Trabajo Fin de Grado

TITULO DEL TFG

Visualización de Datos de Rendimiento en Sistemas de Cómputo de Altas Prestaciones

ULL - ESIT
AUTOR

Jose Luis González Hernández

Tutor
Vicente José Blanco Pérez

La Laguna, 5 de septiembre de 2017

D. **Vicente José Blanco Pérez**, con N.I.F. 42.171.808-C profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada: *Visualización de Datos de Rendimiento en Sistemas de Cómputo de Altas Prestaciones.*

ha sido realizada bajo su dirección por D. **José Luis González Hernández**, con N.I.F. 78567818-W.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de septiembre de 2017.

Agradecimientos

Quiero agradecer a mis padres todo el apoyo que me han dado y el ánimo para poder seguir adelante.

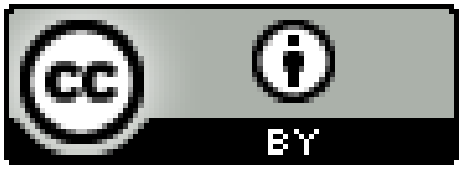
A mi tutor Vicente José Blanco Pérez por su paciencia, dedicación y enseñanza a lo largo de mi carrera y del desarrollo del Trabajo Fin de Grado.

A todos los profesores de la ETSII, con los que he aprendido y he podido recorrer este camino.

A todos mis amigos y compañeros por sus ánimos y apoyo.

Y por último, en especial a mi novia y mi futura esposa Rebeca, por todo su cariño, paciencia y apoyo incondicional.

Licencia



Resumen

Las arquitecturas de procesadores y aceleradores de cómputo que se encuentran actualmente en los sistemas de cómputo de altas prestaciones son grandes consumidoras de energía. Hay soluciones hardware y software para minimizar este consumo, como bajar la frecuencia de reloj o el voltaje en estos dispositivos. Estudiaremos soluciones software que permitan monitorizar el rendimiento y la eficiencia energética de estos sistemas cuando se ejecutan aplicaciones en ellos. No obstante, la gran capacidad computacional conlleva una mayor dificultad para desarrollar aplicaciones, las cuales aprovechan eficientemente los recursos computacionales, para ello se necesita tener mecanismos adecuados para poder analizar, comprender y predecir el comportamiento de las aplicaciones en los sistemas paralelos. Con lo cual, las herramientas de análisis de rendimiento son un utensilio primordial para examinar y explotar el potencial de dichos sistemas. El que exista un desarrollo continuado de nuevas arquitecturas produce un condicionamiento de la investigación asociada al análisis del rendimiento en el ámbito de la computación de altas prestaciones. En cualquier caso, el desarrollo de una metodología universal de análisis del rendimiento es un objetivo prácticamente inalcanzable debido al amplio, y cada vez mayor, espectro de arquitecturas y configuraciones. Por ello desarrollaremos una aplicación para la visualización de los datos de rendimiento obtenidos con el sistema de instrumentación CALL desarrollado en la ULL y poder realizar una primera aproximación al análisis y modelado de rendimiento de aplicaciones.

Palabras clave: Análisis de Rendimiento, Modelado del rendimiento, HPC, CALL, R, RStudio, Shiny, Nginx, Scrum .

Abstract

Actual processors and computing accelerators architectures in High Performance Computing systems usually demands high levels of energy. There are hardware and software solutions to minimize this power consumption, typically pulling down clock frequency or voltage in these devices. We will study software solutions to monitor the performance and energy efficiency of these systems when the applications are running on them. Nevertheless, the great computational capacity entails more difficult to develop applications which leverage computing resources efficiently, so you need to have appropriate mechanisms to analyze, understand and predict the behaviour of applications on parallel systems. Therefore, performance analysis tools are needed to examine and exploit the potential of such systems. There is a continued development of new architectures that produces a conditioning of the research associated with the analysis of the performance in the field of high performance computing. In any case, the development of a universal methodology of performance analysis is a virtually unattainable object due to the large, and more and more, range of architectures and configurations. For this reason, we will develop an application for the visualization of performance data achieved with the instrumentation system CALL developed at ULL. With these tools we can achieve a first overview of codes' performance.

Keywords: *Performance Analysis, Performance Modelling, HPC, CALL, R, RStudio, Shiny, Nginx, Scrum .*

Índice general

Índice de figuras

Listings

Capítulo 1

Introducción

El desarrollo de nuevas arquitecturas en sistemas paralelos ha sido siempre un área de investigación muy activa, esto se debe a la continua demanda que genera la comunidad científica de un gran poder computacional. En las últimas décadas, esta progresión se ha visto acelerada por las nuevas tendencias en el desarrollo de microprocesadores, lo que ha llevado a que los sistemas que existen actualmente dispongan de un número cada vez mayor de núcleos computacionales, los cuales están dispuestos en una estructura intrínsecamente jerárquica. No obstante, la gran capacidad computacional conlleva una mayor dificultad para desarrollar aplicaciones, las cuales aprovechan eficientemente los recursos computacionales, para ello se necesita tener mecanismos adecuados para poder analizar, comprender y predecir el comportamiento de las aplicaciones en los sistemas paralelos. Con lo cual, las herramientas de análisis de rendimiento son un utensilio primordial para examinar y explotar el potencial de dichos sistemas. Los resultados que se obtienen por dichas herramientas nos suministran un enfoque empírico, de las posibilidades de mejora, y así nos muestra posibles direcciones para el desarrollo de futuros sistemas.

En la actualidad, los sistemas de cómputo de altas prestaciones ofrecen un gran rendimiento, pero consumen grandes cantidades de energía. Hoy en día hay diversos estudios que intenta resolver parte del problema mediante cambios dinámicos de la frecuencia de reloj de las CPU. Las frecuencias más bajas requieren menos potencia, que lleva a una reducción del calor generado, e indirectamente a un aumento del tiempo medio entre fallos de los componentes del sistema de cómputo, menos energía necesaria para la refrigeración, y la posibilidad de aumentar la densidad de componentes. Muchos algoritmos intentan reducir el consumo sin reducir el rendimiento, pero muchas veces es mejor llevar el consumo energético a niveles inferiores aunque esto provoque pérdida de rendimiento.

Los sistemas de cómputo normalmente logran menores potencias y mayor eficiencia energética al ejecutar las aplicaciones a la menor frecuencia de reloj de las CPU.

El que exista un desarrollo continuado de nuevas arquitecturas produce un condicionamiento de la investigación asociada al análisis del rendimiento en el ámbito de la computación de altas prestaciones. Las técnicas, los métodos y las herramientas tienen que estar capacitadas para adaptarse a las nuevas exigencias que han sido impuestas por la evolución de los sistemas paralelos.

En cualquier caso, el desarrollo de una metodología universal de análisis del rendimiento es un objetivo prácticamente inalcanzable debido al amplio, y cada vez mayor, espectro de arquitecturas y configuraciones. Por lo que para poder abordar el análisis del rendimiento de sistemas paralelos, han nacido distintas aproximaciones las cuales se centran en aspectos determinados del análisis del rendimiento.

1.1. Objetivos

Nuestro objetivo es ayudar a encontrar un **modelo de rendimiento** para un código o aplicación HPC mediante un ajuste por mínimos cuadrados [14] de los datos de rendimiento obtenidos mediante el sistema de instrumentación CALL [25, 23, 24]. Para ello, implementaremos una interfaz web que interactué con los datos de instrumentación que genera CALL.

Para ello, en primer lugar, trataremos de documentarnos sobre el entorno Rstudio[7] y Shiny [9] para ver si nos aporta las herramientas necesarias para realizar nuestra aplicación. Estudiaremos el framework TIA [22], el cual se basa en un entorno de predicción del rendimiento de aplicaciones basado en la integración de modelos analíticos. Este sistema está compuesto por un traductor código a código (CALL), un conjunto de funciones de instrumentación y una librería de análisis estadístico (LLAC). Dicha librería nos brinda una interfaz fácil de usar, y la cual esta diseñada especialmente para manejar los datos que son generados por CALL y poder así llevar a cabo los análisis. También cabe destacar que esta librería consta de una colección de funciones y estructuras las cuales están desarrolladas dentro del entorno estadístico R [12, 13]. Esta librería nos aportará el poder importar a un determinado formato los datos de rendimiento y así poder calcular los mínimos cuadrados, en función de los datos de rendimiento aportados por CALL. También nos permite añadir diferentes funciones que generan los gráficos específicos con los resultados.

Una vez documentado y verificado que podemos usar estos entornos, veremos como R nos aporta otras funciones que pueden ser útiles a la hora de la implementación de nuestro proyecto, una de ellas es la función plot(), que nos permite generar gráficas con los datos de rendimiento importados, otra función que nos aporta R, es lm(), que nos permite hacer ajustes por mínimos cuadrados con los datos importados. Nos hemos propuesto crear un control que nos permita especificar los rangos de la gráfica, esto nos serviría para especificar que datos queremos ver en nuestra gráfica haciendo un zoom, este control se podría hacer de forma manual, pidiéndole al usuario que introdujese los rangos x e y, y también se podría gestionar mediante slider.

En cuanto a la generación de los ajustes por mínimos cuadrados, nos hemos propuesto crear un apartado en el cual el usuario pueda introducir la expresión de la ecuación a ajustar por mínimos cuadrados y mediante la función lm(), generar dicho ajuste. También se podría crea un menú desplegable en el cual ya estuvieran dichas fórmulas de forma predeterminada, de las cuales el usuario podría escoger y generar dichos ajustes. Una vez generada la gráfica con los ajustes de mínimos cuadrados, nos planteamos crear una opción que permita al usuario guardar dicha gráfica, para su estudio posterior.

1.2. Antecedentes y estado actual del tema

Como ya comentamos anteriormente, hoy en día no hay sistemas que estudien el modelo y la visualización conjuntamente, pero si hay varias herramientas que nos permiten visualizar los datos, algunas de ellas son Vampir y Paraver las cuales emplean visualización por trazas, también nos encontramos con TAU, sin embargo, esta está basada en profiling.

1.2.1. Otras aplicaciones

- Vampir — Intel Trace Analyzer and Collector (ITAC).

  Vampir (Visualization and Analysis of MPI Resources) es una herramienta comercial ampliamente utilizada, desarrollada por el Center for Applied Mathematics of Research Center Jülich y el Center for High Performance Computing de la Technische Universität Dresden [17, 26]. Durante la ejecución de un programa, la herramienta VampirTrace genera una traza en el formato OTF que se visualiza interactivamente con Vampir. VampirTrace instrumenta automáticamente las funciones MPI mediante la técnica de interposición de librería, pero también permite añadir eventos definidos por el usuario utilizando una API para instrumentación de código fuente. Intel Trace Analyzer and Collector es un entorno para la instrumentación y el análisis de códigos paralelos MPI [16]. Este entorno está compuesto por una herramienta de instrumentación (Intel Trace Collector, ITC) y una herramienta de visualización interactiva (Intel Trace Analyzer, ITA). Ambas herramientas son el producto de la colaboración de Intel con los desarrolladores de la herramienta comercial Vampir [18].

- Paraver.

  Paraver es una herramienta flexible para el análisis y visualización del rendimiento de aplicaciones paralelas [21]. Esta herramienta forma parte del entorno CEPBA-Tools, desarrollado en el Barcelona Supercomputing Center Centro Nacional de Supercomputación [1]. Paraver utiliza un formato de traza flexible, que permite realizar diferentes tipos de análisis. Estas trazas pueden obtenerse a partir del código fuente de aplicaciones paralelas escritas en OpenMP o MPI, pero también pueden obtenerse a partir del simulador Dimemas [21]. Además de la representación visual personalizable de los eventos registrados, Paraver ofrece otras posibilidades como, por ejemplo, realizar un análisis cuantitativo de las diferentes métricas consideradas en la traza, la generación de nuevas métricas derivadas o el análisis concurrente de varias trazas. Paraver también proporciona una salida en texto con información muy detallada.

- TAU - Tuning and Analysis Utilities

  Es un sistema de análisis de rendimiento paralelo, es el producto de 14 años de desarrollo para crear un framework robusto, flexible, portable e integrado y un conjunto de herramientas para instrumentación, medición, análisis y visualización de sistemas y aplicaciones paralelos de gran tamaño. TAU soporta la configuración e integración de estas tres capas para conseguir resolver problemas de rendimiento puntuales. Del mismo modo, la exploración efectiva de rendimiento necesita seleccionar un conjunto de

métodos alternativos. TAU [15] permite la combinación de experimentos de rendimiento significativos que facilitan la obtención de propiedades de rendimiento relevantes. Para lograr esto, TAU ofrece soporte para efectuar análisis de rendimiento de distintas maneras, incluyendo la capacidad de realizar un proceso de instrumentación multinivel bastante robusto, modalidades de medición empleando trazado y profiling, análisis de rendimiento interactivo y gestión de datos de rendimiento, los datos obtenidos a través del profiling pueden visualizarse en diferentes gráficos estadísticos.

1.3. Metodología de trabajo

Durante la investigación y desarrollo de este proyecto se ha apostado por las metodologías ágiles, si bien no ha sido en un grupo de trabajo, se puede aplicar en lo que se refiere al análisis, al desarrollo y diseño de la aplicación y a las reuniones con el *cliente*, cuyo rol lo asume el tutor junto con el de *jefe de proyecto*.

Estos métodos están basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos. La idea es minimizar riesgos desarrollando software en períodos cortos. Este período se llama **iteración** cuya duración se define en función de los requisitos del proyecto y los recursos empleados, especialmente el tiempo y el factor humano.

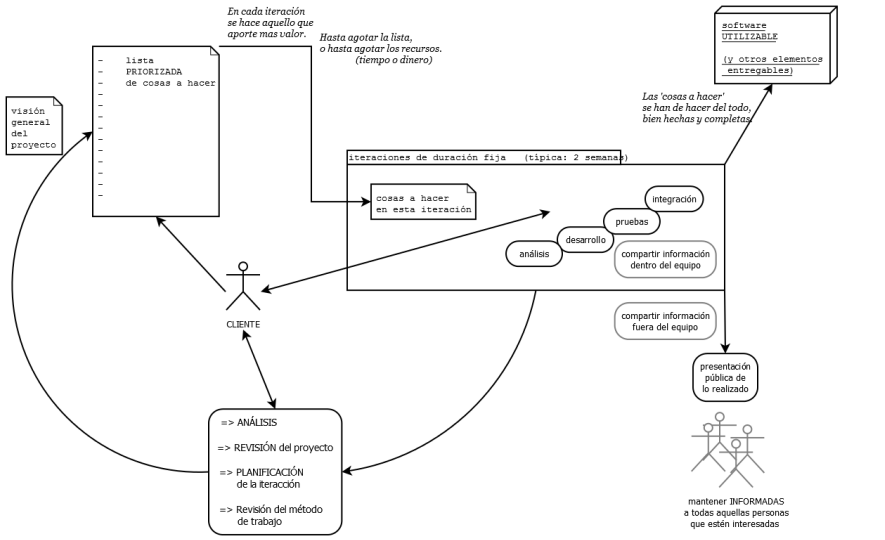


Figura 1.1: Esquema de metodología ágil

Cada iteración del ciclo de vida incluye:

- Planificación: organización del trabajo durante una iteración. En este caso, la planificación corresponde con las tutorías y seguimiento semanal con el tutor.

- Análisis de requisitos: cada nueva funcionalidad o idea para la aplicación debe ser estudiada.
- Diseño: interfaz de la aplicación, características y aspectos de usabilidad.
- Codificación: corresponde por lo general a escribir código, aunque esto no se limita solo a realizar la propia aplicación, además se incluyen todos aquellos prototipos durante la etapa de estudio de campo.
- Revisión: incluye superar los tests con éxito y la aceptación por parte del cliente/Tutor.
- Documentación: para este trabajo en concreto, se documentó cada paso, de instalación y configuración necesarias.

Durante este proyecto se ha seguido dicho ciclo de vida con especial importancia. He intentado adaptar la metodología Scrum a este trabajo, aunque no se ha tenido en cuenta el aspecto de los roles. En Scrum se realizan entregas parciales y regulares del producto final. Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

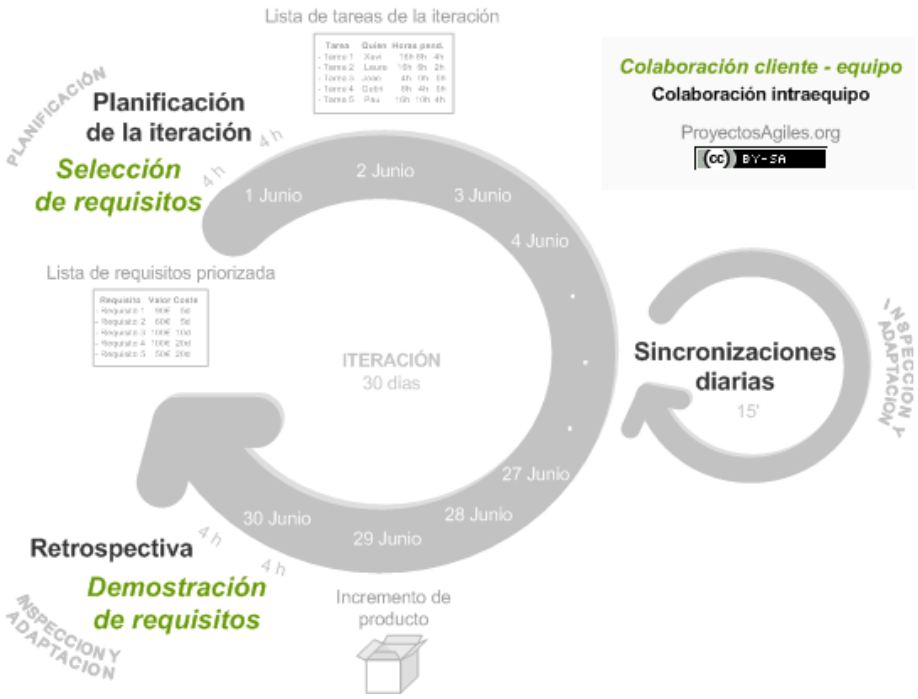


Figura 1.2: Esquema proceso Scrum

Capítulo 2

Descripción de la aplicación

Nuestra aplicación, es una aplicación web que hace de interfaz con el sistema de instrumentación CALL, y en la cual podemos cargar los datos generados por el sistema de instrumentación CALL, y así visualizar los datos obtenidos. En este capítulo, describiremos todos los aspectos de nuestra aplicación, su funcionamiento y sus objetivos.

2.1. Acceso a la aplicación

Para tener acceso a nuestra aplicación, simplemente hay que acceder desde un navegador a la siguiente dirección: rstudio.pcg.ull.es/TFG .

Para realizar la aplicación, en un principio hemos accedido de modo local y con el desarrollo de la aplicación hemos accedido mediante túneles ssh. O bien acceder al repositorio de la aplicación (se trata de licencia libre y código abierto) y seguir las instrucciones de instalación.

2.2. Descripción de uso

En esta sección describiremos el uso de cada apartado de nuestra aplicación.

2.2.1. Cargar Datos

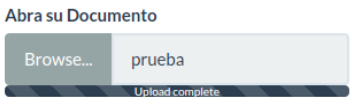


Figura 2.1: Cargamos el Documento

En nuestra aplicación dispondremos de un botón, el cual nos permite cargar los datos que necesitamos, para estudiar el comportamiento de los mismos con nuestra aplicación. El código lo dividiremos en dos, primero para generar el aspecto visual en la aplicación, que corresponde con el código llamado ui.r y después el código llamado server.r que corresponde al código que hace que funcione todo.

2.2.2. Selección de tipo de Gráficos

En esta sección podremos seleccionar el tipo de gráfico que queremos mostrar con nuestros datos. Cada gráfico nos aporta una visión de los datos diferente, y podremos seleccionar uno u otro según la necesidad que tengamos para ver los datos.

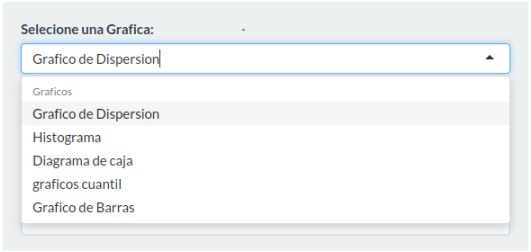


Figura 2.2: Seleccionamos el tipo de gráfico

Hemos puesto 5 tipos de gráficos:

- Gráfico de Dispersión.

  Este gráfico está por defecto y se accede directamente con el comando "plot"

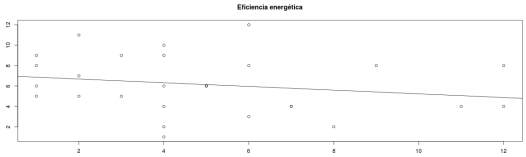


Figura 2.3: Gráfico de dispersión

- Histograma.

  Para mostrar el gráfico histograma usamos el comando "hist"

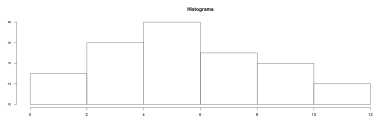


Figura 2.4: Histograma

- Diagrama de cajas.

  Para mostrar el gráfico de diagrama de cajas usamos el comando "boxplot"

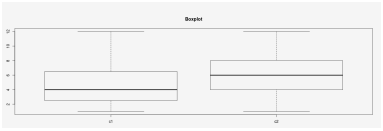


Figura 2.5: Diagrama de cajas

- Gráficos cuartiles.

  Para mostrar el gráfico de cuartiles usamos el comando "qqplot"

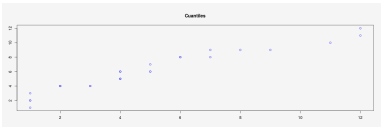


Figura 2.6: Gráfico de cuartiles

- Gráfico de barras.

  Para mostrar el gráfico de barras usamos el comando "barplot"

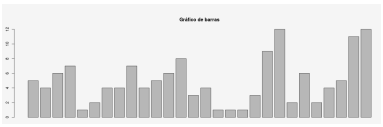


Figura 2.7: Gráfico de barras

2.2.3. Control del Gráfico

En esta sección veremos los controles básicos de nuestra gráfica, este control es muy necesario puesto que nos permite ajustar los ejes de la gráfica, para así poder ver los ajustes de unos datos en concreto u otros. En la aplicación dispondremos del ajuste manual de los ejes y también de un slider, por otro lado también dispondremos de un apartado donde modificar los nombres de los ejes.

2.2.3.1. Control Manual

En este apartado podremos controlar los ejes de nuestra gráfica de una forma más precisa, y ajustarla para poder ver diferentes puntos más de cerca, para poder trabajar con ellos. El control manual hace que se ajuste automáticamente el control del slider y viceversa, lo que provoca que si ajustamos un determinado rango en la opción manual, este rango se actualizará también en el control por slider.

Figura 2.8: Control manual de los ejes del gráfico

2.2.3.2. Control mediante slider

El control mediante slider es una barra con el máximo y el mínimo de nuestros datos en ambos extremos y nos proporciona un ajuste rápido y fácil de nuestra gráfica. Este ajuste al igual que el manual, hace que el ajuste manual se actualice con los ajustes del slider automáticamente. El slider siempre tendrá el máximo y el mínimo de nuestro datos y nos dejará ajustarlo solamente en el rango que proporcione la diferencia del mínimo y el máximo de nuestros datos.

Figura 2.9: Control de los ejes del gráfico mediante slider

2.2.3.3. Edición de nombre de ejes

En esta parte del código tendremos la posibilidad de modificar los nombres de los ejes del gráfico, según el tipo de datos que queramos cargar, podremos ponerle nombres a los ejes.

Figura 2.10: Edición del nombre de los ejes

2.2.4. Seleccionar Ecuaciones

En nuestro proyecto, una vez cargado los datos y dibujado la gráfica, dispondremos de la opción de seleccionar una ecuación y poder ver sus efectos en dicha gráfica. Hay diferentes tipos de ecuaciones y cada una de ellas se representan en la gráfica de diferentes formas y con un color específico para cada una de ellas. Estas ecuaciones nos proporcionan un ajuste aproximado a los datos que cargamos y nos sirven para ver cual sería el ajuste más óptimo de las mismas.

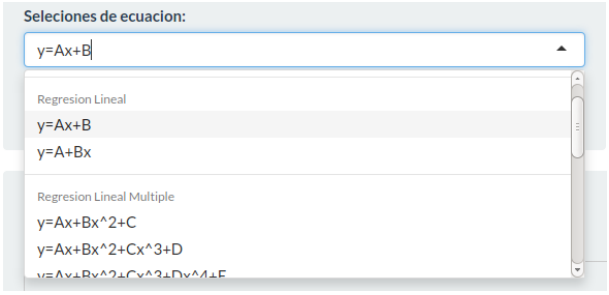


Figura 2.11: Selección de Ecuaciones

Entre las ecuaciones que tenemos a nuestra disposición tenemos:

- Regresión Lineal.

$$"y = A + Bx + C" \tag{2.1}$$

$$"y = A + Bx^3" \tag{2.2}$$

$$"y = A + Bx + Cx^2 + D" \tag{2.3}$$

$$"y = A + Bx + Cx^2 + Dx^3 + E" \tag{2.4}$$

$$"y = A + Bx + Cx^2 + Dx^3 + Ex^4 + F" \tag{2.5}$$

- Logarítmicas.

$$"y = A * ln(x) + B" \tag{2.6}$$

- Exponencial.

$$"y = A^x + B" \tag{2.7}$$

$$"y = Ax + ln(B) \tag{2.8}$$

- Compuesto.

$$"ln(y) = (ln(A) * x) + ln(B)" \tag{2.9}$$

$$"y = A^x * B" \tag{2.10}$$

Para usar las diferentes ecuaciones, hemos utilizado un comando muy específico de R, y no es otro que "lm", a continuación mostraremos las diferentes combinaciones del comando para ajustarla a la ecuación especificada. Una vez seleccionada la ecuación que queremos utilizar, la guardaremos y la compararemos en nuestro código con las ecuaciones que disponemos, y según la ecuación seleccionada se ejecutará una fórmula u otra.

2.2.5. Información

En este apartado mostraremos los datos que una vez seleccionada la ecuación para nuestra gráfica, nos devolverá todos los datos relevantes de dicha ecuación, entre estos datos están: la propia fórmula que se usa para la ecuación, los datos residuales, los coeficientes etc.

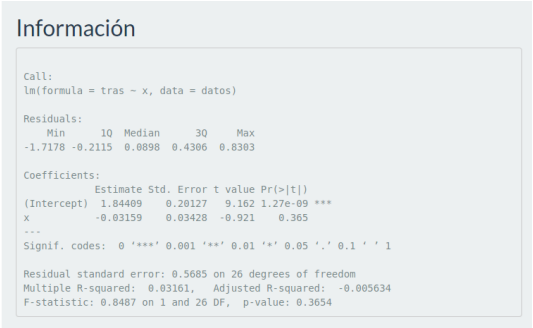


Figura 2.12: Información del gráfico y de las ecuaciones

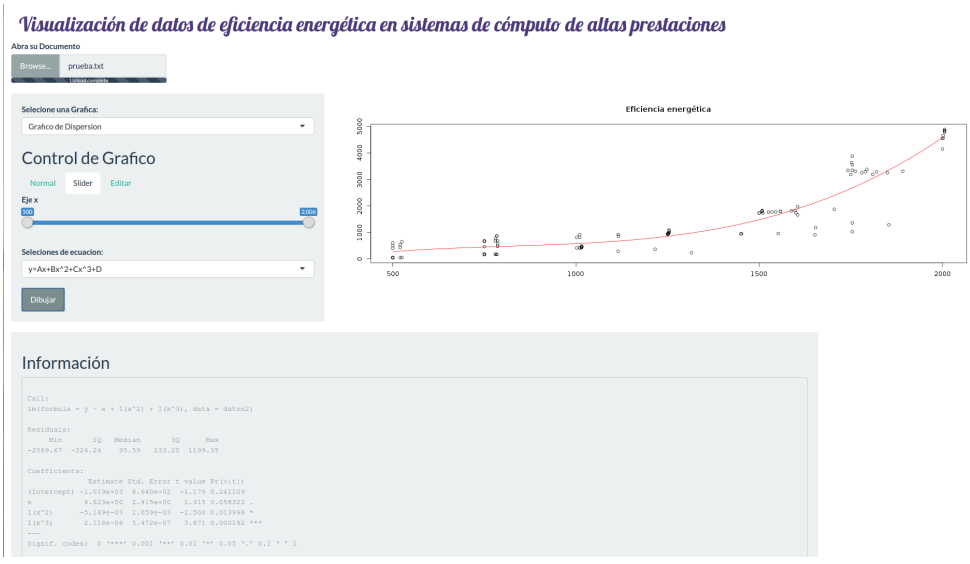


Figura 2.13: Ejemplo de la aplicación

2.3. Desarrollo del Código

2.3.1. Carga de datos

- Código ui.r

  En este código lo que vemos es la parte visual de la carga de datos.

```
#CARGA DE DATOS
fileInput(
  "Datos",
  "Abra su Documento",
  accept = c("text/csv",
             "text/comma-separated-values,text/plain",
             ".csv")

),
```

Lo que podemos ver es como creamos un `fileInput` y las opciones que tiene, como el nombre que le vamos a dar, el mensaje que le pondremos para que el usuario lo vea y las condiciones para que se abra como que sea un txt/csv.

- Código server.r

  Sin embargo, en esta parte del código veremos como cargamos los datos, vemos como se hace un vuelco de los datos a `inFile`, y después guardamos cada dato con su separación (`sep`= ""), en nuestro dataframe datos.

```
output$plots <- renderPlot({


  inFile <- input$Datos

  if (is.null(inFile))
    return(NULL)



  datos <- read.delim(inFile$datapath,
  header = FALSE, sep = " ")
})
```

2.3.2. Selección de gráfico

- Código ui.r

  En este código veremos como creamos un `selectInput` que permitirá al usuario elegir el tipo de gráfico que desea ver.

```
selectInput("Graficos", "Selecione una Grafica:",
            list(
              `Graficos` = c(
                "Grafico de Dispersion",
                "Histograma",
                "Diagrama de caja",
                "graficos cuantil",
                "Grafico de Barras"
              )
            )),
```

- Código server.r

  En este trozo de código vemos como guardamos en *(Grafi) la selección hecha por el usuario y después comparamos si esa selección es un tipo de gráfico que tenemos guardado. Si es así, lo que hacemos es hacer un plot() pasándole los datos que queremos ver, los nombres de los ejes y el título del gráfico.*

```
Grafi <- input$Graficos

if (Grafi == "Grafico de Dispersion") {
  if (t == 2) {
    plot(datos2, xlab <- input$Textx,
         ylab <- input$Texty,
         main = "Eficiencia energetica")

  }

  if (t == 1) {
    plot(datos, xlab <- input$Textx,
         ylab <- input$Texty,
         main = "Eficiencia energetica")
  }
}
```

2.3.3. Control del gráfico

- Código ui.r

  Como podemos ver en el código, este apartado dispone de de varios `tabPanel`, que nos permitirán navegar por el control manual, el slider y editar, cada uno de ellos hace una cosa diferente, en la pestaña normal, vemos como creamos dos `numericInput` para poder guardar los números que el usuario introduzca, en un principio por defecto estos valores valdrán 0. En el slider, creamos un `sliderInput` para crear dicho slider y le pondremos las opciones de los nombres para poder manejarlos en el código, los valores por defecto y el step. Por último en el menú editar, crearemos dos `textAreaInput` para guardar los nombres de los ejes que el usuario introduzca, y ajustaremos el tamaño del área de visualización.

```
titlePanel("Control de Grafico"),
tabsetPanel(
  id = "tabset",

  tabPanel(
    "Normal",

    #SELECCION DE RANGOS MANUALES

    numericInput("captionmin", "Xmin", 0),
    numericInput("captionmax", "Xmax", 0)

  ),
  tabPanel("Slider",
           #SELECCION DE RANGO CON SLIDER
           sliderInput(
             "ejex", "Eje x", 0, 0,
             value = c(0, 0), step = 1
           )),
  tabPanel(
    "Editar",

    textAreaInput("Texty",
                  "Nombre eje y",
                  width = "250px",
                  height = "30px"),
    textAreaInput("Textx",
                  "Nombre eje x",
                  width = "250px",
                  height = "30px")
  )
),
```

- Control mediante Slider
  - Código server.r

    En cuanto al código server del control por slider, lo que hacemos es estar observando a ver si hay algún movimiento de los slider, una vez se activen dichos movimiento, el slider se actualizará con los valores que tenga en ese momento, y no sólo se actualizan los valores del slider, sino que también se actualizan los valores del rango manual.

```
output$plots <- renderPlot({
  if (v$doPlot == FALSE)
    return()

  isolate({
    if (input$tabset == "Slider") {
      MIN <- min(datos$V1)
      MAX <- max(datos$V1)
      val  <- input$ejex

      observe(updateSliderInput(
        session,
        "ejex",
        value = val,
        min = MIN,
        max = MAX,
        step = 1
      ))
      observe(updateNumericInput(session,
                                 "captionmin", value = val[1]))
      observe(updateNumericInput(session,
                                 "captionmax", value = val[2]))
    } else {
      xmin <<- input$captionmin
      xmax <<- input$captionmax
      val2[1] <- xmin
      val2[2] <- xmax
      observe(
        updateSliderInput(
          session,
          "ejex",
          value = val2,
          min = minimo,
          max = maximo,
          step = 1
        )
      )
    }
```

- Editar gráfico
  - Código server.r

    En esta parte del código simplemente guardamos los textos introducidos por el usuario en `textx` y `texty`, que posteriormente en la llamada a la función que muestra la gráfica, se le pasaran estos valores y podremos ver como estos nombres introducidos por el usuario se muestran en los nombres de los ejes de la gráfica.

```
if (input$tabset == "Editar") {

  texty <- input$Texty
  textx <- input$Textx


}
```

2.3.4. Selección de ecuaciones

- Código ui.r

  En este código crearemos un `selectInput` para poder mostrar un menú desplegable con los diferentes tipos de ecuaciones que podemos seleccionar.

```
selectInput(
  "Ecuacion",
  "Seleciones de ecuacion:",
  list(
    `Ecuacion` = c("Ecuaciones"),
    `Regresion Lineal` = c(
      "y=Ax+B",
      "y=Ax+Bx^2+C",
      "y=Ax+Bx^2+Cx^3+D",
      "y=Ax+Bx^2+Cx^3+Dx^4+E"
    ),
    `Logaritmica` = c("y=A*ln(x)+B", "y=A+B*ln(x)"),
    `Exponencial` = c("y=A^x+B", "y=Ax+ln(B)"),
    `Compuesto` = c("y=A^x*B", "ln(y)=(ln(A)*x)+ln(B)")

  )
),
actionButton("go", "Dibujar")

),
```

- Código server.r

  En esta parte de código guardaremos la opción seleccionada por el usuario en expression y posteriormente la compararemos con los nombres de las ecuaciones que tenemos en nuestro programa. Una vez coincida con la ecuación seleccionada, guardaremos en `x` e `y` los datos pertinentes y después se lo pasaremos a la función lm(), a parte también le pasaremos el dataframe, y guardaremos el resultado en `regresion`, con este resultado podemos trazar un ajuste de mínimos cuadrado en nuestra gráfica, en este caso con abline. También cabe destacar que en el apartado del código de Información, lo haremos desde aquí con la función summary() y pasándole los datos generados, guardados en `regresion`, podremos visualizar diversos datos.

```
expresion <- input$Ecuacion

        if (expresion == "y=Ax+B" || expresion == "y=A+Bx") {
          if (t == 1) {
            y <- datos$V2
            x <- datos$V1
            regresion <- lm(y ~ x, datos)
            summary(regresion)
            abline(regresion)

          }
          else{
            y <- datos2$c2
            x <- datos2$c1
            regresion <- lm(y ~ x, datos2)
            summary(regresion)
            abline(regresion)
          }
          output$modelSummary <- renderPrint({
            summary(regresion)
          })

        }
```

2.4. Caso de Uso

En esta sección mostraremos un ejemplo de cómo, a partir de los datos generados por CALL, se crea una gráfica en la que podremos ajustar los mínimos cuadrados para poder verlo reflejado, y también podremos ver toda la información aportada por el ajuste. El modelo que vamos a proponer es lo suficientemente general como para aplicar nuestro análisis de rendimiento energético, por lo que presentaremos un experimento simple en el que podremos ilustrar el análisis de rendimiento energético. El código que instrumentaremos es el producto de matrices. Se han definido dos experimentos, uno que monitoriza la inicialización de las matrices (`init`) y el cálculo del producto (`mat`)

- Código:

```
# pragma cll for( N = MIN ; N <= MAX ; N += STRIDE )

/* TIA experiment : initialization */

# pragma cll init CLOCK , \
 EML_ENERGY_RAPL = init [0]* N * N
  for ( i =0; i < N ; i ++)
   for ( j =0; j < N ; j ++)
    A (i , j ) = B (i , j ) = ( i == j );
# pragma cll end init

/* TIA experiment : multiplication */

# pragma cll mat CLOCK , \
 EML_ENERGY_RAPL = mat [0]* N * N * N
  for ( i = 0; i < N ; i ++) {
   for ( j = 0; j < N ; j ++) {
    sum = 0;
    for( k = 0; k < N ; k ++)
     sum += A (i , k ) * B (k , j );
    C (i , j ) = sum ;
   }
  }

# pragma cll end mat
# pragma cll end for
```

Tanto la inicialización como la multiplicación están incluidas en las directivas #pragma cll delimitando el comienzo y el final de cada experimento. A continuación, se muestra una lista que indica las métricas deseadas: tanto la métrica CLOCK estándar (tiempo del sistema) como la métrica EML ENERGY RAPL (consumo total de energía visto por RAPL [20]) proporcionado por el controlador EML [19]. Finalmente, el usuario puede proporcionar una lista de factores y variables. Las constantes se indican con el nombre del experimento más el array notación del modelo (init [0]).

- Resultados:

  Con un promedio de 10 ejecuciones del código instrumentado con diferentes tamaños de matriz, obtenemos un archivo de seguimiento con los

siguientes datos de consumo de energía:

	Energy [J]	
Size	`init`	`mat`
500	0.19	40.02
750	0.23	173.94
1000	0.31	419.80
1250	0.48	954.19
1500	0.66	1777.57
1750	0.89	3290.00
2000	1.14	4651.50

Figura 2.14: Tabla datos de rendimiento obtenidos con CALL

- Gráfica:

  Con los resultados obtenidos y seleccionando la columna de Size y la de mat, y seleccionando la fórmula

  $$"y = A + Bx + Cx^2 + Dx^3" \tag{2.11}$$

  nuestra aplicación nos genera el siguiente gráfico con el ajuste de mínimos cuadrado.

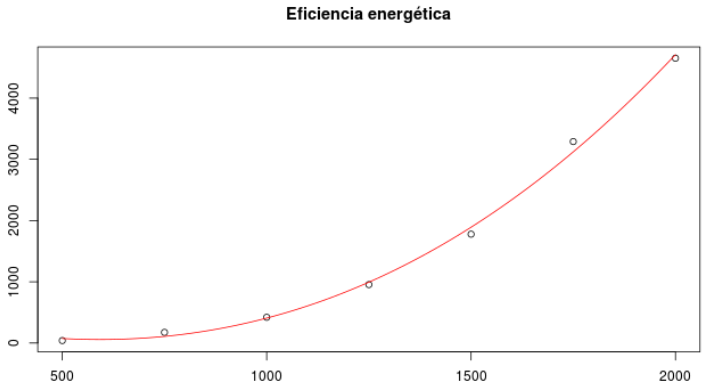


Figura 2.15: Gráfica generada por la tabla

- Resultados:

  Ahora nuestro programa nos genera la información necesaria, sobre la fórmula utilizada y los datos generados por el ajuste de mínimos cuadrados.

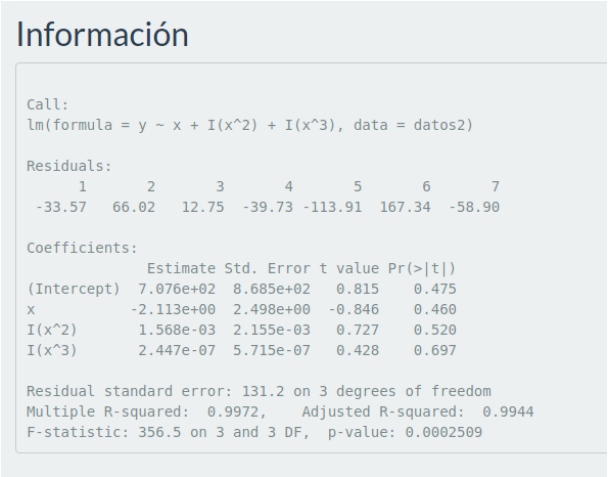


Figura 2.16: Información generada por la tabla

Como vemos reflejado en los datos, el ajuste tiene un coeficiente de regresión bueno, pero los coeficientes tienen errores en el mismo orden de magnitud que su valor. Procederemos a ajustarla con otra ecuación, en este caso:

$$"y = A + Bx^3" \tag{2.12}$$

que como vemos en la información es la que mejor se ajusta. Por lo que el gráfico que nos genera es:

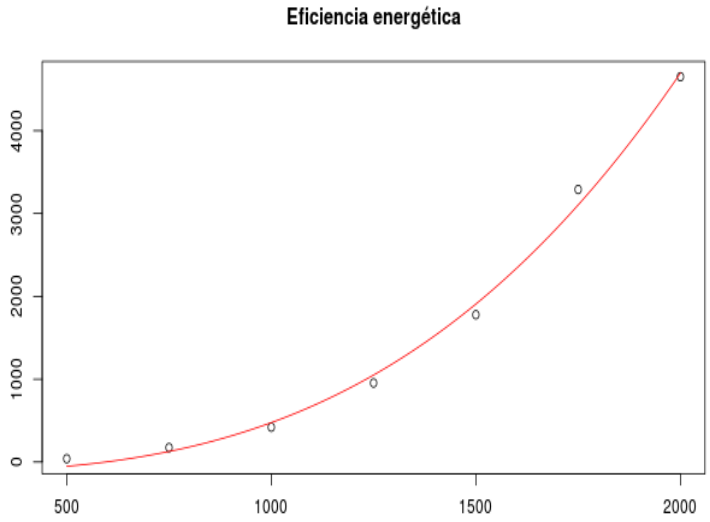


Figura 2.17: Gráfica 2 generada por la tabla

Y la información que nos genera es:

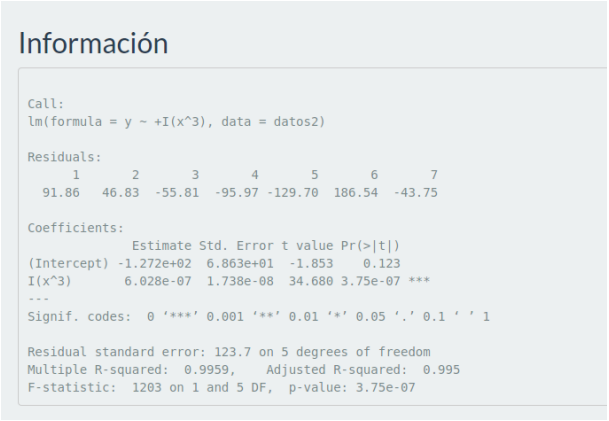


Figura 2.18: Información 2 generada por la tabla

Como podemos observar en esta nueva información generada, vemos que el ajuste es mejor, obteniendo errores en los coeficientes en torno al 10 %.

Capítulo 3

Entornos para el despliegue de la aplicación

En este capítulo describiremos las diferentes tecnologías y frameworks que hemos usado para el desarrollo de nuestra aplicación. Veremos la descripción de cada una de estas tecnologías y frameworks, también describiremos los pasos a realizar para efectuar su correcta instalación y configuración en nuestro sistema.

3.1. R

Figura 3.1: Logo de R

R [12, 13] es un lenguaje abierto y orientado a objetos, está enfocado para el análisis estadístico y para la representación gráfica de los datos obtenidos. Se trata de un lenguaje interpretado, basado en comandos o instrucciones. Este lenguaje es muy complejo y tiene una curva de aprendizaje muy complicada, pero a la hora del manejo de datos estadísticos es muy efectivo, permitiéndonos generar gráficos de de alta calidad. Su sintaxis es muy parecida a la de C ó C++. R es un lenguaje que está siempre evolucionando. En él existen diversas bibliotecas que facilitan su manejo desde varios lenguajes interpretados como Python, Ruby o Perl, y también existen proyectos que nos permiten utilizarlo desde Java o .net. Con R tenemos la facilidad de preparar un gran volumen de datos, con los que se puede preparar una visualización inmejorable y sacar conclusiones de ellas, por lo que para nuestro proyecto se nos hace muy interesante.

Decir también que es un lenguaje gratuito y abierto.

Algunas de las características y ventajas de la programación en R son:

- El lenguaje de programación R es un proyecto colaborativo y abierto, los desarrolladores pueden descargar el código de forma gratuita y modificarlo para incluir mejoras.
- Es un lenguaje interpretado, funciona mediante comandos.
- R proporciona una amplia gama de herramientas estadísticas que incluyen análisis de datos y generación de gráficos. Este lenguaje tiene capacidad de generar gráficos de alta calidad. Estas características lo convierten en una potente herramienta de cálculo.
- Gracias a este lenguaje de programación los científicos y analistas pueden manejar grandes volúmenes de datos.
- Puede integrarse con distintas bases de datos. Una de las ventajas más importantes de R es que funciona con diferentes tipos de hardware y software (Windows, Unix, Linux...)
- El lenguaje R ofrece la posibilidad de cargar bibliotecas y paquetes con diversas funcionalidades lo que permite a los usuarios extender su configuración básica.
- La comunidad en torno a R es muy activa por lo que es sencillo encontrar soluciones rápidamente a los problemas que los usuarios se puedan encontrar.

3.1.1. Instalación R

Para empezar a trabajar con R necesitamos instalarlo en nuestro ordenador, y para eso, lo primero que hay que hacer es visitar `https://cran.r-project.org/` y descargar el ejecutable para Linux, Mac OS X o Windows. Una vez descargado, el proceso es el habitual para cualquier programa o aplicación web instalada en un sistema operativo:

- Hay que descargar el paquete desde los repositorios oficiales, `https://cran.rstudio.com/`
- Para instalar la versión más reciente de R, primero debe agregar el repositorio CRAN a su sistema como se describe aquí:
  - Debian: `https://cran.rstudio.com/bin/linux/debian/`
  - Ubuntu: `https://cran.rstudio.com/bin/linux/ubuntu/`
- Antes de instalar el programa podemos instalar las dependencias opcionales tcl tk y gcc-fortran. (estas dependencias son opcionales. tcl y tk son usados por algunos paquetes para R, mientras que gcc-fortran se utiliza para compilar algunos de esos paquetes.)
- En Ubuntu/Debian: sudo apt-get install r-base

- En Fedora: su -c 'yum install R'
- En Arch Linux: sudo pacman -S r
- Una vez terminada la instalación, podemos ejecutar R con el comando: R

3.2. R-Studio

Figura 3.2: Logo de RStudio

Rstudio [7] es un entorno de desarrollo integrado (IDE) para el lenguaje R. Es un software libre y se puede ejecutar en varias plataformas (linux, Windows, Mac) o también desde la web usando Rstudio Server, la cual utilizaremos en nuestro proyecto. Rstudio incorpora una consola, un editor de resaltado de sintaxis que nos permite la ejecución directa del código así como herramientas para trazar, la historia, la depuración y la gestión del espacio de trabajo. Algunas de sus características son:

- Resaltado de sintaxis, finalización de código y sangría inteligente.
- Ejecutar código R directamente desde el editor fuente.
- Saltar rápidamente a las definiciones de funciones.
- Ayuda y documentación de Integrated R.
- Administrar fácilmente múltiples directorios de trabajo mediante proyectos.
- Navegador de espacio de trabajo y visor de datos.
- Depurador interactivo para diagnosticar y corregir errores rápidamente.
- Extensas herramientas de desarrollo de paquetes.
- Creación con Sweave y R Markdown.
- Integra las herramientas necesarias que utiliza R en un solo entorno.

- Incluye potentes herramientas de codificación diseñadas para mejorar su productividad.
- Permite la navegación rápida a archivos y funciones.
- Facilita el inicio de nuevos proyectos o la búsqueda de proyectos ya creados.
- Contiene soporte integrado para Git.
- Permite la creación de HTML,PDF, Word y presentaciones en diapositivas.
- Soporta gráficos interactivos con shiny.

3.2.1. Ventajas de Rstudio

Algunas de las ventajas que tiene Rstudio son:

- Es un entorno de desarrollo integrado por lo que nos hace muchísimo más fácil el trabajo con R. Una vez dentro de la aplicación nos encontramos con la pantalla dividida en cuatro espacios de trabajo, donde la ventana superior izquierda dispone de un editor adaptado para escribir código en R, en la ventana superior derecha aparece una lista de los objetos (ficheros, vectores, funciones, etc.) que están en el área de trabajo, la ventana inferior izquierda nos encontramos con la consola para utilizar R de forma interactiva, en la ventana inferior derecha aparecen varias pestañas en las que dispondremos de diferentes utilidades, en esta ventana es donde podremos visualizar nuestros gráficos.
- Otra gran ventaja que me he encontrado es la ayuda de los comandos mediante la tecla de tabulación, puesto que una vez escrito el comando, podemos situar el cursor entre los paréntesis de este, apretar el tabulador y se nos abrirá una lista de los principales argumentos que se nos ajusta a nuestro código y a aparte también una ayuda sobre la función de cada argumento.
- Lo más que me ha impresionado y que por eso hemos usado este entorno de trabajo, es la visualización de datos en el área de trabajo. Puesto que podemos cargar cualquier dato y con algunas funciones (por ejemplo la función plot(datos)) nos muestra una gráfica con todos los datos.
- Uno de los comandos que podemos introducir y que nos facilita la ayuda de cualquier comando o función que necesitemos es el símbolo " ? " seguido del comando, esto nos abrirá en el área de visualización una ayuda bastante extensa sobre todas las funcionalidades del comando.

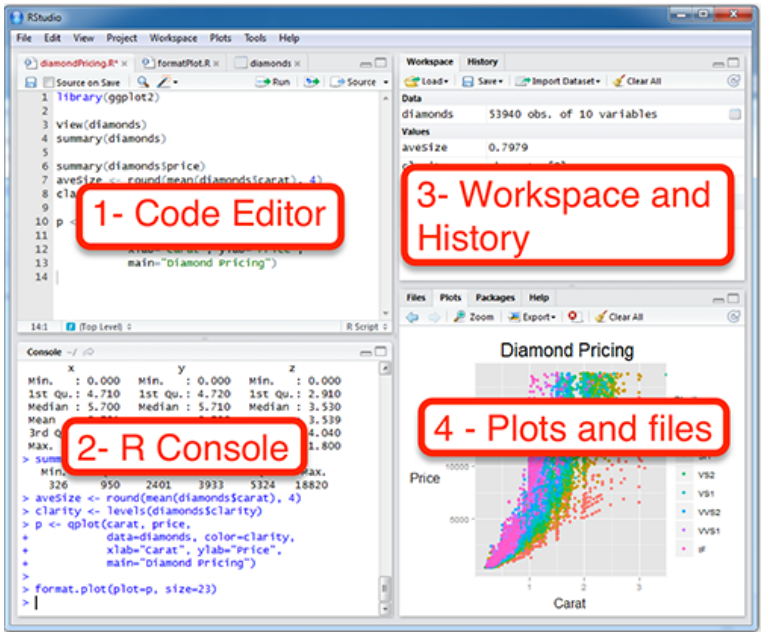


Figura 3.3: Imagen de plataforma Rstudio

Rstudio nos permite crear proyectos nuevos, aplicaciones y packages[5] en R. También dispone de muchas funciones de trabajo como pueden ser búsqueda de código, ordenación de código, instalación de paquetes R, importar dataset, entre muchas otras.

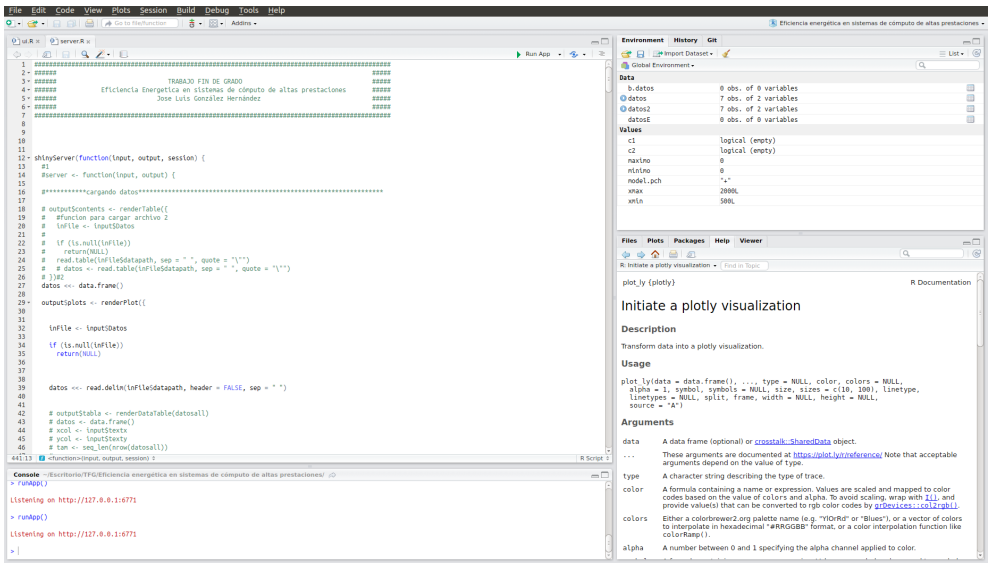


Figura 3.4: Área de trabajo

3.2.2. Instalación de Rstudio

Para la instalación de Rstudio tendremos que ir al sitio oficial `https://www.rstudio.com/products/rstudio/download/` y seleccionar la plataforma en la que vayamos a instalar Rstudio, la descargamos, y después simplemente se accede al instalador mediante el ejecutable o si es en linux mediante el software center de Ubuntu.

3.3. Rstudio server

Como ya hemos comentado en capítulos anteriores, vamos a utilizar Rstudio pero esta vez el Server, que nos proporciona la ejecución de la plataforma en web. Esto nos proporciona una serie de beneficios para la realización de nuestro proyecto, algunos de ellos son:

- La capacidad de acceder a sesiones R desde cualquier ordenador en cualquier lugar.
- Fácil intercambio de código, datos y otros archivos.
- Permitir a varios usuarios compartir el acceso a los recursos de computación más poderosos (memoria, procesadores, etc.).
- Instalación y configuración centralizada de paquetes R, y otras bibliotecas de soporte.

El acceso a Rstudio Server [8] se hará a través de un navegador, y de forma predeterminada se ejecutará desde el puerto 8787 y permitirá las conexiones de todos los usuarios remotos, después de la instalación, que explicaremos en el siguiente apartado, debe de ser capaz de acceder desde un navegador web desde la siguiente dirección: http://server-ip:8787. Rstudio solicitará el nombre de usuario y la contraseña. Para la administración de usuarios y contraseñas se puede usar la herramienta estándar de administración de usuarios de linux como useradd, userdel, etc. Cada usuario debe crearse con un directorio de inicio. En el caso de que no se pueda acceder al servidor después de la instalación, se debe ejecutar el comando para generar un diagnóstico, el comando sería: sudo rstudio-server verify-installation.

3.3.1. Instalación de Rstudio Server en Debian/Ubuntu.

Para la instalación de Rstudio Server se necesita tener instalado la versión más reciente de R, como comentamos en apartados anterior. Una vez instalado R, procederemos a la instalación de Rstudio Server. Lo primero que se debe hacer es ejecutar los siguientes comandos desde nuestra terminal:

- sudo apt-get install gdebi-core
- sudo gdebi rstudio-server-package.deb

Y ya tendremos instalado Rstudio Server en nuestro sistema.

3.3.2. Administración de Rstudio Server.

Las tareas de administración de RStudio Server se realizan utilizando "rstudio-server utility". Esta utilidad permite detener, iniciar y reiniciar el servidor, enumerar y suspender sesiones de usuario, desconectar el servidor, así como la posibilidad de actualizar en caliente una versión en ejecución del servidor.

- Reiniciar servidor:
  -sudo rstudio-server restart
- Iniciar servidor:
  -sudo rstudio-server start
- Parar servidor:
  -sudo rstudio-server stop
- Comprobación del estado actual:
  -sudo rstudio-server status

También nos encontramos con la gestión de sesiones activas, por lo que enumeraremos los comandos para el control de estas sesiones:

- Para enumerar todas las sesiones actualmente activas:
  -sudo rstudio-server active-sessions
- Para suspender una sesión individual:
  -sudo rstudio-server suspend-session pid
- Para suspender todas las sesiones en ejecución:
  -sudo rstudio-server suspend-all

3.3.3. Configuración de Rstudio Server.

Para la configuración del servidor de Rstudio, tendremos que acceder a dos archivos, rserver.conf y rsession.conf, que se encuentran el la siguientes rutas:

- /etc/rstudio/rserver.conf
- /etc/rstudio/rsession.conf

Hay que tener en cuenta que estos archivos no existen por defecto, por lo que tendremos que crearlos para crear una configuración personalizada.
Hay diversas configuraciones que podremos realizar, las más comunes son:

- Puerto y dirección de la red: Por defecto esta predeterminado acceder a Rstudio Server a partir del puerto 8787, pero si deseamos cambiar dicho puerto, debemos editar el archivo rserver.conf y agregar: www-port="numero del puerto". Para cambiar la dirección de red, editaremos el archivo agregando o modificando: www-address="dirección ip".

  Para la configuración de nuestro proyecto utilizaremos: www-port=8787 www-address=127.0.0.1

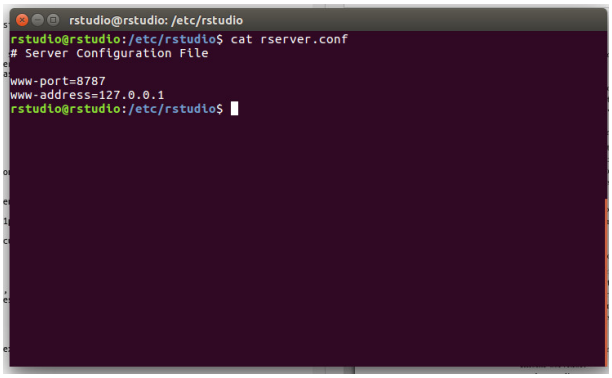


Figura 3.5: Configuración Puerto y dirección red

3.3.4. Ajustes adicionales

El archivo rsession.conf nos permite crear diversos ajustes adicionales, algunos de ellos son:

- Tiempo de espera de sesión:
  -session-timeout-minutes=30
- Camino de biblioteca de paquetes:
  -r-libs-user= /R/packages
- Repositorio CRAN:
  -r-cran-repos=dirección web

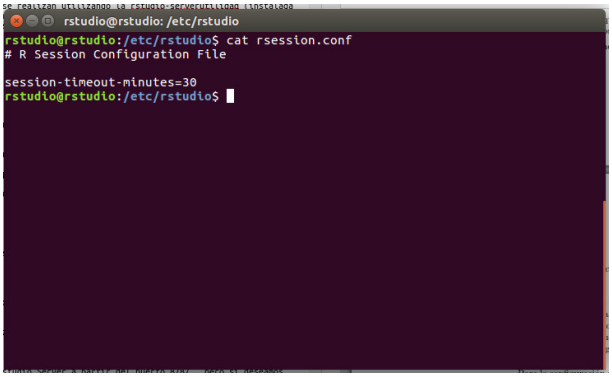


Figura 3.6: Ajustes adicionales

3.4. Shiny

Shiny [9] es un paquete de R y nos permite crear aplicaciones web interactivas a partir de los scripts creados en Rstudio. Por lo que nos es de gran utilidad, puesto que nuestro proyecto necesita una web en la que poder visualizar los datos aportados y poder hacer ajustes. Con lo que Shiny nos da esa posibilidad, y nos sirve para poder ir comprobando el aspecto y el funcionamiento de nuestra aplicación. Una vez acabada la aplicación y en perfecto funcionamiento, configuraremos para que nuestra aplicación se abra directamente desde Shiny Server [10] con Rstudio detrás.

3.4.1. Instalación de Shiny

Para realizar la instalación de Shiny, necesitaremos tener instalados tanto R como Rstudio, una vez instalado lo anteriormente mencionado, procederemos a la instalación de Shiny. Mediante la terminal escribiremos el siguiente comando:

- install.packages("shiny")

Con esto se nos instalará Shiny y para usarla en nuestro programa tendremos que agregar la librería a nuestro código con:

- library(shiny)

Y con esto podemos comprobar que todo funciona, ejecutando nuestra aplicación.

3.5. Shiny Server

Como hemos comentado en el capítulo anterior, una vez acabada la aplicación, lo que necesitamos es alojarla en un servidor para que se ejecute con Rstudio detrás, esto nos lo proporciona Shiny Server. Este es un servidor proporcionado por Rstudio en el que podemos alojar nuestras aplicaciones y administrarlas en la web. Este software es gratuito y de código abierto.

3.5.1. Instalación de Shiny Server

Para proceder a la instalación de Shiny Server, debemos tener instalados previamente el paquete Shiny en nuestro sistema, una vez instalado Shiny, procederemos a la instalación de Shiny Server mediante los siguientes comandos en nuestra terminal:

- Vamos a instalar Shiny Server con la herramienta GDebi, así que primero tenemos que instalarlo.

  -sudo apt-get install gdebi-core

- Dependiendo que sistema estemos utilizando la dirección web será una u otra, en nuestro caso es Ubuntu 64 bits, por lo que el comando será:

  -wget -O shiny-server.deb (dirección de descarga) Para ver otra versión o una versión más actualizada, podemos acceder a ella desde `https://www.rstudio.com/products/shiny/download-server/`

- Ahora hay que instalar con GDebi el archivo descargado anteriormente.

  - sudo gdebi shiny-server.deb

Una vez instalado Shiny Server, debe estar ejecutándose en el puerto 3838, y podemos ver una pantalla de bienvenida predeterminada en: http://ip servidor:3838/.

3.5.2. Administración de Shiny Server

Para el control de Shiny Server utilizaremos Systemd, que es una plataforma de administración y configuración para Linux.

Para iniciar, detener o reiniciar el servidor manualmente, se pueden utilizar los siguientes comandos.

- sudo systemctl start shiny-server
- sudo systemctl stop shiny-server
- sudo systemctl restart shiny-server

Para cargar la configuración, pero mantener el servidor y todos los procesos brillantes en ejecución sin interrupción, puede utilizar el comando systemctl para enviar una SIGHUPseñal:

- sudo systemctl kill -s HUP - -kill-who=main shiny-server

  Esto hará que el servidor se reinicialice, pero no interrumpirá los procesos actuales ni ninguna de las conexiones abiertas al servidor.

Podemos utilizar los comandos enable/ disable para controlar si Shiny Server debe ejecutarse automáticamente al arrancar:

- sudo systemctl enable shiny-server
- sudo systemctl disable shiny-server

3.5.3. Configuración Shiny Server

La configuración que viene predeterminada está en la ubicación: /etc/shiny-server/shiny-server.conf, si queremos modificarla, tendremos que acceder a este fichero y modificar los parámetros que deseemos.

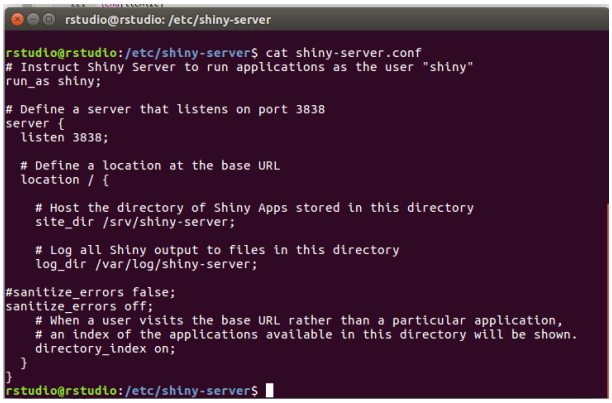


Figura 3.7: Configuración Shiny Server

Para poder alojar nuestras aplicaciones en Shiny server, debemos dirigirnos a /srv/shiny-server/ y ahí es donde crearemos una carpeta con el nombre de nuestra aplicación y en su interior pondremos el código de la misma.

Shiny server sólo dispone de algunas librerías instaladas por defecto, si queremos utilizar otras librerías, debemos seguir los siguientes pasos en la terminal:

- sudo passwd shiny
- Cambie a la cuenta brillante usando: su - shiny
- Llamar R usando R
- Instale los paquetes requeridos: install.packages("nombre de librería")

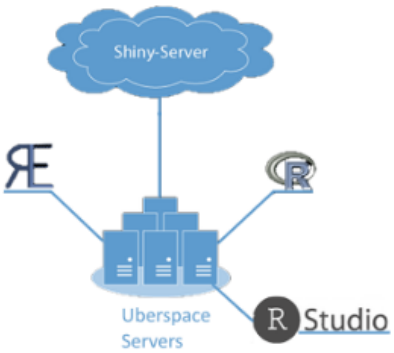


Figura 3.8: Shiny Server

3.6. Nginx

Figura 3.9: Logo Nginx

Una vez tengamos instalado todo lo anterior y en correcto funcionamiento, nos toca instalar un servidor web para que actúe de Front-end, entre los que nos recomienda la plataforma Rstudio, están Apache y Nginx, nosotros nos hemos decantado por Nginx, pero antes de proceder a la instalación, veremos que es y cuales son sus ventajas frente a otros servidores web.

Nginx [4] es un servidor web con proxy inverso, es muy ligero y posee un alto rendimiento. Cabe destacar que es un software libre, gratuito y que es multiplataforma, por lo que funciona tanto en (Linux,Windows o Mac OS X). Las Ventajas de Usar Nginx frente a otros servidores web son:

- Como ya hemos comentado anteriormente, se trata de un software multiplataforma.
- Los recursos que consume son mucho menores comparado con otros servidores web.
- Posee un alto rendimiento soportando una carga mayor y así responde mejor que otros servidores web.
- Se puede utilizar como proxy inverso cacheando el contenido de nuestros sitios web.
- Se puede instalar junto a otro servidor web como es Apache, distribuyéndose el procesamiento estático Nginx y el dinámico Apache.
- Puede usarse como balanceador de carga entre varios servidores, permitiéndonos así una mayor facilidad a la hora de escalar nuestros servidores.

Una vez vistas las ventajas que tiene usar Nginx, procederemos a la instalación del mismo, nosotros al usar linux, explicaremos la instalación en dicho sistema.

3.6.1. Instalación de Nginx en Ubuntu

Para instalar Nginx en Ubuntu, tendremos que acceder a la terminal y escribir el siguiente comando:

- sudo apt-get install nginx

Con esto ya tendríamos Nginx instalado en nuestro sistema.

3.6.2. Configuración Nginx en Rstudio Server

Una vez realizada la instalación de Nginx, procederemos a configurarlo primero con Rstudio Server [2] y después con Shiny Server. La configuración con Rstudio Server para que actúe de front-end, debemos añadir los siguientes comandos al archivo nginx.conf

```
http {

  map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
    }

  server {
    listen 80;



    location / {
      proxy_pass http://localhost:8787;
      proxy_redirect http://localhost:8787/ $scheme://$host/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_read_timeout 20d;
    }
  }
}
```

Figura 3.10: Nginx configuración para Rstudio Server 1

Si queremos que Rstudio Server sirva desde una ruta personalizada, se tiene que editar el archivo nginx.conf de la siguiente manera:

```
http {

  map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
    }

  server {
    listen 80;


    location /rstudio/ {
      rewrite ^/rstudio/(.*)$ /$1 break;
      proxy_pass http://localhost:8787;
      proxy_redirect http://localhost:8787/ $scheme://$host/rstudio/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_read_timeout 20d;
    }
  }
}
```

Figura 3.11: Nginx configuración para Rstudio Server 2

3.6.3. Configuración Nginx en Shiny Server

Al igual que en la configuración de Nginx para que actúe de Front-end para Rstudio Server, en Shiny Server [3] también debemos modificar el archivo nginx.conf de la siguiente manera:

```
http {

  map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
    }

  server {
    listen 80;



    location / {
      proxy_pass http://localhost:3838;
      proxy_redirect http://localhost:3838/ $scheme://$host/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_read_timeout 20d;
      proxy_buffering off;
    }
  }
}
```

Figura 3.12: Nginx configuración para Shiny Server 1

Si queremos que Shiny Server sirva desde una ruta personalizada, se tiene que editar el archivo nginx.conf de la siguiente manera:

```
http {

  map $http_upgrade $connection_upgrade {
      default upgrade;
      ''      close;
    }

  server {
    listen 80;


    location /shiny/ {
      rewrite ^/shiny/(.*)$ /$1 break;
      proxy_pass http://localhost:3838;
      proxy_redirect http://localhost:3838/ $scheme://$host/shiny/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_read_timeout 20d;
      proxy_buffering off;
    }
```

Figura 3.13: Nginx configuración para Shiny Server 2

3.6.4. Mi configuración final

Para usar en un principio Nginx con Rstudio Server pusimos la siguiente configuración:

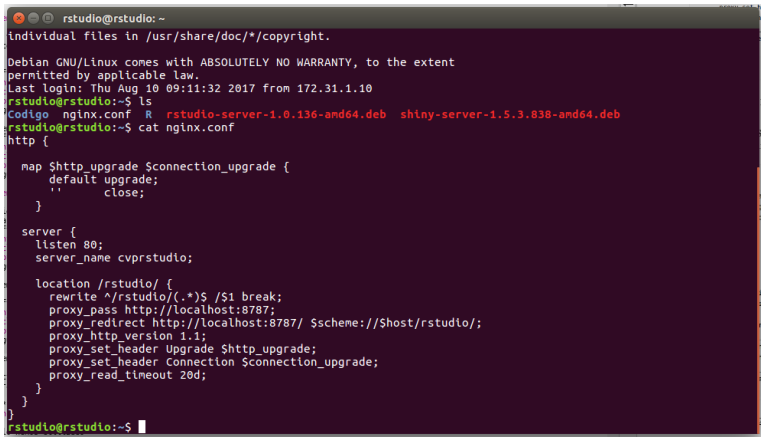


Figura 3.14: Nginx Mi configuración para Rstudi Server

En cuanto tengamos la aplicación terminada, debemos cambiar la configuración para que escuche desde el puerto 3838 que es por donde sirve Shiny Server, y así poder utilizar nuestra aplicación en la web y por detrás estar ejecutando Rstudio. Esta configuración la hemos cambiado en /etc/nginx/sites-available para que se redireccione al puerto 3838.

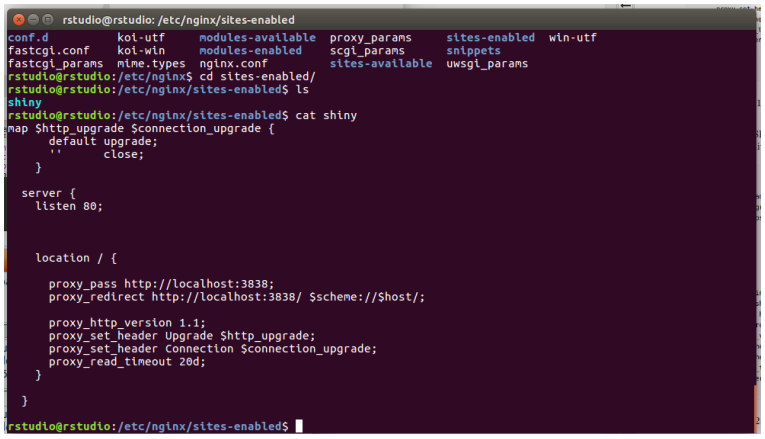


Figura 3.15: Nginx Mi configuración para Shiny Server

3.6.5. Administración de Nginx

A continuación explicaremos los diferentes comandos que disponemos para administrar Nginx.

- Para ver el estado de Nginx:
  - systemctl status nginx
- Para detener su servidor web:
  - systemctl stop nginx
- Para iniciar el servidor web cuando está detenido:
  - systemctl start nginx
- Para detener e iniciar de nuevo el servicio:
  - systemctl restart nginx
- Si simplemente está realizando cambios de configuración, Nginx puede recargar a menudo sin abandonar las conexiones. Para ello, se puede utilizar este comando:
  - systemctl reload nginx

3.7. Conectividad

Ya hemos visto todas las instalaciones y todas las herramientas que vamos a necesitar, ahora nos conectaremos por SSH [11] al cluster de cómputo Verode. Utilizaremos nuestra cuenta en la universidad para conectarnos vía ssh a la ULL, y así poder acceder a Verode, una vez en él, accederemos a Rstudio, un máquina virtual creada para que podamos instalar Nginx, Rstudio, Rstudio Server, Shiny y Shiny Server. Entonces con todo instalado, haremos la configuración necesaria, tal y como describimos anteriormente. Para conectarnos directamente desde nuestro ordenador a la máquina Rstudio, usaremos los túneles SSH que crearán una conexión directa, y con lo cual, podremos usar Rstudio Server y Shiny Server directamente.

3.7.1. Verode

Verode es un Cluster [6] de Cómputo, que dispone de diversos nodos. Los que utilizaremos para nuestro proyecto son el front-end del cluster HPC y Rstudio, estos son virtuales, a los que se le asigna un core para ejecutar tareas sencillas. También dispone de otros nodos con varios cores y GPUs para cómputo paralelo que no utilizaremos. En nuestro proyecto utilizaremos estas máquinas para ejecutar códigos científicos de los que queremos medir su rendimiento, y modelarlo con expresiones matemáticas. Estos modelos sirven luego para predecir el comportamiento de la aplicación en clusters más grandes (escalabilidad), predecir el consumo energético asociado a estos cálculos (sostenibilidad) y para planificar

la ejecución de trabajos de forma eficiente. Por lo que nuestra aplicación es una herramienta útil para los investigadores a la hora de obtener estos modelos.

A continuación mostraremos los comandos necesarios para poder acceder a Verode00 y a Rstudio. Para acceder a Verode00 necesitaremos abrir el terminal e introducir el siguiente comando:

- ssh alu0100611281@verode00.pcg.ull.es

  En este caso accedería con mi alu, proporcionado por la ULL. A continuación debemos introducir la contraseña.

Una vez en Verode00, accederemos a Rstudio y procederemos con la instalación de Nginx, Rstudio Server y SHiny Server, debemos introducir el siguiente comando:

- ssh rstudio@rstudio.pcg.priv

  Introducimos la contraseña y procedemos a la instalación y configuración de todo lo necesario.

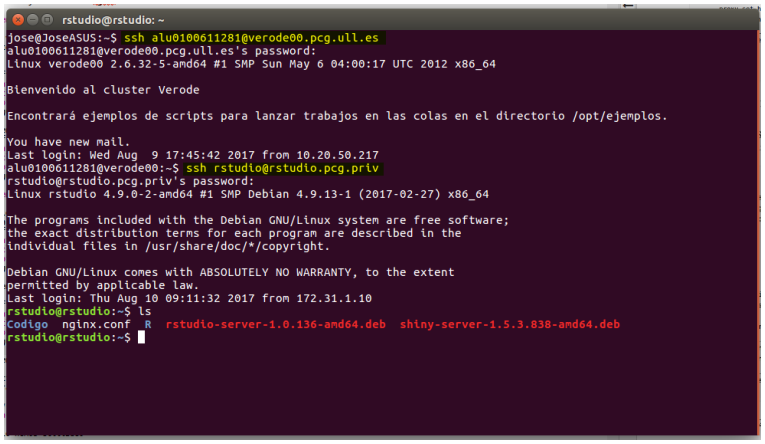


Figura 3.16: Verode

3.7.2. SSH

SSH es un estándar para los accesos remotos seguros y transferencias de archivos a través de redes no confiables. También proporciona una forma de asegurar el tráfico de datos de cualquier aplicación utilizando el reenvío de puertos.

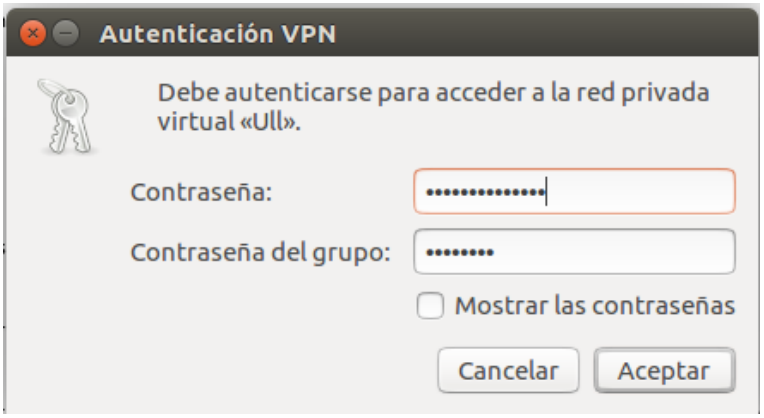


Figura 3.17: Conexión SSH

3.7.2.1. Tunel SSH

Para acceder directamente, hemos establecido una serie de túneles SHH. Los túneles SHH son un método para trasportar datos de red arbitrarios a través de conexiones SSH cifrada, también podremos acceder a servicios de internet a través de cortafuegos. El tunelado SSH permite agregar seguridad de red a aplicaciones heredadas que no admiten de forma nativa el cifrado.

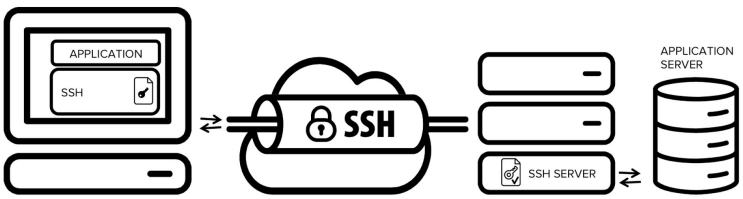


Figura 3.18: ssh túnel

3.7.2.2. Ventajas de usar Túneles SSH

Los túneles SSH son ampliamente utilizados en muchos entornos corporativos que emplean sistemas mainframe como backends de aplicaciones. En esos entornos, las propias aplicaciones pueden tener soporte nativo muy limitado para la seguridad. Mediante la utilización de túneles, el cumplimiento de SOX, HIPAA, PCI-DSS, y otros estándares se puede lograr sin tener que modificar las aplicaciones.

En muchos casos, estas aplicaciones y servidores de aplicaciones son tales que hacer cambios de código a ellos puede ser poco práctico o prohibitivos, debido a su alto costo. El código fuente puede no estar disponible, el proveedor puede haber quebrado, el producto puede estar fuera de soporte, o el equipo de desarrollo se ha ido. La adición de un contenedor de seguridad, como SSH tunneling, ha proporcionado una forma rentable y práctica de agregar seguridad para dichas aplicaciones.

3.7.2.3. Riesgos de usar Túneles SSH

Las conexiones SSH están protegidas con un cifrado fuerte. Esto hace que su contenido sea invisible para la mayoría de las soluciones de monitoreo de red y de filtración de tráfico implementadas. Esta invisibilidad conlleva un potencial de riesgo considerable si se utiliza con fines maliciosos, como la exfiltración de datos. Los ciberdelincuentes o malware podrían explotar SSH para ocultar sus comunicaciones no autorizadas o para exfiltrar datos robados de la red de destino.

3.7.2.4. Comando usados para crear los túneles:

- Conexion a Rstudio Server:

  -ssh -L8787:rstudio.pcg.priv:8787 alu0100611281@verode00.pcg.ull.es

- Conexion a Shiny Server

  -ssh -L3838:rstudio.pcg.priv:3838 alu0100611281@verode00.pcg.ull.es

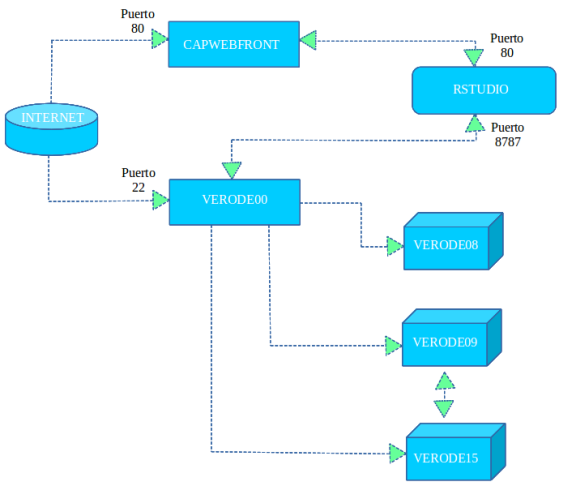


Figura 3.19: Esquema de conectividad al cluster Verode

Capítulo 4

Herramientas

En este capítulo se pretende especificar el conjunto de herramientas utilizadas a lo largo del proyecto para conseguir los objetivos planteados.

4.1. Entorno de Desarrollo

4.1.1. GitHub -Control de Versiones

El control de versiones es un sistema que permite registrar los cambios realizados sobre archivos a lo largo del tiempo, durante el Grado de Ingeniería Informática se ha trabajado con múltiples opciones para el control de versiones, en este caso se ha usado GitHub. Con el fin de que el código mantuviese la calidad y persistencia a errores se realiza un commit, tras llevar a cabo cada funcionalidad de forma correcta, de esta forma en caso de un fallo posterior se puede revertir y volver a un estado en el que la aplicación funcione correctamente. También se puede usar GitHub como nube, una vez terminado las diferentes fases del proyecto, se van subiendo a GitHub y se van etiquetando, y si queremos descargarnos el proyecto en cualquier fase, lo podemos hacer en cualquier máquina, clonando el proyecto.

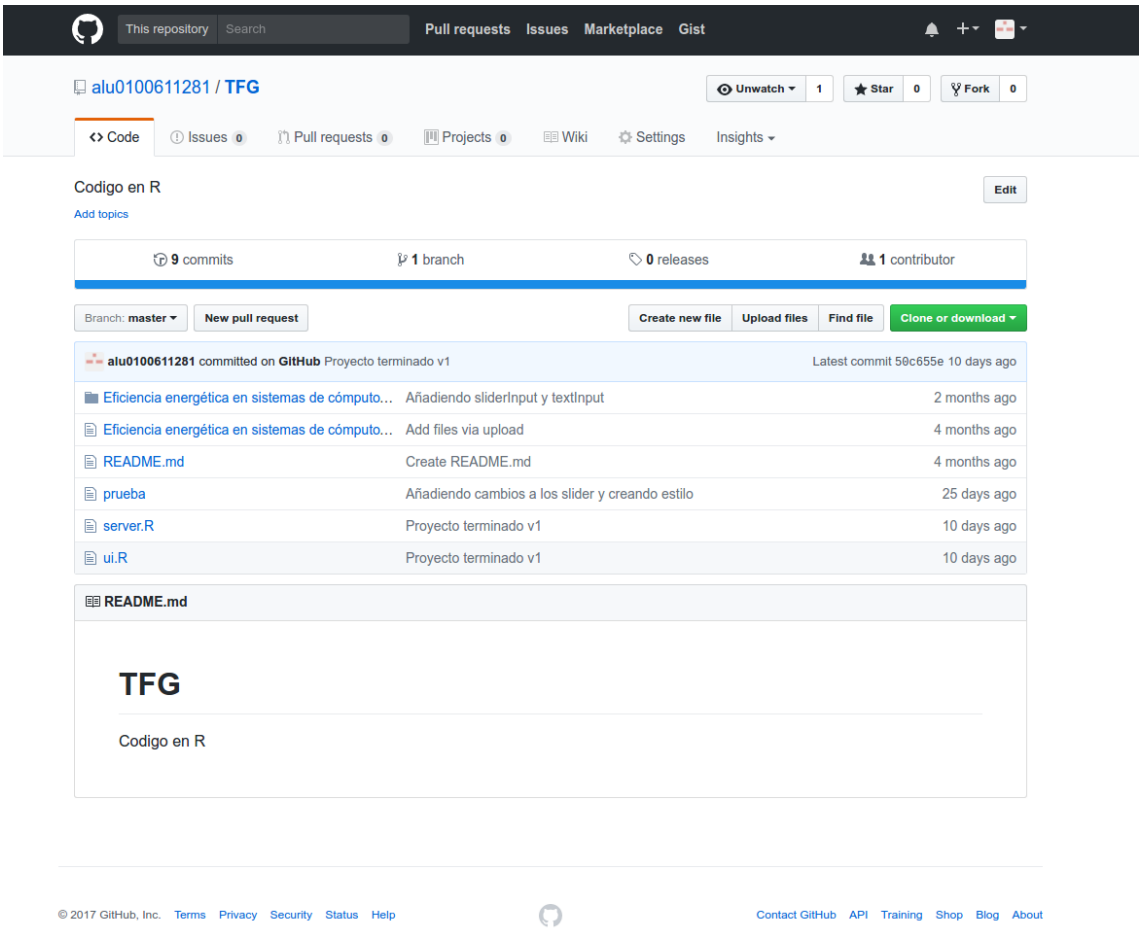


Figura 4.1: GitHub

4.2. Trello

Tal y como se mencionó previamente con el objetivo de seguir la metodología, se estudió la posibilidad del uso de herramientas que facilitasen esta tarea y se eligió Trello. Trello permite asignar un tablero para el proyecto, dentro de este tablero se crean listas con las distintas funcionalidades o sprints a realizar. Las listas a su vez están formadas por tarjetas, cada una de ellas representa una tarea del conjunto de tareas que forman un sprint. Estas tarjetas se van moviendo, según la función que se este desempeñando, la lista Pendiente, en un principio contenía todas las tarjetas que se iban a realizar en un futuro, cada vez que se disponía a realizar el trabajo de una tarjeta, esta se movía a la lista de Proceso, en esta lista se encuentran todas las tarjetas que estamos realizando en este momento, una vez acabadas se movían a la lista de terminado, también puede ocurrir que dicha tarea no se pueda realizar por diferentes motivos y por esto hay una lista de Descarte, a la cual moveremos las tarjetas que no podamos realizar.

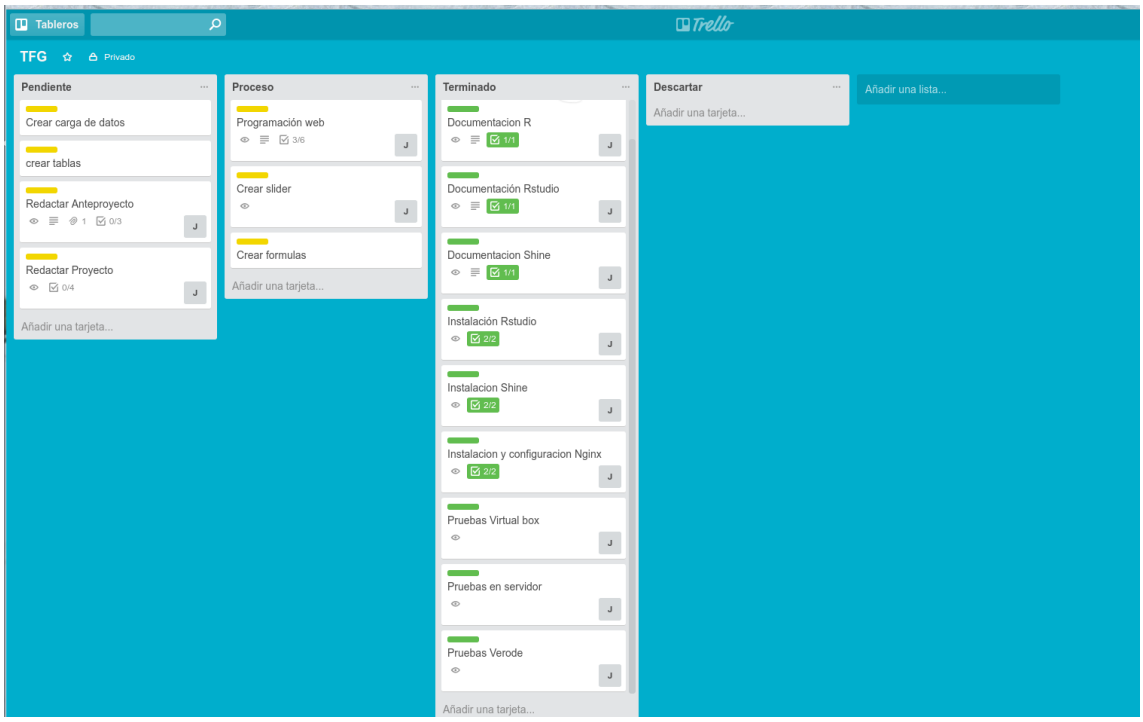


Figura 4.2: Tablero del Proyecto en Trello

4.3. Sharelatex

Para el desarrollo de la memoria, hemos usado latex y decidido utilizar la herramienta online de sharelatex. LaTeX es un editor de texto diseñado especialmente para los usuarios que tienen la necesidad de crear complejos documentos matemáticos, libros de texto u otros documentos técnicos. LaTeX utiliza su propio lenguaje, muy similar al HTML en cuanto a aspecto y estructura, y permite realizar fácilmente tareas que con otro tipo de procesadores de texto serían bastante complejas. En cuanto a sharelatex lo primero que destaca es la interfaz, mucho más fácil de usar y más cómoda a la hora de gestionar proyectos con varios archivos. Desde la misma barra lateral podemos acceder a los logs de compilación y al documento PDF, que se crea automáticamente cada poco tiempo. ShareLaTeX introduce también una característica muy interesante: colaboración entre varios usuarios. Podemos editar al mismo tiempo un documento. Este programa es totalmente gratuito y ofrece su código fuente completo a través de GitHub de manera que cualquiera podrá revisarlo y reportar posibles fallos y vulnerabilidades.

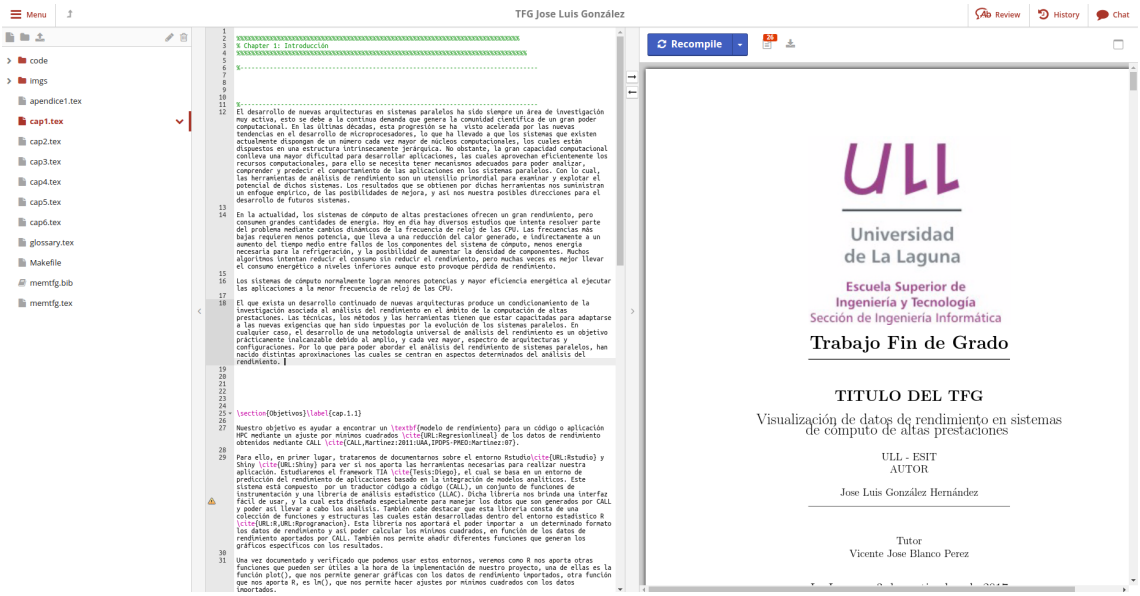


Figura 4.3: Sharelatex

Capítulo 5

Conclusiones y trabajos futuros

A lo largo de este documento hemos visto los distintos aspectos relacionados con la visualización de datos de rendimiento en sistemas de cómputo de altas prestaciones. Se han usado los conocimientos adquiridos durante la formación académica, esto incluye la investigación y análisis y el desarrollo del código. Gracias a este trabajo de fin de grado, he podido aprender un nuevo lenguaje de programación, como es R. También descubrir entornos como Rstudio y Shiny. Resumiendo este proyecto, hemos creado una aplicación que, a partir de datos de rendimiento obtenidos por el sistema de instrumentación CALL, visualiza de forma interactiva la información obtenida en la instrumentación y permite obtener un modelo de rendimiento asociado a estos datos. La aplicación dispone de controles para la selección de rangos en eje de coordenadas (manualmente o mediante slider) lo que permite obtener información y modelos específicos para una región de interés. Cabe destacar que nuestra aplicación finalmente tendrá una opción de selección de ecuaciones para poder obtener el modelo que más se ajuste a los datos. Para todo esto hemos usado el entorno Rstudio server como plataforma de desarrollo y el paquete Shiny server para alojar la aplicación ya terminada.

A continuación se proponen mejoras como **trabajo futuro**, para incluir en módulos de nuevas versiones de la aplicación:

- Una de las mejoras que se podría realizar es la integración con las herramientas de modelado disponible en TIA y que trabajan directamente con la instrumentación de CALL.

- Otra de las mejoras que se podría hacer, es ajustar la visualización de los datos, con plotly() en vez de con plot(), ya que esto nos permitiría un mayor control de nuestra gráfica, pudiendo usar el ratón para ajustar los puntos que queramos visualizar mas de cerca, guardar la gráfica en un documento, resetear la gráfica, poder desplazarnos por la gráfica con el ratón.

- Desarrollar un parser para las ecuaciones de los modelos. En vez de seleccionar la ecuación directamente de algunas predefinidas, el usuario pueda

introducir directamente la ecuación que mejor le convenga para el estudio de rendimiento.

- Se podría añadir un botón de guardar gráfica, y así se exportaría la gráfica a otro documento. Esto ya se podría hacer si actualizamos la forma de visualizar a plotly() ya que esta función aporta este apartado.

Capítulo 6

Summary, conclusions and future works

Throughout this document we have seen various aspects related with the visualization of performance data in High Performance Computing systems. The skills and knowledge obtained during my academic training allowed me to afford this Final Grade with success, specially skills in research, analysis, and code development. Thanks to this Final Grade, I have been able to learn a new programming language: R. I also discover frameworks to work with this language like Rstudio and Shiny. In this project, we have created an application to visualize performance data obtained with CALL instrumentation tool. The application allow interactive controls to select the region of interest, manually or with sliders. Performance models can be obtained for the selected data through a equations selector. We have used Rstudio server framework as a development platform and Shiny server package to deploy the applications on a web server.

For **future work** we propose the following features for new versions of the application:

- Integration with TIA modeling tools. This R software interacts easily with CALL instrumentation tool.
- Adjust the display of the data with plotly() instead of plot (). This would allow us greater control of our plots, use the mouse to adjust the point that we want to display more closely, save the graphic in a document, reset the plot, or move by the plot with the mouse.
- Improve the code's selection of equation. Instead of directly selecting the equation from some predefined, the user can directly enter model equation that best suits for the performance data.
- Add a button to save the graphics, in order to export them to other formats. This is already available with the use of plotly() package.

Apéndice A

Enlaces de Interés

A.1. Repositorio de Código del Proyecto

https://github.com/alu0100611281/TFG

A.2. Imágenes completas

https://drive.google.com/drive/folders/0B1i0LuuF4zGQTGlrczlhcVBycFE

A.3. Imágenes del código

https://drive.google.com/drive/folders/0B1i0LuuF4zGQSEpxekpuRGxVYWM

Bibliografía

[1] Barcelona supercomputing center - centro nacional de supercomputación. http://www.bsc.es.

[2] Configuración de nginx en rstudio server. https://support.rstudio.com/hc/en-us/articles/200552326-Configuring-the-Server/, https://www.digitalocean.com/community/tutorials/how-to-configure-nginx-as-a-web-server-and-reverse-proxy-for-apache-on-one-ubuntu-14-

[3] Configuración de nginx en shiny server. https://support.rstudio.com/hc/en-us/articles/213733868-Running-Shiny-Server-with-a-Proxy.

[4] Página de nginx. https://nginx.org/en/.

[5] Página donde están diversos packages. https://www.rstudio.com/products/rpackages/.

[6] Página información cluster. https://en.wikipedia.org/wiki/Computer_cluster.

[7] Página inicial rstudio. https://www.rstudio.com/products/RStudio/.

[8] Página inicial rstudio server. https://www.rstudio.com/products/RStudio/#Server.

[9] Página inicial shiny. http://shiny.rstudio.com/.

[10] Página inicial shiny server. https://www.rstudio.com/products/shiny/shiny-server/.

[11] Página túneles ssh. https://www.ssh.com/ssh/tunneling/.

[12] R development core team: R: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria, 2009. http://www.R-project.org.

[13] R language. https://www.genbetadev.com/formacion/r-un-lenguaje-y-entorno-de-programacion-para-analisis-estadistico, https://www.cs.us.es/~fran/curso_unia/introduccion_R.html,http://blog.datatons.com/2016/04/08/que-es-lenguaje-programacion-r/,

https://bbvaopen4u.com/es/actualidad/
taller-para-novatos-en-r-ventajas-instalacion-y-paquetes.

[14] Regresión lineal o mínimos cuadrados.
https://ww2.coastal.edu/kingw/statistics/R-tutorials/
simplelinear.html,
https://www.uam.es/personal_pdi/ciencias/joser/paginaR/
regresion.html,
https://web.ua.es/es/lpa/docencia/analisis-estadistico-de-datos-geoquimicos-con-r/
regresion-lineal-simple-y-multiple-regresion-no-lineal.html,
http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/
regresion1/regresion1.htm,
https://www.varsitytutors.com/hotmath/hotmath_help/spanish/
topics/line-of-best-fit.

[15] Tau performance system. http://www.cs.uoregon.edu/research/tau/
home.php.

[16] Trace analyzer and collector. http://software.intel.com/en-us/
articles/intel-trace-analyzer.

[17] Vampir. http://www.vampir.eu.

[18] Vampir: product history. http://www.vampir.eu/Historie.html.

[19] Alberto Cabrera, F. Almeida, and Vicente Blanco. EML, an Energy Measurement Library. In Karin Anna Hummel, editor, *31st International Symposium on Computer Performance, Modeling, Measurements and Evaluation 2013: Student Poster Abstracts*, Vienna, Austria, September 24–26, 2013. Forschungsgruppe Entertainment Computing, Fakultät für Informatik, University of Vienna.

[20] Marcus Hähnel, Björn Döbel, Marcus Völp, and Herman Härtig. Measuring energy consumption for short code paths using rapl. In *GreenMetrics 2012 Workshop, in conjuction with ACM Sigmetrics/Performance 2012*, ACM Sigmetrics, London, UK, June 11, 2012. ACM.

[21] Vincent Pillet Toni Cortés y Luis Gregoris Jesús Labarta, Sergi Girona. *A Parallel Program Development Environment.* En 2 nd International Euro-Par Conference, EuroPar, 1996.

[22] Diego Rodríguez Martínez. *Modelado analítico do rendemento de aplicacións en sistemas paralelos.* PhD thesis, Universidade de Santiago de Compostela, July 2011.

[23] D.R. Martínez, J.L. Albín, T.F. Pena, J.C. Cabaleiro, F.F. Rivera, and V. Blanco. Using accurate AIC–based performance models to improve the scheduling of parallel applications. *Journal of SuperComputing*, 58(3):332–340, December 2011. In press: Online http://dx.doi.org/doi:10.1007/s11227-011-0589-1.

[24] D.R. Martínez, V. Blanco, M. Boullón, J.C. Cabaleiro, C. Rodríguez, and F.F. Rivera. Software tools for performance modeling of parallel programs. In *21 st IEEE International Parallel and Distributed Processing*

Symposium (IPDPS 2007). 6th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2007), volume 07TH8938, Long Beach, California. USA, April 2007. IEEE Computer Society.

[25] C. León C. Rodríguez G. Rodríguez y M. Printista V. Blanco, J. A. González. *Predicting the performance of parallel programs.* Parallel Computing, 30:337–356,2004.

[26] M. Weber H. Ch. Hoppe y K. Solchenbach W. E. Nagel, A. Arnold. *VAMPIR:Visualization and Analysis of MPI Resources.* Supercomputer, 12:69–80, 1996.