. Además puedes asignar las funciones a variables, almacenarlas en listas, devolverlas como resultados de otras funciones o incluso pasarlas como argumentos de otras funciones
- Ofrece múltiples posibilidades para atacar a datos almacenados en distintos tipos de bases de datos. También presenta múltiples bindings y paquetes que permiten a R interactuar con otros lenguajes (como Perl, Ruby o Python) e intercambiar objetos con ellos.
- Existen librerías para R que permiten generar una extensa variedad de gráficos, desde la completísima ggplot2 hasta otras más simples pero también potentes como corrplot

¿Por qué usar R?

- R mantiene todos los objetos que definimos en nuestro programa en la memoria de nuestra máquina. Por ello, es importante entender cómo gestiona la memoria, para poder optimizar nuestro código. Así evitamos, por ejemplo, copias innecesarias de objetos que pueden ralentizarlo y hacer llegar a un límite nuestra máquina.
- Este lenguaje de programación fue concebido para el análisis estadístico, aunque también se utiliza en la minería y análisis de datos, investigación biomédica, bioinformática, machine learning... Esto es porque proporciona un amplio abanico de herramientas estadísticas y gráficas, además de tener una gran potencia como herramienta de cálculo.

Instalación

- http://cran.r-project.org/
- R Studio

Iniciar R

```
#R version 3.5.1 (2018-07-02) -- "Feather Spray"
#Copyright (C) 2018 The R Foundation for Statistical   Comp
#Platform: x86_64-w64-mingw32/x64 (64-bit)

#R is free software and comes with ABSOLUTELY NO WARRANTY.
#You are welcome to redistribute it under certain #conditi
#Type 'license()' or 'licence()' for distribution details.

#R is a collaborative project with many contributors.
#Type 'contributors()' for more information and
#'citation()' on how to cite R or R packages in publicatio

#Type 'demo()' for some demos, 'help()' for on-line help,
#'help.start()' for an HTML browser interface to help.
#Type 'q()' to quit R.
```

Iniciar R

Debajo de este encabezado veremos una línea impresa con $>$ de símbolo en la parte izquierda del margen. Este es llamado el prompt y es el lugaro donde escribiremos nuestros comando. Cuando trabajama algunas veces veremos un símbolo como $+$ el la parte izpuierda del margen, lo que indicará que el comando escrito no está completo.

Podremos ejecutar los comando oprimiendo Enter y si cometes un error oprimiendo Esc se logrará retornar al prompt

The Comprehensive R Archive Network (CRAN)

Es nuestro primer sitio par acudir y llamar cualquier cosa que queramos hacer en R, es desde aquí que podemos descargar, instalar R, encontrar ppaquetes para resolver cierto problemas y encontrar las respuestas y hacer preguntas acerca de los últimos desarrollos

- http://cran.r-project.org/

Obtener ayuda en R

Si tienes el nombre específico de la función

```
?read.table
```

Si no tienes el nombre de la función pero sabes el tema con el que tratas

```
help.search("data input")
```

Obtener ayuda en R

para encontrar el paquete correspondiente a una función

```
find("lowess")
```

```
## [1] "package:stats"
```

para encorntrar ciertos elementos asociados a una función o palabra que estemos buscando

```
apropos("lm")
```

Obtener ayuda en R

Para ver ejemplos asociados a las funciones

```
example(lm)
```

demostraciones

```
demo(persp)
demo(graphics)
demo(Hershey)
demo(plotmath)
```

Paquetes

Encontrar paquetes que han sido contibuídos en R puede ser una tarea ardua, además que no siempre el nombre del paquete es una vía para encontrar cierta función; para esto R tiene asignados los paquetes por ciertas categorías que pueden ser visualizadas en el CRAN

- http://cran.r-project.org/ -> Task Views

Para poder usar uno de los paquetes previamente descargardos usamos la función library

```
library(spatial)
```

Paquetes

Usando la ayuda de R podemos descubrir los contenidos de librarías de los paquetes así

```
library(help=spatial)
```

Puedes ver una lista de los contenidos de una librería usando objects con search()

Paquetes

```
objects(grep("spatial",search()))
```

```
##  [1] "anova.trls"      "anovalist.trls" "correlogram"
##  [5] "gaucov"          "Kaver"          "Kenvl"
##  [9] "plot.trls"       "ppgetregion"    "ppinit"
## [13] "ppregion"        "predict.trls"   "prmat"
## [17] "semat"           "sphercov"       "SSI"
## [21] "surf.gls"        "surf.ls"        "trls.influence"
## [25] "variogram"
```

Paquetes

Para instalar paquetes usamos

```
install.packages("tree")
```

Línea de Comando versus scrpits

Cuando escribimos funciones y otras secciones multi-línea de salida, algunas personas le encuentran más utilidad en usar un editor de texto en vez de ejecutar directamente todo directamente en la línea de comando. Muchas personas prefieren usar el ambiente original de R, R Studio, Tinn-R, etc.

Para correr ciertas líneas desde un script, se seleccionana y se oprime Ctrl+R y las líneas son automáticamente tranferidas a la línea de comando.

Para guardar usamos Ctrl+S se guerdan la ventana editor en un archivo con extensión .R

Buena limpieza

Para ver que variables ha creado en su actual sesión usamos

```
objects()
```

para ver que paquetes y dataframes se ha usado

```
search()
```

```
##  [1] ".GlobalEnv" "package:spatial" "package:stats"
##  [4] "package:graphics" "package:grDevices" "package:uti
##  [7] "package:datasets" "package:methods" "Autoloads"
## [10] "package:base"
```

Buena limpieza

para eliminar variables usamos

```
rm()
```

El comando detach no desaparece una variable, solo hace que las variables del dataframe no estén accesibles de manera directa

```
detach()
```

Buena limpieza

para deshacerse de todo incluyendo dataframes usamos

```
rm(list=ls())
```

Apartes del Curso

- 1er Corte:
  - parcial (14 marzo): 40%
  - Taller: 15%
  - Quiz: 25%
  - Lectura (21 marzo): 10%
  - Pre-anteproyecto (14 marzo): 10%
- 2do Corte:
  - parcial (25 abril): 40%
  - Taller: 20%
  - Quiz: 30%
  - Propuesta proyecto final (25 abril): 10 %
- 3er Corte:
  - Quiz: 40%
  - Proyecto Final (29 y 30 mayo): 60%