Universidad Zaragoza
1542

Trabajo Fin de Grado

Optimización de una red de aprovisionamiento

Optimization of a supply network

Autor/es

Alejandro Sarabia Hernando

Director/es

Luis Mariano Esteban Escaño

Escuela Universitaria Politécnica La Almunia

2016

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Optimización de una red de aprovisionamiento

Optimization of a supply network

425.16.108

Autor: **Alejandro Sarabia Hernando**
Director: **Luis Mariano Esteban Escaño**
Fecha: **29 de junio de 2016**

ÍNDICE DE CONTENIDO SIMPLE

1.RESUMEN **1**
2.ABSTRACT **2**
3.INTRODUCCIÓN **3**
4.OBJETIVOS **4**
5.ORÍGENES **5**
6.CARACTERÍSTICAS DEL TSP **10**
7.HERRAMIENTAS **12**
8.ANÁLISIS DE LA RED DE APROVISIONAMIENTO **20**
9.OBTENCIÓN DE LA SOLUCIÓN **39**
10.CREACIÓN DE LA APLICACIÓN **58**
11.INTERFAZ DE LA APLICACIÓN **72**
12.CONCLUSIONES **78**
13.BIBLIOGRAFÍA **79**

ÍNDICE DE CONTENIDO COMPLETO

1.RESUMEN **1**
2.ABSTRACT **2**
3.INTRODUCCIÓN **3**
4.OBJETIVOS **4**
5.ORÍGENES **5**
6.CARACTERÍSTICAS DEL TSP **10**
6.1.NP-Hard 10
6.2.Algoritmo voraz 10
6.3.Principales variantes 11
7.HERRAMIENTAS **12**
7.1.R-Project 12

7.2.RSTUDIO....13
7.3.PAQUETES DE R NECESARIOS....14
7.3.1.lp_Solve....14
7.3.2.TSP....15
7.3.3.rJava....16
7.3.4.xlsx....17
7.3.5.shiny....18
7.3.6.igraph....19
7.4.OTROS SOFTWARES: MICRROSOFT OFFICE....19
8.ANÁLISIS DE LA RED DE APROVISIONAMIENTO....20
8.1.CAPITALES AUTONÓMICAS Y PROVINCIALES....20
8.2.MAPA POLÍTICO DE ESPAÑA....23
8.3.RED DE ADYACENCIA....24
9.OBTENCIÓN DE LA SOLUCIÓN....39
9.1.GRAFO....40
9.2.MATRIZ PRINCIPAL....42
9.3.CÁLCULO DE LA SOLUCIÓN....56
9.3.1.lp....56
9.3.2.solve_TSP....57
10.CREACIÓN DE LA APLICACIÓN....58
10.1.ARCHIVOS GENERADORES DE LA APLICACIÓN....58
10.1.1.Archivo server.R....59
10.1.2.Archivo ui.R....67
11.INTERFAZ DE LA APLICACIÓN....72
11.1.INPUTS....72
11.1.1.Introducir archivos de precios....72
11.1.2.Coste transporte por kilometro....73
11.1.3.Origen Ruta....73
11.1.4.Algoritmo....73
11.2.OUTPUTS....74
11.2.1.Introducción....74
11.2.2.Matriz de costes....75
11.2.3.Ruta de compra....76
11.2.4.Información complementaria....77
12.CONCLUSIONES....78

13.BIBLIOGRAFÍA.....79

ÍNDICE DE ILUSTRACIONES

Ilustración 1: R-project.....12

Ilustración 2: Descarga R-Project.....13

Ilustración 3: RStudio.....13

Ilustración 4: Descarga RStudio.....14

Ilustración 5: lp_Solve.....14

Ilustración 6: TSP.....15

Ilustración 7: rJava.....16

Ilustración 8: xlsx.....17

Ilustración 9: shiny.....18

Ilustración 10: igraph.....19

Ilustración 11: Microsoft Office.....19

Ilustración 12: Mapa de España.....23

Ilustración 13: Red de adyacencia.....38

Ilustración 14: Grafo R-project.....39

Ilustración 15: attributes.....39

Ilustración 16: graph.adjacency.....40

Ilustración 17: Vertices.....40

Ilustración 18: Conexiones.....41

Ilustración 19: Grafo.....41

Ilustración 20: Archivos RStudio.....58

Ilustración 21: Paquetes server.R.....59

Ilustración 22: shinyServer.....59

Ilustración 23: Matriz Distancias.....60

Ilustración 24: Nombres Matriz Distancias 1.....60

Ilustración 25: Nombres Matriz Distancias 2.....60

Ilustración 26: Dataset.....61

Ilustración 27: outputRuta.....61

Ilustración 28: Matrizkm.....61

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Ilustración 29: Matriz precios.....61

Ilustración 30: Matriz MM.....62

Ilustración 31: objective.in.....62

Ilustración 32: const.mat.....62

Ilustración 33: sol lp.....63

Ilustración 34: Ciudadesruta.....63

Ilustración 35: Duplicidad Ciudad Origen.....63

Ilustración 36: Ciudad Origen unica en la ruta.....63

Ilustración 37: Objetos Ciudad Origen y Algoritmo.....64

Ilustración 38: dummy city.....64

Ilustración 39: Resultado nombres de las ciudades.....64

Ilustración 40: outputdistancia.....65

Ilustración 41: Longitud de la ruta igual a cero.....65

Ilustración 42: tour_length.....65

Ilustración 43: outputCiudadesProducto.....65

Ilustración 44: Ciudades cada producto.....66

Ilustración 45: outputDatosCostes.....66

Ilustración 46: outputCiudadesConProducto.....66

Ilustración 47: outputRutaDeCompra.....66

Ilustración 48: outputLongitudDeLaRuta.....67

Ilustración 49: Paquetes ui.R.....67

Ilustración 50: shinyUI.....67

Ilustración 51: Título de la aplicación.....67

Ilustración 52: Panel con inputs.....68

Ilustración 53: input archivo de precios 68

Ilustración 54: input coste kilometro.....68

Ilustración 55: input origen ruta.....68

Ilustración 56: input algoritmo.....69

Ilustración 57: width.....69

Ilustración 58: Panel con outputs.....69

Ilustración 59: output introducción.....70

Ilustración 60: output matriz de costes.....70

Ilustración 61: input ruta de compra.....70

Ilustración 62: Logotipo app.....72

Ilustración 63: Introducir archivo de precios.....72

Ilustración 64: Coste transporte por kilometro.....73

Ilustración 65: Origen Ruta.....73

Ilustración 66: Algoritmo.....74

Ilustración 67: Zona inputs.....74

Ilustración 68: Introducción.....75

Ilustración 69: Matriz de costes.....75

Ilustración 70: Ruta de compra.....76

Ilustración 71: Mapa de España.....77

ÍNDICE DE TABLAS

Tabla 1: Ciudades.....22

Tabla 2: Matriz de adyacencia 25

Tabla 3: Matriz de adyacencia 27

Tabla 4: Matriz de adyacencia.....29

Tabla 5: Matriz de adyacencia.....31

Tabla 6: Matriz de adyacencia.....33

Tabla 7: Matriz de adyacencia.....35

Tabla 8: Matriz de adyacencia.....37

Tabla 9: Matriz principal.....43

Tabla 10: Matriz principal.....45

Tabla 11: Matriz principal.....47

Tabla 12: Matriz principal.....49

Tabla 13: Matriz principal.....51

Tabla 14: Matriz principal.....53

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Tabla 15: Matriz principal.....55

Tabla 16: Funciones tamaño y formato de texto.....71

1. RESUMEN

En este Trabajo Fin de Grado se crea una aplicación que permite al usuario seleccionar una serie de campos (Lista de precios de productos, Coste por kilometro, Almacén de origen y Algoritmo) de forma sencilla para conocer la ruta más económica que posibilite la compra de los productos que éste necesita.

Se plantea el problema del viajante comprador (TPP, Travelling purchaser problem) que es una generalización más compleja del problema del viajante (TSP, Travelling salesman problem).

Se presentan los orígenes y características del Travelling Salesman Problem.

Se describen las herramientas (programas y paquetes) necesarias para la realización de la aplicación.

Se analiza la red de aprovisionamiento que en este caso son todas las capitales de Provincia y Comunidad Autónoma del territorio peninsular español.

Se explica como se obtiene la solución del problema.

Se detalla paso a paso el código realizado para conseguir que la aplicacíon funcione.

Por último se muestra la interfaz de la aplicación y se describe cada uno de los campos.

2. ABSTRACT

In this Degree Final Project a web application was built, this tool allows the user to select some fields simply (Product price list, cost per kilometer, warehouse of beginning and algorithm) to know the most economical route that enables the purchase of the products that the user needs.

The app was based on the Travelling purchaser problem (TPP), that is a more complex generalization of the travelling salesman problem (TSP). The origins and features of the Travelling Salesman Problem are presented .

In addition, the tools (programs and packages) that are necessary to realization for application are described. Also, the supply network is analyzed, that in this case are the provincial capitals and the capitals of the autonomous community of the peninsular spanish territory. Moreover, It includes a detailed explanation of how the solution of the problem is obtained. The code done to get the application works is provided.

Finally the interface of the application is shown and each of the fields are described.

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

3. INTRODUCCIÓN

La logística, tanto en la compra como venta de productos, es muy importante para cualquier empresa debido a que supone costes elevados.

Aunque parezca no tener importancia, es muy necesario invertir el tiempo que se necesite en mejorar las redes de transporte porque supone una reducción significativa de los costes y una disminución de los tiempos, lo que a la larga tendrá un impacto positivo sobre el cliente que es lo realmente importante.

Por ello el diseñar una red de transporte lo más eficiente posible es tan importante como que la calidad de los productos sea excelente.

4. OBJETIVOS

La motivación y objetivo final del TFG es la creación de una app que permita al usuario seleccionar una serie de campos (Lista de precios de productos, Coste por kilometro , Almacén de origen y Algoritmo) de forma sencilla para conocer la ruta más económica que posibilite la compra de los productos que éste necesita.

Se pretende seleccionar aquellas ciudades en las cuales el coste, formado por el coste del viaje y el coste del producto, sea el menor y una vez elegidos generar la ruta óptima que permita recorrer dichas ciudades. Hay que mencionar que en las diferentes ciudades los productos tienen precios diferentes o puede ocurrir que algún producto no se pueda adquirir en todas ellas.

Actualmente se desconoce la existencia de apps que presenten lo que va a ofrecer la aplicación que se va a desarrollar en este trabajo.

Se quiere poner en práctica los conocimientos adquiridos en las asignaturas de Investigación Operativa y Ampliación de Investigación Operativa, aunque se trata de un problema que está asociado a otras asignaturas, ya que por ejemplo la logística o minimización de costes son básicas en este estudio.

En este trabajo se plantea el problema del viajante comprador (TPP, Travelling purchaser problem) que es una generalización más compleja del problema del viajante (TSP, Travelling salesman problem). Mientras que el problema del viajante tiene como objetivo hallar el itinerario más corto que pasa por cada ciudad (una vez) de una lista determinada y luego vuelve a la ciudad de la que se ha partido, el problema del viajante comprador se encarga de encontrar el recorrido, entre un subconjunto de ciudades, que permita la compra de la totalidad de los productos requeridos minimizando el coste total integrado por los costes del viaje propiamente dicho y los costes de la compra de los productos.

Enunciado del problema del viajante:

Un viajante tiene que visitar una lista de ciudades, una sola vez, saliendo de una de ellas y volviendo a esta. ¿Cuál es la ruta que minimiza la distancia a recorrer?

Es un problema de optimización combinatoria, muy importante en la investigación de operaciones y en la computación cuya resolución optima es compleja a pesar de ser definido fácilmente.

5. ORÍGENES

El origen del TSP se remonta a un problema anterior que se origina en la teoría de grafos.

Se conoce como grafo al par:

G=(V,E) ————›V – Conjunto finito de elementos (vértices).

E – Conjunto de pares de vertices (ramas)

Un ciclo es la sucesión de vértices V1, V2,…, V_p tales que V1, V2,…, V_{p-1} son diferentes, $V_p = V_1$ y $(V_i, V_{i+1}) \in E$. Si el ciclo comprende todos los vértices se denomina hamiltoniano (en honor al matemático irlandés Sir William Rowan Hamilton).

El TSP para un grafo en el cual cada rama tiene un peso asignado, es encontrar un ciclo hamiltoniano de mínimo peso entendiéndose como peso de ciclo a la suma de los pesos de todas las ramas que lo recorre.

Inversamente, el problema de decidir si un grafo tiene un ciclo hamiltoniano es un caso especial del TSP (si se le asigna a todas las ramas del grafo peso 0 y a las ramas faltantes se les agrega asignándoles peso 1, se tendrá otro grafo que tiene ciclos hamiltonianos. Resolviendo el TSP, el nuevo grafo tiene un ciclo hamiltoniano de mínimo peso = 0 sólo si el grafo original contiene un ciclo hamiltoniano).

Anterior al análisis de Hamilton, Euler y Vandermonde discutieron el problema del tour del caballo que se trata de encontrar un ciclo hamiltoniano para el grafo cuyos vértices representan los 64 cuadrados del ajedrez, con dos vértices adyacentes sólo si un caballo puede moverse en un paso de un cuadrado de otro.

El reverendo T. P. Kirkman fue el primero en considerar ciclos hamiltonianos en un contexto general. Dio una condición suficiente para que un grafo poliédrico admita un tal ciclo y además mostró que un poliedro con un número impar de vértices en donde cada cara tiene un número par de aristas, no tiene dichos ciclos.

Al mismo tiempo, Hamilton inventaba un sistema de álgebra no conmutativa. Lo llamó cálculo icosaédrico (vértices adyacentes del dodecaedro se corresponden con caras adyacentes del icosaedro). Usó la interpretación gráfica como la base para un juego llamado el juego del icosaedro, éste consistía de varios problemas, como por ejemplo terminar de encontrar un ciclo teniendo prefijadas las 5 primeras posiciones.

Un precursor más directo del TSP, en donde el largo de las ramas jugaba un rol

predominante fue, una nueva definición de la medida de una curva que propuso Menger. Él la definió como el supremo del conjunto formado por todos los números que pueden ser obtenidos de tomar cada conjunto finito de puntos sobre la curva y determinar la medida de la menor poligonal que los une. A este problema se lo llamó el del mensajero. La resolución no requiere un ciclo, sólo un camino que contenga todos los vértices.

En 1832 en el último capítulo del libro "El problema del viajero, cómo debe hacer para obtener éxito en sus negocios" se vislumbra la esencia del TSP cuando se comenta que con una elección apropiada del tour, se puede ganar mucho tiempo y que el aspecto más importante es cubrir tantas ciudades como sea posible sin visitar una de ella dos veces.

Merrill Flood fue el responsable de divulgar el nombre de TSP. Le habló acerca de él a A.W Tucker en 1937. Tucker le comentó que recordaba haberlo escuchado de boca de Hassler Whitney de la Universidad de Princeton pero no podía confirmar con certeza esta historia. De ser verídica asegura que ocurrió en los años 1931-1932 pues fue en ese entonces cuando se hallaba terminando su tesis con Lefschetz. Whitney era un compañero que se encontraba en su etapa posdoctoral y estaba trabajando en teoría de grafos, especialmente en planaridad y en el problema de los cuatro colores y Flood era un estudiante recién graduado. El problema del viajante ya tenía un nombre.

John Williams incitó a Flood en 1948 a popularizar el TSP en la corporación RAND, motivado por el propósito de crear talentos intelectuales para modelos fuera de la teoría de juegos. No hay dudas que la reputación y la autoridad de RAND, que rápidamente se convirtió en el centro intelectual de muchas de las investigaciones sobre esta teoría, amplificó la publicidad de Flood. Otra razón de la popularidad del problema fue su íntima conexión con el problema de la asignación y del transporte.

La aparición del articulo "Soluciones de un problema del viajante de gran tamaño" de Dantzig, Fulkerson y Johnson en el Journal of the Operations Research Society of América fue uno de los principales eventos en la historia de la optimización. Para entender su importancia necesitamos conocer el estado en que se encontraba la optimización combinatoria en el momento en que el artículo apareció y dos problemas en particular de la programación lineal: el problema del transporte y el de asignación.

El problema de la asignación es elegir n elementos, uno por cada fila y columna de una matriz C = (c_{ij}) de n x n tales que la suma de los elementos elegidos sea la menor posible. Hay n! maneras posibles de hacer la elección, por lo tanto, un algoritmo efectivo debería hacer algo diferente que considerar todas las posibilidades.

Una alternativa dentro de la programación lineal es considerar el poliedro P en el espacio n^2 definido como el conjunto de todas las matrices $X = (x_{ij})$ que satisface las condiciones $x_{ij} \geq 0$, $\sum_j x_{ij} = 1 \forall_i$, $\sum_i x_{ij} = 1 \forall_j$ y minimizar $\sum c_{ij}x_{ij}$.

De acuerdo a Birkhoff, los vértices de P son precisamente todas las matrices X en donde cada fila y columna contiene exactamente un 1 y todas las otras entradas son 0. Por lo tanto, los n! vértices de P se corresponden con las n! elecciones posibles y la ∑cijxij calcula el valor de cada elección. Como el óptimo de esta función se alcanza en un vértice, se pueden utilizar los algoritmos de la programación lineal.

El problema del transporte es más general que el problema de la asignación. Es el problema de elegir una matriz $X = (x_{ij})$ de m × n que satisfaga $x_{ij} \geq 0$, $\sum_j x_{ij} = a_i \forall_i$, $\sum_i x_{ij} = b_j \forall_j$ que minimiza $\sum c_{ij}x_{ij}$ donde a_i y b_j son enteros no negativos que cumplen $\sum_i a_i = \sum_j b_j$. El problema del transporte modela la siguiente situación: se tienen m recursos i(i = 1,..., m), de cada uno de los cuales hay una cierta cantidad a_i, que se quieren enviar a n destinos j(j = 1,..., n) sabiendo que cada uno ha pedido una cierta cantidad de mercadería b_j. Si cuesta exactamente c_{ij} enviar una unidad de mercadería i al destino j, ¿Cómo podría organizarse la entrega para cumplir con todos los requerimientos y minimizar el coste total?

Dantzig desarrolló el método simplex para resolver problemas de programación lineal. En 1953, existían códigos de implementación efectivos del método simplex en general y adaptaciones especiales para los casos del problema del transporte y de la asignación.

En 1954 Dantzig, Fulkerson y Johnson hicieron un método que resolvía el problema del horario de los buques. La solución llegó varios años después de que el modelo se volviera obsoleto, pero logró sobrevivir porque el método podía ser usado para estudiar las preguntas básicas de la teoría combinatoria. Ford y Fulkerson escribieron su primer reporte sobre sistemas de flujo en 1956, iniciando de este modo un tópico mayor del cual derivó un resultado de Johnson sobre la secuencia de trabajos. El TSP, sin embargo, no estaba relacionado a simple vista con estos desarrollos pero había esperanza de que lo estuviera.

Volviendo al problema de asignación descrito arriba, si c_{ij} es la distancia de la ciudad i a la ciudad j, entonces el TSP guarda cierta similitud con el problema de la asignación. Podemos interpretar que $x_{ij} = 1$ significa que el viajero se mueve de la ciudad i a la j en su tour. Una solución al problema de la asignación, bajo esta interpretación, puede ser un conjunto de subtours disconexos en donde cada ciudad es visitada exactamente una vez, con lo cual no resolvería el TSP. Se debería imponer

la condición adicional de no permitir subtours, que se puede expresar matemáticamente mediante 2^{n-1} inecuaciones. Desafortunadamente, el conjunto de vértices del nuevo poliedro Q, a diferencia del conjunto de vértices del poliedro original P, contiene matrices con entradas distintas de 0 y 1. Estos cambios producen dificultades. La primera es que en vez de 2^n ecuaciones de variables no negativas, tenemos un enorme número de inecuaciones extras a considerar. Una dificultad aún mayor es el requerimiento que las variables sean 0 o 1, que es una condición de la programación lineal entera, un tema aún no estudiado en 1950.

Consideramos el conjunto de todas las matrices X que satisfacen los requerimientos del TSP (matrices que en sus entradas solo tienen ceros y unos y que describen tours) y supongamos que conocemos, además de las inecuaciones que excluyen los subtours, todas las otras inecuaciones necesarias para crear un nuevo poliedro R cuyos vértices son precisamente los tours, así, en principio, se puede aplicar una programación lineal. Dantzig, Fulkerson y Johnson especulaban que, empezando de un tour óptimo o tal vez cercano al óptimo, era posible probar optimalidad utilizando pocas ecuaciones adicionales (llamadas cortes). El método parecía funcionar en pequeños problemas, por eso, pasaron a trabajar sobre uno de 49 ciudades. Dantzig, Fulkerson y Johnson sugirieron la posibilidad de que fuesen necesarios un gran número de cortes. Dantzig, optimista, le aposto Fulkerson que el número de cortes necesarios eran a lo sumo 25, en cambio Fulkerson más pesimista opinaba que se necesitaban al menos 26. Las predicciones fueron bastante acertadas: la cantidad correcta resultó ser 26 pero en el artículo que se publicó se decía que sólo 25 eran necesarios.

Dantzig, Fulkerson y Johnson no solo resolvieron un TSP de tamaño considerable sino que también demostraron que la complejidad de la estructura de un problema de optimización combinatoria no era un obstáculo insuperable para resolverlo. Ellos utilizaron por primera vez el concepto de branch y bound, que es un método computacional muy popular, en particular, cuando se requiere que sólo alguna de las variables sea entera. A grandes rasgos se trata de tomar una variable x que debe ser entera pero cuyo valor no lo es y a través del branching se consideran dos casos: x es al menos el mayor valor entero más cercano o x es a lo sumo el menor valor entero más cercano.

Esta construcción genera un árbol de búsqueda con nodos correspondientes a programas lineales con varias restricciones en donde no es necesario crecer pasado los nodos donde el bound sobre el valor de la solución indica que el árbol no necesita

ser explorado después de esos nodos. Todas estas ideas fueron ingredientes indispensables en la solución de la mayoría de los problemas de optimización combinatoria que surgen en la programación lineal entera.

Es claro que la publicación del artículo en cuestión fue un verdadero logro, aun cuando los autores se rehusaron a afirmar el desarrollo de un algoritmo general. Es de remarcar, haciendo una lectura en retrospectiva, cuánto de la filosofía de los últimos avances de la optimización combinatoria fueron imaginados en el exitoso abordaje de una instancia del TSP.

Otro método que se ha aplicado en la resolución del TSP fue la programación dinámica pero debido a la enorme cantidad de condiciones que incluye puede resolver instancias de problemas relativamente pequeños.(«Stockdale_Lorena.pdf», s. f.)

6. CARACTERÍSTICAS DEL TSP

6.1. NP-HARD

El problema del viajero, dentro de los problemas de optimización combinatoria, pertenece al tipo NP-Hard (NP-Difícil). Esto significa que es un problema cuya solución no puede ser verificada en tiempo real ya que su complejidad no es polinómica.

El hecho de que sea un problema NP-Hard hace que para dimensiones altas encontrar la solución óptima no sea posible sino que se intenta aproximar a las soluciones más próximas a esta solución óptima. En definitiva cuantas más ciudades comprende el problema menos exacta es la solución.(«Problema del agente viajero», s. f.)

6.2. ALGORITMO VORAZ

La solución del TSP puede seguir un algoritmo voraz, que trata de elegir la opción óptima en cada paso local, con la esperanza de llegar a una solución general óptima.

Por ejemplo estando en la ciudad "A" dirigirse a la más cercana que aun no hayamos visitado. Una vez allí, repetir el procedimiento una y otra vez hasta que se haya complementado el recorrido. Seguramente no se obtendrá un resultado óptimo, pero puede encontrarse un camino bastante bueno en un tiempo despreciable.

El término "voraz" se debe a que, en cada paso, el algoritmo escoge el mejor pedazo que es capaz de comer en ese momento sin preocuparse de los pasos que restan hasta encontrar la solución. Un algoritmo de este tipo nunca deshace una decisión ya tomada, una vez incorporado un candidato a la solución (ir de la ciudad "A" a la "B", por ejemplo) formará parte de la solución y cada candidato rechazado es eliminado definitivamente.

Este tipo de algoritmo es el que menos dificultades presentan a los investigadores que deben diseñar y comprobar el funcionamiento de diferentes estrategias.(Palazzesi, 2010)

6.3. PRINCIPALES VARIANTES

- Simétrico o STSP[2], si $d_{ij} = d_{ji} \; \forall \; (i, j) \in E$.
- Asimétrico o ATSP[3], si $d_{ij} \neq d_{ji}$ para por lo menos una $(i, j) \in E$.
- Problema del viajante triangular o ΔTSP4, si $d_{ik} \leq d_{ij}+d_{jk} \; \forall i, j, k$, pues satisface la desigualdad que lleva el mismo nombre.

> La diferencia entre el caso simétrico y el asimétrico es que en el primero la distancia entre dos ciudades es la misma en ambos sentidos mientras que en el caso del segundo las distancias no son iguales para todos los pares de ciudades.(«Problema del viajante», 2016)

7. HERRAMIENTAS

7.1. R-PROJECT

La principal herramienta utilizada para llevar a cabo el trabajo es el software R-Project for Statistical Computing, concretamente la versión empleada es la 3.2.4.

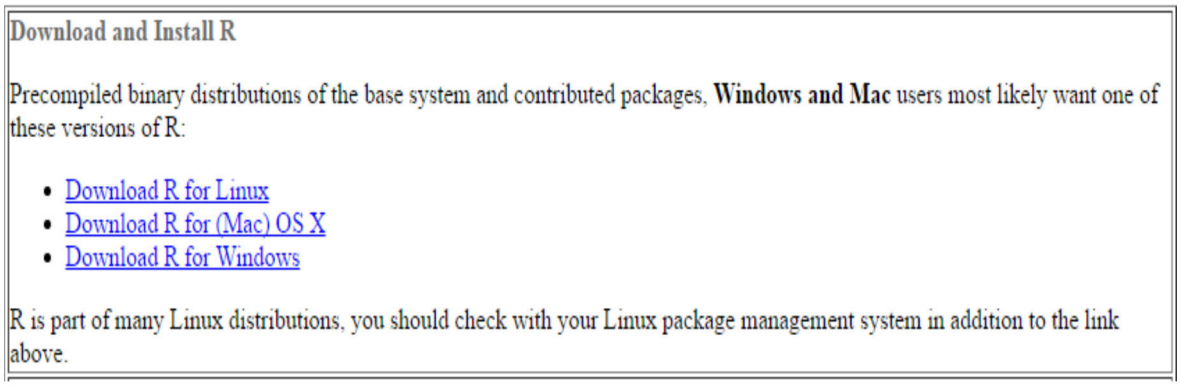
Ilustración 1: R-project

R es un software libre que permite la manipulación de datos, la realización de cálculos y la obtención de gráficos con enfoque al análisis estadístico. Algunas de las características que tiene son:

- Conjunto de operadores para cálculo, en particular para matrices.
- Almacenamiento y manejo eficaz de datos.
- Colección de herramientas coherente, amplia e integrada.
- Un simple y eficaz lenguaje de programación bien desarrollado que incluye condicionales, bucles, funciones recursivas y posibilidad de entradas y salidas
- Orientado a objetos.
- Posibilidades gráficas.

Este programa está constantemente en evolución debido a que es un proyecto colaborativo y abierto donde los usuarios pueden publicar paquetes que complementan su configuración básica. Actualmente en el listado oficial existen unos 7234 paquetes estadísticos, cada uno realizando una tarea específica dentro del mundo de la estadística.(«R (lenguaje de programación)», 2016)

Su descarga e instalación es muy sencilla, el CRAN del programa se puede obtener de la página web oficial de R, *https://www.r-project.org/*, donde se selecciona el país que se desea y da la opción de elegir para qué sistema operativo se quiere el programa.


Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Ilustración 2: Descarga R-Project

7.2. RSTUDIO

RStudio es un entorno de desarrollo integrado (IDE) para R que tiene la misión de proporcionar el entorno informático estadístico de R.

Ilustración 3: RStudio

Al igual que R es un software libre y se ejecuta en el escritorio (Windows, Mac y Linux) o en un navegador conectado a RStudio Server. Algunas de sus características son:

- Resaltado de sintaxis, auto completado de código y sangría inteligente.
- Ejecutar código R directamente desde el editor de código fuente.
- Salto rápido a las funciones.
- Depurador interactivo para diagnosticar y corregir los errores rápidamente.
- Documentación y soporte integrado.

Incluye una consola, editor de resaltado de sintaxis que soporta la ejecución de código directo, así como herramientas para el trazado, la historia, la depuración y la gestión del espacio de trabajo.

El objetivo de RStudio es ser lo más sencillo e intuitivo posible para proporcionar un entorno amigable para el usuario.(«RStudio», 2015)

Su descarga e instalación, al igual que R, es muy fácil. El programa se puede descargar de la web oficial, *https://www.rstudio.com/*, en el apartado de productos.

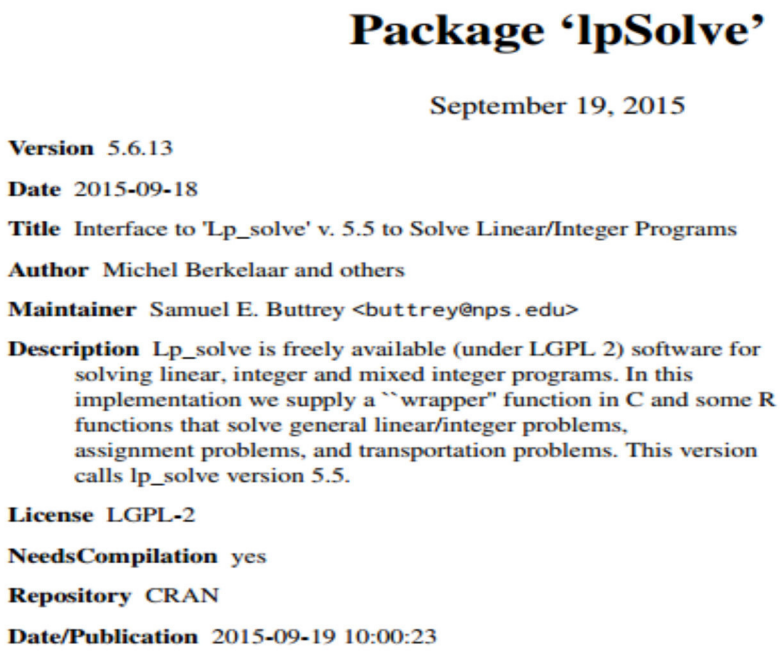
Ilustración 4: Descarga RStudio

Es importante destacar que para el uso de RStudio es necesario tener la versión 2.11.1 o superior de R.

7.3. PAQUETES DE R NECESARIOS

7.3.1. lp_Solve

lp_Solve permite resolver problemas de programación lineal, entera y mixta. Está basado en el método simplex que busca el máximo de una función lineal sobre un conjunto de variables que satisfaga un conjunto de inecuaciones lineales. Posibilita resolver problemas de asignación y de transporte.(«R Package Snapshot», s. f.-a)



Package 'lpSolve'

September 19, 2015

Version 5.6.13

Date 2015-09-18

Title Interface to 'Lp_solve' v. 5.5 to Solve Linear/Integer Programs

Author Michel Berkelaar and others

Maintainer Samuel E. Buttrey <buttrey@nps.edu>

Description Lp_solve is freely available (under LGPL 2) software for solving linear, integer and mixed integer programs. In this implementation we supply a ``wrapper'' function in C and some R functions that solve general linear/integer problems, assignment problems, and transportation problems. This version calls lp_solve version 5.5.

License LGPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-09-19 10:00:23

Ilustración 5: lp_Solve

7.3.2. TSP

El paquete TSP contiene algoritmos que permite minimizar la distancia recorrida de una ruta pasando una vez por cada una de las ciudades requeridas y volviendo a la ciudad de origen.(«R Package Snapshot», s. f.-b)

Package 'TSP'

February 22, 2016

Type Package

Title Traveling Salesperson Problem (TSP)

Version 1.1-4

Date 2016-2-21

Description Basic infrastructure and some algorithms for the traveling salesperson problem (also traveling salesman problem; TSP). The package provides some simple algorithms and an interface to the Concorde TSP solver and its implementation of the Chained-Lin-Kernighan heuristic. Concorde itself is not included in the package and has to be obtained separately from http://www.math.uwaterloo.ca/tsp/concorde.html.

Classification/ACM G.1.6, G.2.1, G.4

URL http://lyle.smu.edu/IDA/seriation

BugReports https://github.com/mhahsler/TSP/issues

Depends R (>= 2.14.0)

Imports graphics, foreach, utils, methods, stats, grDevices

Suggests maps, sp, maptools, testthat

License GPL-3

Copyright All code is Copyright (C) Michael Hahsler and Kurt Hornik.

NeedsCompilation yes

Author Michael Hahsler [aut, cre, cph], Kurt Hornik [aut, cph]

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Repository CRAN

Date/Publication 2016-02-22 08:04:07

Ilustración 6: TSP

7.3.3. rJava

Es una sencilla interfaz de R a Java. rJava ofrece un puente de bajo nivel entre R y Java. Permite crear objetos, métodos de llamadas y acceder a campos de objetos Java de R. («R Package Snapshot», s. f.-c)

Package 'rJava'

January 7, 2016

Version 0.9-8

Title Low-Level R to Java Interface

Author Simon Urbanek <simon.urbanek@r-project.org>

Maintainer Simon Urbanek <simon.urbanek@r-project.org>

Depends R (>= 2.5.0), methods

Description Low-level interface to Java VM very much like .C/.Call and friends. Allows creation of objects, calling methods and accessing fields.

License GPL-2

URL http://www.rforge.net/rJava/

SystemRequirements Java JDK 1.2 or higher (for JRI/REngine JDK 1.4 or higher), GNU make

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-01-07 15:09:40

Ilustración 7: rJava

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

7.3.4. xlsx

Este paquete deja importar archivos en formato Excel. Permite leer y escribir este tipo de archivos. Está basado en Java, por ese motivo es necesario el paquete anteriormente descrito "rJava".(«R Package Snapshot», s. f.-d)

Package 'xlsx'

February 20, 2015

Type Package

Title Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files

Version 0.5.7

Date 2014-08-01

Depends rJava, xlsxjars

LazyLoad yes

Author Adrian A. Dragulescu

Maintainer Adrian A. Dragulescu <adrian.dragulescu@gmail.com>

Description
Provide R functions to read/write/format Excel 2007 and Excel 97/2000/XP/2003 file formats.

License GPL-3

URL http://code.google.com/p/rexcel/,
http://groups.google.com/group/R-package-xlsx

BugReports https://code.google.com/p/rexcel/issues/list

NeedsCompilation no

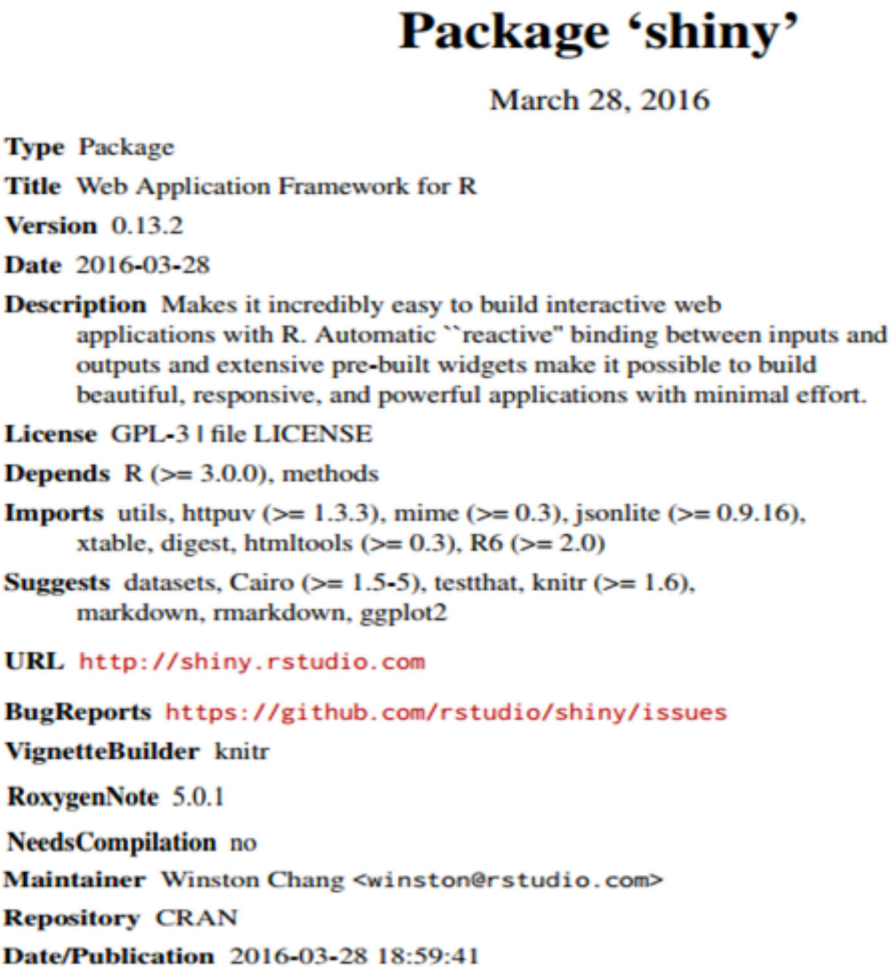
Repository CRAN

Date/Publication 2014-08-02 23:32:07

Ilustración 8: xlsx

7.3.5. shiny

Shiny permite de manera sencilla construir aplicaciones web interactivas con R. El usuario de la app solo tiene que seleccionar los campos correspondientes.(«R Package Snapshot», s. f.-e)



Package 'shiny'

March 28, 2016

Type Package

Title Web Application Framework for R

Version 0.13.2

Date 2016-03-28

Description Makes it incredibly easy to build interactive web applications with R. Automatic ``reactive" binding between inputs and outputs and extensive pre-built widgets make it possible to build beautiful, responsive, and powerful applications with minimal effort.

License GPL-3 | file LICENSE

Depends R (>= 3.0.0), methods

Imports utils, httpuv (>= 1.3.3), mime (>= 0.3), jsonlite (>= 0.9.16), xtable, digest, htmltools (>= 0.3), R6 (>= 2.0)

Suggests datasets, Cairo (>= 1.5-5), testthat, knitr (>= 1.6), markdown, rmarkdown, ggplot2

URL http://shiny.rstudio.com

BugReports https://github.com/rstudio/shiny/issues

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Maintainer Winston Chang <winston@rstudio.com>

Repository CRAN

Date/Publication 2016-03-28 18:59:41

Ilustración 9: shiny

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

7.3.6. igraph

El paquete igraph proporciona rutinas y funciones para generar y manipular grafos facilmente.(«R Package Snapshot», s. f.-f)

Package 'igraph'

June 26, 2015

Version 1.0.1
Title Network Analysis and Visualization
Author See AUTHORS file.
Maintainer Gabor Csardi <csardi.gabor@gmail.com>
Description Routines for simple graphs and network analysis. It can handle large graphs very well and provides functions for generating random and regular graphs, graph visualization, centrality methods and much more.
Depends methods
Imports Matrix, magrittr, NMF, irlba
Suggests igraphdata, stats4, rgl, tcltk, graph, ape, scales
License GPL (>= 2)
URL http://igraph.org
SystemRequirements gmp, libxml2
BugReports https://github.com/igraph/igraph/issues
Collate 'adjacency.R' 'auto.R' 'assortativity.R' 'attributes.R' 'basic.R' 'bipartite.R' 'centrality.R' 'centralization.R' 'cliques.R' 'cocitation.R' 'cohesive.blocks.R' 'printr.R' 'community.R' 'components.R' 'console.R' 'conversion.R' 'data_frame.R' 'decomposition.R' 'degseq.R' 'demo.R' 'embedding.R' 'epi.R' 'fit.R' 'flow.R' 'foreign.R' 'games.R' 'glet.R' 'hrg.R' 'igraph-package.R' 'incidence.R' 'indexing.R' 'interface.R' 'iterators.R' 'layout.R' 'layout_drl.R' 'lazyeval.R' 'make.R' 'mgclust.R' 'minimum.spanning.tree.R' 'motifs.R' 'nexus.R' 'operators.R' 'other.R' 'package.R' 'palette.R' 'par.R' 'paths.R' 'plot.R' 'plot.common.R' 'plot.shapes.R' 'pp.R' 'print.R' 'random_walk.R' 'rewire.R' 'scan.R' 'scg.R' 'sgm.R' 'similarity.R' 'simple.R' 'sir.R' 'socnet.R' 'sparsedf.R' 'structural.properties.R' 'structure.info.R' 'test.R' 'tkplot.R' 'topology.R' 'triangles.R' 'utils.R' 'uuid.R' 'versions.R' 'weakref.R' 'zzz-deprecate.R'
NeedsCompilation yes

Ilustración 10: igraph

7.4. Otros softwares: Micrrosoft Office

Dentro del paquete Microsoft Office 2007 se ha utilizado Excel para realizar las tablas de datos como son las distancias entre ciudades y los precios de los productos.

Ilustración 11: Microsoft Office

8. ANÁLISIS DE LA RED DE APROVISIONAMIENTO

En este TFG el conjunto de ciudades que forman parte de la red de aprovisionamiento son todas las capitales de Provincia y Comunidad Autónoma del territorio peninsular español. Hacen una suma de 49 ciudades

Como se describió anteriormente entre las variantes del TSP se podía diferenciar entre simétrico o asimétrico según si la distancia entre las ciudades era la misma o no en ambos sentidos. En este caso para que sea más sencillo se decide que sea simétrico, es decir, que las distancias entre las ciudades sea la misma en ambos sentidos.

Las distancias entre las capitales se han obtenido de una página web donde se introducía los nombres de dos ciudades y te devolvía la distancia por carretera entre ellas. («Distancia entre dos ciudades», s. f.)

8.1. CAPITALES AUTONÓMICAS Y PROVINCIALES

Comunidad autónoma	Capitales autonómicas y provinciales
Andalucía	Almería Cádiz Córdoba Granada Huelva Jaén Málaga Sevilla
Aragón	Huesca Teruel Zaragoza

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Principado de Asturias	Oviedo
Cantabria	Santander
Castilla- La Mancha	Albacete Ciudad Real Cuenca Guadalajara Toledo
Castilla y León	Ávila Burgos León Palencia Salamanca Segovia Soria Valladolid Zamora
Cataluña	Barcelona Gerona Lérida Tarragona
Comunidad Foral de Navarra	Pamplona
Comunidad de Madrid	Madrid
Comunidad Valenciana	Alicante Castellón de la Plana

	Valencia
Extremadura	Badajoz Cáceres Mérida
Galicia	La Coruña Lugo Orense Pontevedra Santiago de Compostela
La Rioja	Logroño
País Vasco	Bilbao San Sebastián Vitoria
Región de Murcia	Murcia

Tabla 1: Ciudades

8.2. Mapa político de España

Ilustración 12: Mapa de España

8.3. RED DE ADYACENCIA

	Albacete	Alicante	Almerìa	Ávila	Badajoz	Barcelona	Bilbao
Albacete	0	168	Inf	Inf	Inf	Inf	Inf
Alicante	168	0	Inf	Inf	Inf	Inf	Inf
Almería	Inf	Inf	0	Inf	Inf	Inf	Inf
Ávila	Inf	Inf	Inf	0	Inf	Inf	Inf
Badajoz	Inf	Inf	Inf	Inf	0	Inf	Inf
Barcelona	Inf	Inf	Inf	Inf	Inf	0	Inf
Bilbao	Inf	Inf	Inf	Inf	Inf	Inf	0
Burgos	Inf	Inf	Inf	Inf	Inf	Inf	158
Cáceres	Inf	Inf	Inf	238	93	Inf	Inf
Cádiz	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Castellón	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ciudad Real	200	Inf	Inf	Inf	Inf	Inf	Inf
Córdoba	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Cuenca	142	Inf	Inf	Inf	Inf	Inf	Inf
Gerona	Inf	Inf	Inf	Inf	Inf	103	Inf
Granada	Inf	Inf	168	Inf	Inf	Inf	Inf
Guadalajara	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Huelva	Inf	Inf	Inf	Inf	292	Inf	Inf
Huesca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Jaén	305	Inf	225	Inf	Inf	Inf	Inf
La Coruña	Inf	Inf	Inf	Inf	Inf	Inf	Inf
León	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Lérida	Inf	Inf	Inf	Inf	Inf	163	Inf
Logroño	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Lugo	Inf	Inf	Inf	Inf	Inf	Inf	Inf

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Madrid	Inf	Inf	Inf	108	Inf	Inf	Inf
Málaga	Inf	Inf	201	Inf	Inf	Inf	Inf
Mérida	Inf	Inf	Inf	Inf	66	Inf	Inf
Murcia	144	82	218	Inf	Inf	Inf	Inf
Orense	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Oviedo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Palencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Pamplona	Inf	Inf	Inf	Inf	Inf	Inf	155
Pontevedra	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Salamanca	Inf	Inf	Inf	109	Inf	Inf	Inf
S.Sebastian	Inf	Inf	Inf	Inf	Inf	Inf	101
Santander	Inf	Inf	Inf	Inf	Inf	Inf	100
S.Compostela	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Segovia	Inf	Inf	Inf	66	Inf	Inf	Inf
Sevilla	Inf	Inf	Inf	Inf	210	Inf	Inf
Soria	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Tarragona	Inf	Inf	Inf	Inf	Inf	100	Inf
Teruel	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Toledo	247	Inf	Inf	131	Inf	Inf	Inf
Valencia	187	166	Inf	Inf	Inf	Inf	Inf
Valladolid	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Vitoria	Inf	Inf	Inf	Inf	Inf	Inf	62
Zamora	Inf	Inf	Inf	179	Inf	Inf	Inf
Zaragoza	Inf	Inf	Inf	Inf	Inf	313	Inf

Tabla 2: Matriz de adyacencia

	Burgos	Cáceres	Cádiz	Castellón	Ciudad Real	Córdoba	Cuenca
Albacete	Inf	Inf	Inf	Inf	200	Inf	142
Alicante	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Almería	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ávila	Inf	238	Inf	Inf	Inf	Inf	Inf
Badajoz	Inf	93	Inf	Inf	Inf	Inf	Inf
Barcelona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Bilbao	158	Inf	Inf	Inf	Inf	Inf	Inf
Burgos	0	Inf	Inf	Inf	Inf	Inf	Inf
Cáceres	Inf	0	Inf	Inf	Inf	Inf	Inf
Cádiz	Inf	Inf	0	Inf	Inf	Inf	Inf
Castellón	Inf	Inf	Inf	0	Inf	Inf	263
Ciudad Real	Inf	Inf	Inf	Inf	0	192	267
Córdoba	Inf	Inf	Inf	Inf	192	0	Inf
Cuenca	Inf	Inf	Inf	263	267	Inf	0
Gerona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Granada	Inf	Inf	Inf	Inf	Inf	202	Inf
Guadalajara	Inf	Inf	Inf	Inf	Inf	Inf	135
Huelva	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Huesca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Jaén	Inf	Inf	Inf	Inf	170	120	Inf
La Coruña	Inf	Inf	Inf	Inf	Inf	Inf	Inf
León	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Lérida	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Logroño	134	Inf	Inf	Inf	Inf	Inf	Inf
Lugo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Madrid	Inf	Inf	Inf	Inf	Inf	Inf	165
Málaga	Inf	Inf	234	Inf	Inf	158	Inf

Mérida	Inf	75	Inf	Inf	243	245	Inf
Murcia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Orense	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Oviedo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Palencia	90	Inf	Inf	Inf	Inf	Inf	Inf
Pamplona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Pontevedra	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Salamanca	Inf	201	Inf	Inf	Inf	Inf	Inf
S.Sebastian	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Santander	181	Inf	Inf	Inf	Inf	Inf	Inf
S.Compostela	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Segovia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Sevilla	Inf	Inf	122	Inf	Inf	140	Inf
Soria	142	Inf	Inf	Inf	Inf	Inf	Inf
Tarragona	Inf	Inf	Inf	187	Inf	Inf	Inf
Teruel	Inf	Inf	Inf	144	Inf	Inf	147
Toledo	Inf	260	Inf	Inf	118	Inf	179
Valencia	Inf	Inf	Inf	73	Inf	Inf	199
Valladolid	127	Inf	Inf	Inf	Inf	Inf	Inf
Vitoria	118	Inf	Inf	Inf	Inf	Inf	Inf
Zamora	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Zaragoza	Inf	Inf	Inf	Inf	Inf	Inf	Inf

Tabla 3: Matriz de adyacencia

	Gerona	Granada	Guadalajara	Huelva	Huesca	Jaén	La Coruña
Albacete	Inf	Inf	Inf	Inf	Inf	305	Inf
Alicante	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Almería	Inf	168	Inf	Inf	Inf	225	Inf
Ávila	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Badajoz	Inf	Inf	Inf	292	Inf	Inf	Inf
Barcelona	103	Inf	Inf	Inf	Inf	Inf	Inf
Bilbao	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Burgos	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Cáceres	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Cádiz	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Castellón	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ciudad Real	Inf	Inf	Inf	Inf	Inf	170	Inf
Córdoba	Inf	202	Inf	Inf	Inf	120	Inf
Cuenca	Inf	Inf	135	Inf	Inf	Inf	Inf
Gerona	0	Inf	Inf	Inf	340	Inf	Inf
Granada	Inf	0	Inf	Inf	Inf	92	Inf
Guadalajara	Inf	Inf	0	Inf	Inf	Inf	Inf
Huelva	Inf	Inf	Inf	0	Inf	Inf	Inf
Huesca	340	Inf	Inf	Inf	0	Inf	Inf
Jaén	Inf	92	Inf	Inf	Inf	0	Inf
La Coruña	Inf	Inf	Inf	Inf	Inf	Inf	0
León	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Lérida	231	Inf	Inf	Inf	112	Inf	Inf
Logroño	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Lugo	Inf	Inf	Inf	Inf	Inf	Inf	97
Madrid	Inf	Inf	62	Inf	Inf	Inf	Inf
Málaga	Inf	124	Inf	Inf	Inf	Inf	Inf

Mérida	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Murcia	Inf	277	Inf	Inf	Inf	335	Inf
Orense	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Oviedo	Inf	Inf	Inf	Inf	Inf	Inf	286
Palencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Pamplona	563	Inf	Inf	Inf	165	Inf	Inf
Pontevedra	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Salamanca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
S.Sebastian	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Santander	Inf	Inf	Inf	Inf	Inf	Inf	454
S.Compostela	Inf	Inf	Inf	Inf	Inf	Inf	74
Segovia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Sevilla	Inf	248	Inf	94	Inf	Inf	Inf
Soria	Inf	Inf	171	Inf	Inf	Inf	Inf
Tarragona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Teruel	Inf	Inf	245	Inf	Inf	Inf	Inf
Toledo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Valencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Valladolid	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Vitoria	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Zamora	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Zaragoza	Inf	Inf	257	Inf	74	Inf	Inf

Tabla 4: Matriz de adyacencia

	León	Lérida	Logroño	Lugo	Madrid	Málaga	Mérida	Murcia
Albacete	Inf	Inf	Inf	Inf	Inf	Inf	Inf	144
Alicante	Inf	Inf	Inf	Inf	Inf	Inf	Inf	82
Almería	Inf	Inf	Inf	Inf	Inf	201	Inf	218
Ávila	Inf	Inf	Inf	Inf	108	Inf	Inf	Inf
Badajoz	Inf	Inf	Inf	Inf	Inf	Inf	66	Inf
Barcelona	Inf	163	Inf	Inf	Inf	Inf	Inf	Inf
Bilbao	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Burgos	183	Inf	134	Inf	Inf	Inf	Inf	Inf
Cáceres	Inf	Inf	Inf	Inf	Inf	Inf	75	Inf
Cádiz	Inf	Inf	Inf	Inf	Inf	234	Inf	Inf
Castellón	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ciudad Real	Inf	Inf	Inf	Inf	Inf	Inf	243	Inf
Córdoba	Inf	Inf	Inf	Inf	Inf	158	245	Inf
Cuenca	Inf	Inf	Inf	Inf	165	Inf	Inf	Inf
Gerona	Inf	231	Inf	Inf	Inf	Inf	Inf	Inf
Granada	Inf	Inf	Inf	Inf	Inf	124	Inf	277
Guadalajara	Inf	Inf	Inf	Inf	62	Inf	Inf	Inf
Huelva	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Huesca	Inf	112	Inf	Inf	Inf	Inf	Inf	Inf
Jaén	Inf	Inf	Inf	Inf	Inf	Inf	Inf	335
La Coruña	Inf	Inf	Inf	97	Inf	Inf	Inf	Inf
León	0	Inf	Inf	223	Inf	Inf	Inf	Inf
Lérida	Inf	0	Inf	Inf	Inf	Inf	Inf	Inf
Logroño	Inf	Inf	0	Inf	Inf	Inf	Inf	Inf
Lugo	223	Inf	Inf	0	Inf	Inf	Inf	Inf
Madrid	Inf	Inf	Inf	Inf	0	Inf	Inf	Inf
Málaga	Inf	Inf	Inf	Inf	Inf	0	Inf	Inf

Mérida	Inf	Inf	Inf	Inf	Inf	Inf	0	Inf
Murcia	Inf	Inf	Inf	Inf	Inf	Inf	Inf	0
Orense	296	Inf	Inf	94	Inf	Inf	Inf	Inf
Oviedo	125	Inf	Inf	227	Inf	Inf	Inf	Inf
Palencia	131	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Pamplona	Inf	Inf	85	Inf	Inf	Inf	Inf	Inf
Pontevedra	Inf	Inf	Inf	195	Inf	Inf	Inf	Inf
Salamanca	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
S.Sebastian	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Santander	264	Inf	Inf	Inf	Inf	Inf	Inf	Inf
S.Compostela	Inf	Inf	Inf	115	Inf	Inf	Inf	Inf
Segovia	Inf	Inf	Inf	Inf	92	Inf	Inf	Inf
Sevilla	Inf	Inf	Inf	Inf	Inf	206	192	Inf
Soria	Inf	Inf	101	Inf	Inf	Inf	Inf	Inf
Tarragona	Inf	100	Inf	Inf	Inf	Inf	Inf	Inf
Teruel	Inf	323	Inf	Inf	Inf	Inf	Inf	Inf
Toledo	Inf	Inf	Inf	Inf	72	Inf	301	Inf
Valencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf	228
Valladolid	136	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Vitoria	Inf	Inf	93	Inf	Inf	Inf	Inf	Inf
Zamora	142	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Zaragoza	Inf	152	171	Inf	Inf	Inf	Inf	Inf

Tabla 5: Matriz de adyacencia

	Orense	Oviedo	Palencia	Pamplona	Pontevedra	Salamanca
Albacete	Inf	Inf	Inf	Inf	Inf	Inf
Alicante	Inf	Inf	Inf	Inf	Inf	Inf
Almería	Inf	Inf	Inf	Inf	Inf	Inf
Ávila	Inf	Inf	Inf	Inf	Inf	109
Badajoz	Inf	Inf	Inf	Inf	Inf	Inf
Barcelona	Inf	Inf	Inf	Inf	Inf	Inf
Bilbao	Inf	Inf	Inf	155	Inf	Inf
Burgos	Inf	Inf	90	Inf	Inf	Inf
Cáceres	Inf	Inf	Inf	Inf	Inf	201
Cádiz	Inf	Inf	Inf	Inf	Inf	Inf
Castellón	Inf	Inf	Inf	Inf	Inf	Inf
Ciudad Real	Inf	Inf	Inf	Inf	Inf	Inf
Córdoba	Inf	Inf	Inf	Inf	Inf	Inf
Cuenca	Inf	Inf	Inf	Inf	Inf	Inf
Gerona	Inf	Inf	Inf	563	Inf	Inf
Granada	Inf	Inf	Inf	Inf	Inf	Inf
Guadalajara	Inf	Inf	Inf	Inf	Inf	Inf
Huelva	Inf	Inf	Inf	Inf	Inf	Inf
Huesca	Inf	Inf	Inf	165	Inf	Inf
Jaén	Inf	Inf	Inf	Inf	Inf	Inf
La Coruña	Inf	286	Inf	Inf	Inf	Inf
León	296	125	131	Inf	Inf	Inf
Lérida	Inf	Inf	Inf	Inf	Inf	Inf
Logroño	Inf	Inf	Inf	85	Inf	Inf
Lugo	94	227	Inf	Inf	195	Inf

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Madrid	Inf	Inf	Inf	Inf	Inf	Inf
Málaga	Inf	Inf	Inf	Inf	Inf	Inf
Mérida	Inf	Inf	Inf	Inf	Inf	Inf
Murcia	Inf	Inf	Inf	Inf	Inf	Inf
Orense	0	Inf	Inf	Inf	117	Inf
Oviedo	Inf	0	Inf	Inf	Inf	Inf
Palencia	Inf	Inf	0	Inf	Inf	Inf
Pamplona	Inf	Inf	Inf	0	Inf	Inf
Pontevedra	117	Inf	Inf	Inf	0	Inf
Salamanca	Inf	Inf	Inf	Inf	Inf	0
S.Sebastian	Inf	Inf	Inf	83	Inf	Inf
Santander	Inf	194	Inf	Inf	Inf	Inf
S.Compostela	106	Inf	Inf	Inf	70	Inf
Segovia	Inf	Inf	Inf	Inf	Inf	Inf
Sevilla	Inf	Inf	Inf	Inf	Inf	Inf
Soria	Inf	Inf	Inf	Inf	Inf	Inf
Tarragona	Inf	Inf	Inf	Inf	Inf	Inf
Teruel	Inf	Inf	Inf	Inf	Inf	Inf
Toledo	Inf	Inf	Inf	Inf	Inf	Inf
Valencia	Inf	Inf	Inf	Inf	Inf	Inf
Valladolid	Inf	Inf	51	Inf	Inf	Inf
Vitoria	Inf	Inf	Inf	99	Inf	Inf
Zamora	269	Inf	149	Inf	Inf	66
Zaragoza	Inf	Inf	Inf	177	Inf	Inf

Tabla 6: Matriz de adyacencia

	S.Sebatián	Santander	S.Compostela	Segovia	Sevilla	Soria	Tarragona
Albacete	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Alicante	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Almería	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ávila	Inf	Inf	Inf	66	Inf	Inf	Inf
Badajoz	Inf	Inf	Inf	Inf	210	Inf	Inf
Barcelona	Inf	Inf	Inf	Inf	Inf	Inf	100
Bilbao	101	100	Inf	Inf	Inf	Inf	Inf
Burgos	Inf	181	Inf	Inf	Inf	142	Inf
Cáceres	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Cádiz	Inf	Inf	Inf	Inf	122	Inf	Inf
Castellón	Inf	Inf	Inf	Inf	Inf	Inf	187
Ciudad Real	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Córdoba	Inf	Inf	Inf	Inf	140	Inf	Inf
Cuenca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Gerona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Granada	Inf	Inf	Inf	Inf	248	Inf	Inf
Guadalajara	Inf	Inf	Inf	Inf	Inf	171	Inf
Huelva	Inf	Inf	Inf	Inf	94	Inf	Inf
Huesca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Jaén	Inf	Inf	Inf	Inf	Inf	Inf	Inf
La Coruña	Inf	454	74	Inf	Inf	Inf	Inf
León	Inf	264	Inf	Inf	Inf	Inf	Inf
Lérida	Inf	Inf	Inf	Inf	Inf	Inf	100
Logroño	Inf	Inf	Inf	Inf	Inf	101	Inf
Lugo	Inf	Inf	115	Inf	Inf	Inf	Inf
Madrid	Inf	Inf	Inf	92	Inf	Inf	Inf
Málaga	Inf	Inf	Inf	Inf	206	Inf	Inf

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Mérida	Inf	Inf	Inf	Inf	192	Inf	Inf
Murcia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Orense	Inf	Inf	106	Inf	Inf	Inf	Inf
Oviedo	Inf	194	Inf	Inf	Inf	Inf	Inf
Palencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Pamplona	83	Inf	Inf	Inf	Inf	Inf	Inf
Pontevedra	Inf	Inf	70	Inf	Inf	Inf	Inf
Salamanca	Inf	Inf	Inf	Inf	Inf	Inf	Inf
S.Sebastian	0	Inf	Inf	Inf	Inf	Inf	Inf
Santander	Inf	0	Inf	Inf	Inf	Inf	Inf
S.Compostela	Inf	Inf	0	Inf	Inf	Inf	Inf
Segovia	Inf	Inf	Inf	0	Inf	192	Inf
Sevilla	Inf	Inf	Inf	Inf	0	Inf	Inf
Soria	Inf	Inf	Inf	192	Inf	0	Inf
Tarragona	Inf	Inf	Inf	Inf	Inf	Inf	0
Teruel	Inf	Inf	Inf	Inf	Inf	Inf	336
Toledo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Valencia	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Valladolid	Inf	Inf	Inf	116	Inf	208	Inf
Vitoria	101	Inf	Inf	Inf	Inf	Inf	Inf
Zamora	Inf	Inf	Inf	190	Inf	Inf	Inf
Zaragoza	Inf	Inf	Inf	Inf	Inf	159	236

Tabla 7: Matriz de adyacencia

	Teruel	Toledo	Valencia	Valladolid	Vitoria	Zamora	Zaragoza
Albacete	Inf	247	187	Inf	Inf	Inf	Inf
Alicante	Inf	Inf	166	Inf	Inf	Inf	Inf
Almería	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Ávila	Inf	131	Inf	Inf	Inf	179	Inf
Badajoz	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Barcelona	Inf	Inf	Inf	Inf	Inf	Inf	313
Bilbao	Inf	Inf	Inf	Inf	62	Inf	Inf
Burgos	Inf	Inf	Inf	127	118	Inf	Inf
Cáceres	Inf	260	Inf	Inf	Inf	Inf	Inf
Cádiz	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Castellón	144	Inf	73	Inf	Inf	Inf	Inf
Ciudad Real	Inf	118	Inf	Inf	Inf	Inf	Inf
Córdoba	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Cuenca	147	179	199	Inf	Inf	Inf	Inf
Gerona	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Granada	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Guadalajara	245	Inf	Inf	Inf	Inf	Inf	257
Huelva	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Huesca	Inf	Inf	Inf	Inf	Inf	Inf	74
Jaén	Inf	Inf	Inf	Inf	Inf	Inf	Inf
La Coruña	Inf	Inf	Inf	Inf	Inf	Inf	Inf
León	Inf	Inf	Inf	136	Inf	142	Inf
Lérida	323	Inf	Inf	Inf	Inf	Inf	152
Logroño	Inf	Inf	Inf	Inf	93	Inf	171
Lugo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Madrid	Inf	72	Inf	Inf	Inf	Inf	Inf
Málaga	Inf	Inf	Inf	Inf	Inf	Inf	Inf

Mérida	Inf	301	Inf	Inf	Inf	Inf	Inf
Murcia	Inf	Inf	228	Inf	Inf	Inf	Inf
Orense	Inf	Inf	Inf	Inf	Inf	269	Inf
Oviedo	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Palencia	Inf	Inf	Inf	51	Inf	149	Inf
Pamplona	Inf	Inf	Inf	Inf	99	Inf	177
Pontevedra	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Salamanca	Inf	Inf	Inf	Inf	Inf	66	Inf
S.Sebastian	Inf	Inf	Inf	Inf	101	Inf	Inf
Santander	Inf	Inf	Inf	Inf	Inf	Inf	Inf
S.Compostela	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Segovia	Inf	Inf	Inf	116	Inf	190	Inf
Sevilla	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Soria	Inf	Inf	Inf	208	Inf	Inf	159
Tarragona	336	Inf	Inf	Inf	Inf	Inf	236
Teruel	0	Inf	144	Inf	Inf	Inf	171
Toledo	Inf	0	Inf	Inf	Inf	Inf	Inf
Valencia	144	Inf	0	Inf	Inf	Inf	Inf
Valladolid	Inf	Inf	Inf	0	Inf	101	Inf
Vitoria	Inf	Inf	Inf	Inf	0	Inf	Inf
Zamora	Inf	Inf	Inf	101	Inf	0	Inf
Zaragoza	171	Inf	Inf	Inf	Inf	Inf	0

Tabla 8: Matriz de adyacencia

En esta matriz de adyacencia hay que tener en cuenta:

- Los *ceros* representan que la distancia entre una ciudad y ella misma es nula.
- Los *inf* representan que entre dos ciudades no hay conexión directa.
- Los *numeros* reperesentan la distancia entre dos ciudades medida en kilometros.

Ilustración 13: Red de adyacencia

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

9. OBTENCIÓN DE LA SOLUCIÓN

Una vez definida la red de adyacencia se vuelcan los datos de las distancias a R-project para la obtención del grafo. Es importante mencionar que los elementos *inf* son sustituidos por ceros para que pueda ser leído por el programa.

Para este volcado de los datos se utiliza la función `matrix(rbind())`.

Los nombres de las ciudades se añaden con la función `rownames()` para las filas y `colnames()` para las columnas.

```
> Grafo
           Albacete Alicante Almeria Avila Badajoz Barcelona Bilbao Burgos
Albacete          0      168       0     0       0         0      0       0
Alicante        168        0       0     0       0         0      0       0
Almeria           0        0       0     0       0         0      0       0
Avila             0        0       0     0       0         0      0       0
Badajoz           0        0       0     0       0         0      0       0
Barcelona         0        0       0     0       0         0      0       0
Bilbao            0        0       0     0       0         0      0     158
Burgos            0        0       0     0       0         0    158       0
Caceres           0        0       0   238      93         0      0       0
Cadiz             0        0       0     0       0         0      0       0
Castellon         0        0       0     0       0         0      0       0
Ciudad Real     200        0       0     0       0         0      0       0
```

Ilustración 14: Grafo R-project

Con la función `attributes()` se comprueba que la matriz tiene dimensiones 49x49 y que los nombres de las ciudades son los correctos.

```
> attributes(Grafo)
$dim
[1] 49 49

$dimnames
$dimnames[[1]]
 [1] "Albacete"      "Alicante"      "Almeria"       "Avila"
 [5] "Badajoz"       "Barcelona"     "Bilbao"        "Burgos"
 [9] "Caceres"       "Cadiz"         "Castellon"     "Ciudad Real"
[13] "Cordoba"       "Cuenca"        "Gerona"        "Granada"
[17] "Guadalajara"   "Huelva"        "Huesca"        "Jaen"
[21] "La Coruña"     "Leon"          "Lerida"        "Logroño"
[25] "Lugo"          "Madrid"        "Malaga"        "Merida"
[29] "Murcia"        "Orense"        "Oviedo"        "Palencia"
[33] "Pamplona"      "Pontevedra"    "Salamanca"     "San Sebastian"
[37] "Santander"     "S.Compostela"  "Segovia"       "Sevilla"
[41] "Soria"         "Tarragona"     "Teruel"        "Toledo"
[45] "Valencia"      "Valladolid"    "Vitoria"       "Zamora"
[49] "Zaragoza"
```

Ilustración 15: attributes

9.1. GRAFO

Para obtener el grafo asociado a la matriz descrita anteriormente se utiliza la librería `igraph` mediante la funcion `graph.adjacency()`. Los parámetros de esta función son los siguientes:

```
graph_adjacency(adjmatrix, mode = c("directed", "undirected", "max", "min", "upper", "lower", "plus"), weighted = NULL, diag = TRUE, add.colnames = NULL, add.rownames = NA).
```

El significado de sus parámetros es el siguiente:

- adjmatrix: Matriz cuadrada de adyacencia.
- mode: Especifica cómo debe interpretar igraph la matriz suministrada.
- weighted: Determina si se quiere mantener los pesos.
- diag: Especifica si se incluye la diagonal de la matriz en el cálculo.
- add.colnames: Establece si se desea añadir los nombres de las columnas como atributos de vértice.
- add.rownames: Establece si se desea añadir los nombres de las filas como atributos de vértice.

Concretamente, en este caso, se utilizan `adjmatrix` y `weighted` y el resto de parámetros aparecerán con su opción por defecto.

```
G<-graph.adjacency(Grafo, weighted=TRUE)
```

Ilustración 16: graph.adjacency

Con las funciones `V()` y `E()` se verifica que el grafo contiene los 49 nodos y las 257 conexiones entre ellos.(«R: Create graphs from adjacency matrices», s. f.)

```
> V(G)
+ 49/49 vertices, named:
 [1] Albacete      Alicante      Almeria       Avila         Badajoz
 [6] Barcelona     Bilbao        Burgos        Caceres       Cadiz
[11] Castellon     Ciudad Real   Cordoba       Cuenca        Gerona
[16] Granada       Guadalajara   Huelva        Huesca        Jaen
[21] La Coruña     Leon          Lerida        Logroño       Lugo
[26] Madrid        Malaga        Merida        Murcia        Orense
[31] Oviedo        Palencia      Pamplona      Pontevedra    Salamanca
[36] San Sebastian Santander     S.Compostela  Segovia       Sevilla
[41] Soria         Tarragona     Teruel        Toledo        Valencia
[46] Valladolid    Vitoria       Zamora        Zaragoza
```

Ilustración 17: Vertices

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

```
> E(G)
+ 257/257 edges (vertex names):
  [1] Albacete ->Alicante     Albacete ->Ciudad Real Albacete ->Cuenca
  [4] Albacete ->Jaen         Albacete ->Murcia      Albacete ->Toledo
  [7] Albacete ->Valencia     Alicante ->Albacete    Alicante ->Murcia
 [10] Alicante ->Valencia     Almeria  ->Granada     Almeria  ->Jaen
 [13] Almeria  ->Malaga       Almeria  ->Murcia      Avila    ->Caceres
 [16] Avila    ->Madrid       Avila    ->Salamanca   Avila    ->Segovia
 [19] Avila    ->Toledo       Avila    ->Zamora      Badajoz  ->Caceres
 [22] Badajoz  ->Huelva       Badajoz  ->Merida      Badajoz  ->Sevilla
 [25] Barcelona->Gerona       Barcelona->Lerida      Barcelona->Tarragona
 [28] Barcelona->Zaragoza     Bilbao   ->Burgos      Bilbao   ->Pamplona
+ ... omitted several edges
```

Ilustración 18: Conexiones

A continuación se muestra el grafo dinámico obtenido con la función `tkplot()`.

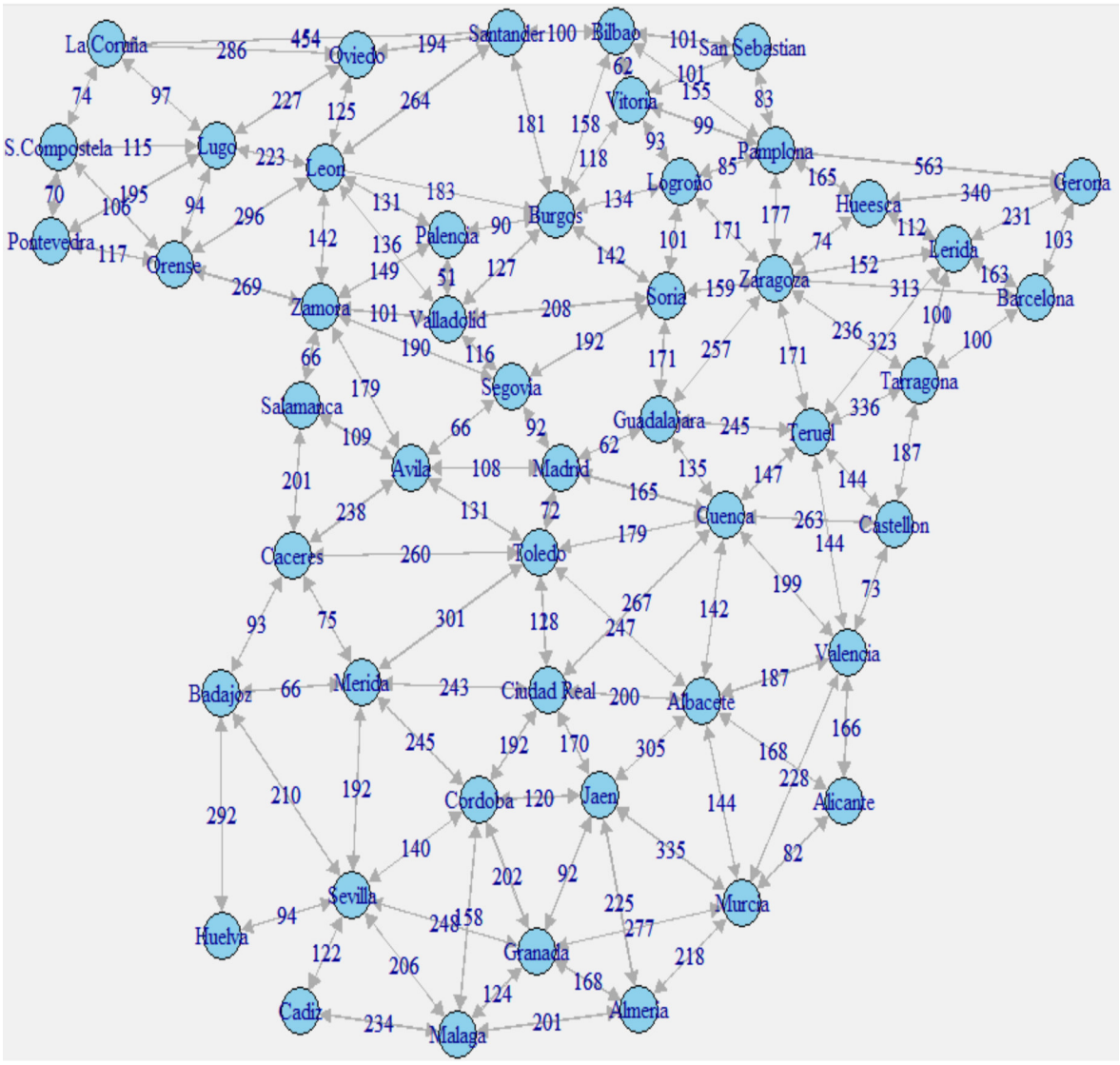


Ilustración 19: Grafo

9.2. MATRIZ PRINCIPAL

En la aplicación se necesita que la matriz contenga la distancias entre todas las ciudades. Esto se puede obtener con la función shortest.paths().

	Albacete	Alicante	Almería	Ávila	Badajoz	Barcelona	Bilbao
Albacete	0	168	354	368	503	511	639
Alicante	168	0	291	534	688	524	787
Almería	354	291	0	658	610	796	942
Ávila	368	534	658	0	328	731	416
Badajoz	503	688	610	328	0	1026	691
Barcelona	511	524	796	731	1026	0	610
Bilbao	639	787	942	416	691	610	0
Burgos	486	652	789	263	538	606	158
Cáceres	507	673	664	238	93	920	601
Cádiz	621	640	436	615	328	1116	978
Castellón	235	248	520	531	726	280	611
Ciudad Real	200	385	392	250	311	695	598
Córdoba	370	555	353	504	265	865	788
Cuenca	142	308	496	276	547	541	552
Gerona	604	617	889	809	1104	103	687
Granada	360	351	168	526	464	888	811
Guadalajara	281	447	601	168	464	564	399
Huelva	595	692	497	580	292	1090	943
Huesca	469	561	834	493	788	272	372
Jaén	305	407	225	438	376	799	723
La Coruña	851	1017	1141	519	655	1088	543
León	597	762	886	264	498	784	336

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Lérida	473	486	758	571	866	163	450
Logroño	549	654	868	390	665	477	136
Lugo	759	925	1049	427	660	995	484
Madrid	253	419	552	108	404	624	402
Málaga	502	472	201	636	421	997	920
Mérida	442	626	590	308	66	961	671
Murcia	144	82	218	511	670	589	783
Orense	760	926	1049	427	567	1057	609
Oviedo	705	871	995	373	607	890	282
Palencia	499	665	789	187	462	691	243
Pamplona	611	662	934	467	743	485	155
Pontevedra	872	1038	1162	540	523	1170	640
Salamanca	473	639	763	109	292	845	397
S.Sebastian	693	748	996	470	745	571	101
Santander	666	832	984	382	657	708	100
S.Compostela	862	1028	1151	529	591	1104	560
Segovia	351	517	640	66	390	663	352
Sevilla	501	595	401	498	210	996	861
Soria	449	551	769	253	632	467	222
Tarragona	419	432	704	655	949	100	533
Teruel	221	319	591	412	706	428	473
Toledo	247	413	499	131	367	692	468
Valencia	187	166	438	465	661	351	611
Valladolid	450	616	739	120	414	728	280
Vitoria	598	744	901	374	650	567	62
Zamora	513	679	803	179	358	825	377
Zaragoza	392	484	756	424	718	313	302

Tabla 9: Matriz principal

	Burgos	Cáceres	Cádiz	Castellón	Ciudad Real	Córdoba	Cuenca
Albacete	486	507	621	235	200	370	142
Alicante	652	673	640	248	385	555	308
Almería	789	664	436	520	392	353	496
Ávila	263	238	615	531	250	504	276
Badajoz	538	93	328	726	311	265	547
Barcelona	606	920	1116	280	695	865	541
Bilbao	158	601	978	611	598	788	552
Burgos	0	449	826	516	445	636	399
Cáceres	449	0	382	694	272	265	440
Cádiz	826	382	0	839	448	260	684
Castellón	516	694	839	0	417	587	263
Ciudad Real	445	272	448	417	0	192	267
Córdoba	636	265	260	587	192	0	431
Cuenca	399	440	684	263	267	431	0
Gerona	685	997	1208	372	786	955	633
Granada	659	518	291	610	259	202	455
Guadalajara	247	357	708	383	255	446	135
Huelva	790	347	210	812	417	233	657
Huesca	369	681	1032	386	579	769	367
Jaén	570	376	330	521	170	120	367
La Coruña	487	662	1041	1014	796	986	760
León	183	408	786	759	541	731	505
Lérida	447	759	1110	241	657	847	441
Logroño	134	575	953	479	523	713	369
Lugo	395	570	948	921	703	894	668
Madrid	249	297	648	419	206	397	165
Málaga	768	475	234	719	341	158	564

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Mérida	519	75	309	658	243	245	480
Murcia	631	651	566	311	361	476	286
Orense	456	526	905	922	704	841	668
Oviedo	296	516	895	868	650	840	614
Palencia	90	372	750	555	444	634	400
Pamplona	211	652	1031	487	585	775	432
Pontevedra	569	573	862	1035	817	797	781
Salamanca	244	201	580	636	352	515	382
S.Sebastian	213	655	1033	573	651	841	517
Santander	181	567	945	710	639	829	578
S.Compostela	504	628	929	1024	806	865	770
Segovia	199	300	678	513	295	485	259
Sevilla	708	265	122	718	323	140	564
Soria	142	525	876	377	423	613	270
Tarragona	531	842	1023	187	601	770	448
Teruel	378	600	782	144	361	530	147
Toledo	315	260	611	433	118	344	179
Valencia	589	629	774	73	352	521	199
Valladolid	127	324	702	612	394	584	358
Vitoria	118	559	938	569	555	745	510
Zamora	225	268	646	675	458	582	422
Zaragoza	300	611	962	309	510	700	289

Tabla 10: Matriz principal

	Gerona	Granada	Guadalajara	Huelva	Huesca	Jaén	La Coruña
Albacete	604	360	281	595	469	305	851
Alicante	617	351	447	692	561	407	1017
Almería	889	168	601	497	834	225	1141
Ávila	809	526	168	580	493	438	519
Badajoz	1104	464	464	292	788	376	655
Barcelona	103	888	564	1090	272	799	1088
Bilbao	687	811	399	943	372	723	543
Burgos	685	659	247	790	369	570	487
Cáceres	997	518	357	347	681	376	662
Cádiz	1208	291	708	210	1032	330	1041
Castellón	372	610	383	812	386	521	1014
Ciudad Real	786	259	255	417	579	170	796
Córdoba	955	202	446	233	769	120	986
Cuenca	633	455	135	657	367	367	760
Gerona	0	980	641	1182	340	891	1165
Granada	980	0	468	344	792	92	1009
Guadalajara	641	468	0	672	327	382	651
Huelva	1182	344	672	0	997	348	943
Huesca	340	792	327	997	0	706	849
Jaén	891	92	382	348	706	0	922
La Coruña	1165	1009	651	943	849	922	0
León	862	754	396	751	546	667	315
Lérida	231	870	405	1075	112	783	928
Logroño	555	736	270	918	239	649	614
Lugo	1073	916	559	914	756	830	97
Madrid	702	419	62	613	385	333	591
Málaga	1089	124	580	303	903	202	1119

Mérida	1038	438	398	274	722	350	732
Murcia	681	277	425	621	625	335	994
Orense	1135	917	559	855	818	830	173
Oviedo	967	863	505	860	651	776	286
Palencia	768	656	248	715	452	570	440
Pamplona	563	798	333	996	165	712	692
Pontevedra	1247	1030	672	812	931	943	133
Salamanca	922	631	273	545	606	544	460
S.Sebastian	638	864	418	998	251	777	639
Santander	786	852	426	911	469	765	454
S.Compostela	1182	1019	661	879	866	932	74
Segovia	741	508	150	643	424	421	527
Sevilla	1088	248	590	94	914	247	922
Soria	545	636	171	841	228	550	619
Tarragona	192	793	488	998	212	706	1011
Teruel	520	553	245	758	249	466	895
Toledo	770	366	129	576	453	280	658
Valencia	443	498	333	749	387	457	948
Valladolid	805	607	249	667	489	520	439
Vitoria	645	768	358	903	329	682	599
Zamora	903	670	313	611	587	584	395
Zaragoza	390	722	257	928	74	636	781

Tabla 11: Matriz principal

	León	Lérida	Logroño	Lugo	Madrid	Málaga	Mérida	Murcia
Albacete	597	473	549	759	253	502	442	144
Alicante	762	486	654	925	419	472	626	82
Almería	886	758	868	1049	552	201	590	218
Ávila	264	571	390	427	108	636	308	511
Badajoz	498	866	665	660	404	421	66	670
Barcelona	784	163	477	995	624	997	961	589
Bilbao	336	450	136	484	402	920	671	783
Burgos	183	447	134	395	249	768	519	631
Cáceres	408	759	575	570	297	475	75	651
Cádiz	786	1110	953	948	648	234	309	566
Castellón	759	241	479	921	419	719	658	311
Ciudad Real	541	657	523	703	206	341	243	361
Córdoba	731	847	713	894	397	158	245	476
Cuenca	505	441	369	668	165	564	480	286
Gerona	862	231	555	1073	702	1089	1038	681
Granada	754	870	736	916	419	124	438	277
Guadalajara	396	405	270	559	62	580	398	425
Huelva	751	1075	918	914	613	303	274	621
Huesca	546	112	239	756	385	903	722	625
Jaén	667	783	649	830	333	202	350	335
La Coruña	315	928	614	97	591	1119	732	994
León	0	619	305	223	337	865	479	741
Lérida	619	0	316	834	463	981	800	550
Logroño	305	316	0	521	329	847	646	692
Lugo	223	834	521	0	498	1026	640	902
Madrid	337	463	329	498	0	529	342	400
Málaga	865	981	847	1026	529	0	395	400

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Mérida	479	800	646	640	342	395	0	604
Murcia	741	550	692	902	400	400	604	0
Orense	296	896	583	94	501	991	597	904
Oviedo	125	729	416	227	446	974	587	850
Palencia	131	530	217	348	240	768	443	644
Pamplona	383	324	85	599	389	910	723	725
Pontevedra	409	1009	696	195	613	948	584	1017
Salamanca	206	684	371	367	215	666	272	618
S.Sebastian	385	410	167	602	452	975	726	838
Santander	264	547	234	395	424	964	638	810
S.Compostela	332	943	630	115	603	1015	699	1006
Segovia	274	502	326	435	92	620	412	495
Sevilla	668	992	835	829	534	206	192	523
Soria	309	306	101	526	227	745	566	593
Tarragona	702	100	401	919	545	904	883	495
Teruel	545	323	341	802	302	664	640	382
Toledo	404	531	397	565	72	478	301	391
Valencia	695	313	479	856	357	620	595	228
Valladolid	136	567	254	346	191	719	395	594
Vitoria	289	406	93	506	356	880	630	742
Zamora	142	664	351	303	254	733	339	658
Zaragoza	472	152	171	688	314	834	652	546

Tabla 12: Matriz principal

	Orense	Oviedo	Palencia	Pamplona	Pontevedra	Salamanca
Albacete	760	705	499	611	872	473
Alicante	926	871	665	662	1038	639
Almería	1049	995	789	934	1162	763
Ávila	427	373	187	467	540	109
Badajoz	567	607	462	743	523	292
Barcelona	1057	890	691	485	1170	845
Bilbao	609	282	243	155	640	397
Burgos	456	296	90	211	569	244
Cáceres	526	516	372	652	573	201
Cádiz	905	895	750	1031	862	580
Castellón	922	868	555	487	1035	636
Ciudad Real	704	650	444	585	817	352
Córdoba	841	840	634	775	797	515
Cuenca	668	614	400	432	781	382
Gerona	1135	967	768	563	1247	922
Granada	917	863	656	798	1030	631
Guadalajara	559	505	248	333	672	273
Huelva	855	860	715	996	812	545
Huesca	818	651	452	165	931	606
Jaén	830	776	570	712	943	544
La Coruña	173	286	440	692	133	460
León	296	125	131	383	409	206
Lérida	896	729	530	324	1009	684
Logroño	583	416	217	85	696	371
Lugo	94	227	348	599	195	367
Madrid	501	446	240	389	613	215
Málaga	991	974	768	910	948	666

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Mérida	597	587	443	723	584	272
Murcia	904	850	644	725	1017	618
Orense	0	335	343	661	117	324
Oviedo	335	0	250	436	384	324
Palencia	343	250	0	296	458	168
Pamplona	661	436	296	0	774	449
Pontevedra	117	384	458	774	0	438
Salamanca	324	324	168	449	438	0
S.Sebastian	664	378	299	83	736	451
Santander	542	194	201	253	550	363
S.Compostela	106	307	447	709	70	426
Segovia	435	435	165	405	549	172
Sevilla	786	786	632	914	745	462
Soria	550	550	204	178	664	323
Tarragona	981	814	616	408	1095	768
Teruel	803	665	419	348	917	518
Toledo	566	514	326	459	680	237
Valencia	856	804	591	486	970	571
Valladolid	346	255	51	332	461	120
Vitoria	568	343	203	99	682	356
Zamora	269	251	149	430	374	66
Zaragoza	750	583	385	177	865	538

Tabla 13: Matriz principal

	S.Sebatian	Santander	S.Compostela	Segovia	Sevilla	Soria	Tarragona
Albacete	693	666	862	351	501	449	419
Alicante	748	832	1028	517	595	551	432
Almería	996	984	1151	640	401	769	704
Ávila	470	382	529	66	498	253	655
Badajoz	745	657	591	390	210	632	949
Barcelona	571	708	1104	663	996	467	100
Bilbao	101	100	560	352	861	222	533
Burgos	213	181	504	199	708	142	531
Cáceres	655	567	628	300	265	525	842
Cádiz	1033	945	929	678	122	876	1023
Castellón	573	710	1024	513	718	377	187
Ciudad Real	651	639	806	295	323	423	601
Córdoba	841	829	865	485	140	613	770
Cuenca	517	578	770	259	564	270	448
Gerona	638	786	1182	741	1088	545	192
Granada	864	852	1019	508	248	636	793
Guadalajara	418	426	661	150	590	171	488
Huelva	998	911	879	643	94	841	998
Huesca	251	469	866	424	914	228	212
Jaén	777	765	932	421	247	550	706
La Coruña	639	454	74	527	922	619	1011
León	385	264	332	274	668	309	702
Lérida	410	547	943	502	992	306	100
Logroño	167	234	630	326	835	101	401
Lugo	602	395	115	435	829	526	919
Madrid	452	424	603	92	534	227	545
Málaga	975	964	1015	620	206	745	904

Mérida	726	638	699	412	192	566	883
Murcia	838	810	1006	495	523	593	495
Orense	664	542	106	435	786	550	981
Oviedo	378	194	307	435	786	550	814
Palencia	299	201	447	165	632	204	616
Pamplona	83	253	709	405	914	178	408
Pontevedra	736	550	70	549	745	664	1095
Salamanca	451	363	426	172	462	323	768
S.Sebastian	0	196	656	407	916	263	493
Santander	196	0	470	360	828	321	632
S.Compostela	656	470	0	536	806	636	1029
Segovia	407	360	536	0	559	192	589
Sevilla	916	828	806	559	0	759	903
Soria	263	321	636	192	759	0	390
Tarragona	493	632	1029	589	903	390	0
Teruel	433	572	904	395	663	239	336
Toledo	523	522	667	158	477	299	618
Valencia	571	710	957	448	654	377	259
Valladolid	335	247	448	116	585	208	651
Vitoria	101	161	616	315	820	187	491
Zamora	443	344	361	190	529	305	749
Zaragoza	263	401	798	358	845	159	236

Tabla 14: Matriz principal

	Teruel	Toledo	Valencia	Valladolid	Vitoria	Zamora	Zaragoza
Albacete	221	247	187	450	598	513	392
Alicante	319	413	166	616	744	679	484
Almería	591	499	438	739	901	803	756
Ávila	412	131	465	120	374	179	424
Badajoz	706	367	661	414	650	358	718
Barcelona	428	692	351	728	567	825	313
Bilbao	473	468	611	280	62	377	302
Burgos	378	315	589	127	118	225	300
Cáceres	600	260	629	324	559	268	611
Cádiz	782	611	774	702	938	646	962
Castellón	144	433	73	612	569	675	309
Ciudad Real	361	118	352	394	555	458	510
Córdoba	530	344	521	584	745	582	700
Cuenca	147	179	199	358	510	422	289
Gerona	520	770	443	805	645	903	390
Granada	553	366	498	607	768	670	722
Guadalajara	245	129	333	249	358	313	257
Huelva	758	576	749	667	903	611	928
Huesca	249	453	387	489	329	587	74
Jaén	466	280	457	520	682	584	636
La Coruña	895	658	948	439	599	395	781
León	545	404	695	136	289	142	472
Lérida	323	531	313	567	406	664	152
Logroño	341	397	479	254	93	351	171
Lugo	802	565	856	346	506	303	688
Madrid	302	72	357	191	356	254	314
Málaga	664	478	620	719	880	733	834

Mérida	640	301	595	395	630	339	652
Murcia	382	391	228	594	742	658	546
Orense	803	566	856	346	568	269	750
Oviedo	665	514	804	255	343	251	583
Palencia	419	326	591	51	203	149	385
Pamplona	348	459	486	332	99	430	177
Pontevedra	917	680	970	461	682	374	865
Salamanca	518	237	571	120	356	66	538
S.Sebastian	433	523	571	335	101	443	263
Santander	572	522	710	247	161	344	401
S.Compostela	904	667	957	448	616	361	798
Segovia	395	158	448	116	315	190	358
Sevilla	663	477	654	585	820	529	845
Soria	239	299	377	208	187	305	159
Tarragona	336	618	259	651	491	749	236
Teruel	0	326	144	422	431	558	171
Toledo	326	0	371	258	428	321	384
Valencia	144	371	0	548	569	612	309
Valladolid	422	258	548	0	238	101	420
Vitoria	431	428	569	238	0	337	261
Zamora	558	321	612	101	337	0	519
Zaragoza	171	384	309	420	261	519	0

Tabla 15: Matriz principal

9.3. CÁLCULO DE LA SOLUCIÓN

La solución se obtiene en dos pasos, en primer lugar se seleccionan las ciudades en las cuales el coste (coste de producto + coste de viaje) es mínimo y en segundo lugar se genera la ruta de compra óptima que recorre las ciudades seleccionadas.

Teniendo la matriz de distancias entre todas las posibles ciudades de compra, se puede acometer el problema de la ruta y para ello se necesitan dos funciones principalmente. Por un lado la función `lp` necesaria para seleccionar las ciudades que minimizan el coste total de la compra de los productos y por otro lado la función `solve_TSP` que permite obtener la mejor ruta una vez conocidas las ciudades donde se compran los productos.

Ahora se va a detallar como funciona cada una de ellas.

9.3.1. lp

Esta función está incluida en el paquete `lpSolve` de optimización e incluye los siguientes parámetros:

```
lp (direction = "min", objective.in, const.mat, const.dir, const.rhs, transpose.constraints = TRUE, int.vec, presolve=0, compute.sens=0, binary.vec, all.int=FALSE, all.bin=FALSE, scale = 196, dense.const, num.bin.solns=1, use.rw=FALSE)
```

Sus parámetros significan lo siguiente:

- direction: "min" para minimizar y "max" para maximizar.
- objective.in: Vector con los coeficientes de la función objetivo.
- const.mat: Matriz de coeficientes de restricción. Una fila por restricción y una columna por variable.
- const.dir: Vector de dirección de las restricciones. "<","<=",">",">=", "=".
- const.rhs: Vector de valores numéricos a la derecha de las restricciones.
- transpose.costraints: De forma predeterminada cada restricción ocupa una fila de const.mat. En caso de querer en columna, transpose.costraints:= FALSE.
- int.vec: Vector numérico que da los índices de las variables que se requieren para ser un número entero.
- presolve: Numérico: ¿presolve? Por defecto 0(no).
- compute.sens: Numérico: ¿calcular la sensibilidad? Por defecto 0(no).
- binary.vec: Vector numérico que da los índices de las variables que se

requieren para ser un número binario.

- all.int: Todas las variables de número entero Por defecto FALSE.
- all.bin: Todas las variables de número binario. Por defecto FALSE.
- scale: Escala. 196 por defecto.
- num.bin.solns: si all.bin = TRUE, el usuario puede solicitar hasta num.bin.solns soluciones óptimas.
- use.rw: Si es TRUE y num.bin.solns > 1 , comprueba si hay algún error. Concretamente, en este caso, solo se utilizan las 5 primeras y el resto de parámetros aparecerán con su opción por defecto.(«lpSolve.pdf», s. f.)

9.3.2. solve_TSP

Esta función está incluida en el paquete "TSP" e incluye los siguientes parámetros:

```
solve_TSP(x, method = NULL, control = NULL, …)
```

Sus parámetros significan lo siguiente:

- x: Objeto *TSP*, generalmente una matriz cuadrada que informa de las distancias entre las ciudades.
- method: Algoritmo con el cual se obtiene la solución. Hay 4 tipos:
  - `nearest_insertion:` En cada paso elige la ciudad más cercana a una ciudad que pertenece a la ruta.
  - `farthest_insertion:` En cada paso elige la ciudad más lejana a una ciudad que pertenece a la ruta.
  - `cheapest_insertion:` En cada paso elige la ciudad que minimice el incremento de la longitud de la ruta.
  - `arbitrary_insertion:` En cada paso elige la ciudad de forma aleatoria entre las que quedan por incluir en la ruta.
- control: Permite controlar el cálculo de las soluciones. En este caso se utilizara para fijar la ciudad origen.
- …: Argumentos adicionales de control.(«R: TSP solver interface», s. f.)

10. CREACIÓN DE LA APLICACIÓN

La aplicación se ha generado a través del software libre RStudio (Version 0.99.893 – © 2009-2016 RStudio, In) descrito anteriormente. Este software permite cargar espacios de trabajos generados anteriormente en R-Project mediante el tipo de archivo R Workspace (.RData), incluye toda la funcionalidad de R y además permite el trabajo con la librería shiny para generación de apps.

Se crean los archivos `.R` que contienen los códigos que harán que la aplicación facilite los resultados que se desean obtener.

Es importante mencionar que todos los archivos deben encontrarse en la misma carpeta. Es posible conocer la ubicación de los archivos con los que se está trabajando mediante la utilización del comando `getwd()` en la consola de RStudio.

10.1. ARCHIVOS GENERADORES DE LA APLICACIÓN

Los archivos .R se generan desde la interfaz de RStudio a través del siguiente proceso:

File ——> New File ——> R Script

Los archivos principales son dos, *server.R* y *ui.R*. Estos dos archivos son indispensables y obligatorios para que la aplicación pueda funcionar y necesariamente el nombre de los dos debe ser ese.(«Shiny», s. f.)

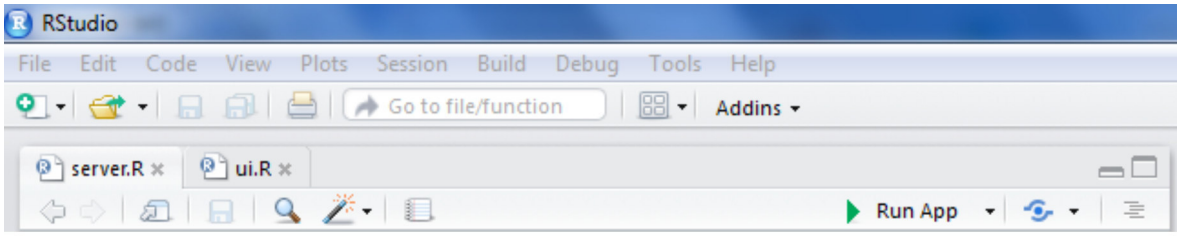


Ilustración 20: Archivos RStudio

server.R

En este archivo se incluye la secuencia de comandos que necesita el servidor, es decir, las instrucciones que el equipo precisa para construir la aplicación. En él se plantean los objetos necesarios para la resolución del problema.

El código mínimo que debe contener es: shinyServer(function(input, output) {}).

ui.R

Este archivo contiene la secuencia de comandos de interfaz de usuario que controla el diseño y aspecto de la aplicación. En él se definen las variables de entrada que el usuario elegirá y los elementos de salida que se mostrarán en la página html.

El código mínimo que debe contener es: shinyUI(fluidPage()).

10.1.1.Archivo server.R

Lo primero es cargar todos los paquetes que se utilizan en este archivo. Esto se realiza a través de la función `library`, que es la que permite el cargado de todo tipo de paquetes.

En este caso los paquetes necesarios son: `shiny`, `TSP`, `rJava`, `xlsx` y `lpSolve`. Todos ellos explicados en el apartado *Paquetes de R necesarios.*

```
2  library(shiny)
3  library(TSP)
4  library(rJava)
5  library(xlsx)
6  library(lpSolve)
```

Ilustración 21: Paquetes server.R

Después de haber cargado los paquetes, se crean las instrucciones que hacen que la aplicación obtenga el resultado que se desea. Como se mencionó anteriormente se debe empezar con el código mínimo `shinyServer(function(input, output){})`, dentro del cual se encuentra el resto de código del archivo.

```
9 ▾ shinyServer(function(input, output) {
```

Ilustración 22: shinyServer

Luego se define `Distancias` que es la matriz que contiene todas las distancias entre las ciudades, las cuales pueden verse de forma clara en el apartado anteriormente nombrado *Matriz Principal.*

```
Distancias<-matrix(rbind(
  c(0,168,354,368,503,511,639,486,507,621,235,200,370,142,604,360,28
  c(168,0,291,534,688,524,787,652,673,640,248,385,555,308,617,351,44
  c(354,291,0,658,610,796,942,789,664,436,520,392,353,496,889,168,60
  c(368,534,658,0,328,731,416,263,238,615,531,250,504,276,809,526,16
  c(503,688,610,328,0,1026,691,538,93,328,726,311,265,547,1104,464,4
  c(511,524,796,731,1026,0,610,606,920,1116,280,695,865,541,103,888,
  c(639,787,942,416,691,610,0,158,601,978,611,598,788,552,687,811,39
  c(486,652,789,263,538,606,158,0,449,826,516,445,636,399,685,659,24
  c(507,673,664,238,93,920,601,449,0,382,694,272,265,440,997,518,357
  c(621,640,436,615,328,1116,978,826,382,0,839,448,260,684,1208,291,
  c(235,248,520,531,726,280,611,516,694,839,0,417,587,263,372,610,38
  c(200,385,392,250,311,695,598,445,272,448,417,0,192,267,786,259,25
```

Ilustración 23: Matriz Distancias

A este matriz se le añaden los nombres de las ciudades, tanto en fila como en columna.

```
attributes(Distancias)$dimnames[[1]]<-c("ALBACETE","ALICANTE","ALMERIA","AVILA","BADAJOZ","BARCELONA","BILBAO","BURGOS",
                                        "CACERES","CADIZ","CASTELLON","CIUDAD.REAL",
                                        "CORDOBA","CUENCA","GERONA","GRANADA","GUADALAJARA","HUELVA","HUESCA","JAEN",
                                        "LA.CORUNA","LEON","LERIDA","LOGRONO",
                                        "LUGO","MADRID","MALAGA","MERIDA",
                                        "MURCIA","ORENSE","OVIEDO","PALENCIA",
                                        "PAMPLONA","PONTEVEDRA","SALAMANCA","SAN.SEBASTIAN",
                                        "SANTANDER","SANTIAGO.DE.COMPOSTELA","SEGOVIA","SEVILLA",
                                        "SORIA","TARRAGONA","TERUEL","TOLEDO",
                                        "VALENCIA","VALLADOLID","VITORIA","ZAMORA",
                                        "ZARAGOZA")
```

Ilustración 24: Nombres Matriz Distancias 1

```
attributes(Distancias)$dimnames[[2]]<-c("ALBACETE","ALICANTE","ALMERIA","AVILA","BADAJOZ","BARCELONA","BILBAO","BURGOS",
                                        "CACERES","CADIZ","CASTELLON","CIUDAD.REAL",
                                        "CORDOBA","CUENCA","GERONA","GRANADA","GUADALAJARA","HUELVA","HUESCA","JAEN",
                                        "LA.CORUNA","LEON","LERIDA","LOGRONO",
                                        "LUGO","MADRID","MALAGA","MERIDA",
                                        "MURCIA","ORENSE","OVIEDO","PALENCIA",
                                        "PAMPLONA","PONTEVEDRA","SALAMANCA","SAN.SEBASTIAN",
                                        "SANTANDER","SANTIAGO.DE.COMPOSTELA","SEGOVIA","SEVILLA",
                                        "SORIA","TARRAGONA","TERUEL","TOLEDO",
                                        "VALENCIA","VALLADOLID","VITORIA","ZAMORA",
                                        "ZARAGOZA")
```

Ilustración 25: Nombres Matriz Distancias 2

Para continuar se hace referencia a la lectura del archivo de precios. Se realiza a través de la función `read.xlsx` y en caso de no haber un archivo cargado sale el mensaje `NULL`.

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

```
11  dataset=reactive({
12    infile=input$file1
13    if(is.null(infile))
14      return(NULL)
15    read.xlsx(infile$datapath,1,header=FALSE,colClasses=rep("numeric",10))
16  })
```

Ilustración 26: Dataset

Posteriormente se crea un objeto, llamado `RutaCompra`, que contiene todos los pasos para la obtención del resultado final. En este caso ese resultado es el nombre de las ciudades que forman parte de la ruta.

Además es importante indicar que se utiliza la función `reactive ()`, que ofrece shiny, para que se vuelva a calcular el resultado inmediatamente una vez cambiadas las variables de entradas.

```
19  #output$ruta
20    RutaCompra<-reactive({
```

Ilustración 27: outputRuta

A continuación se realizan una serie de funciones para dejar la matriz de distancias, como se desea, para que sea utilizada en la obtención del resultado.

Se define `Matrizkm`, que es una matriz 49x10, en la cual todas sus columnas son iguales debido a que se coge la columna que contiene las distancias entre la ciudad que el usuario escoge como origen y todas las ciudades que forman parte de la ruta. Esta columna se copia 10 veces originando dicha matriz.

```
21      Matrizkm<-matrix(rep(Distancias[,input$Origen],10),nrow=49)
```

Ilustración 28: Matrizkm

También se prepara la matriz de los precios de los productos para su uso, porque en el caso de que un producto no esté en una ciudad, su hueco vacio sea sustituido por un valor muy grande, 10000.Esto es muy necesario debido a que sino el programa consideraría esos huecos como cero y haría que el resultado que sacara la aplicación fuera erróneo.

```
22  A<-dataset()
23  FuncionCerof<-function(x,y){if(is.na(A[x,y])) 10000 else A[x,y]}
24  B<-as.data.frame(matrix(mapply(FuncionCerof,rep(seq(1,49),rep(10,49)),rep(seq(1,10),49)),byrow=T,ncol=10))
```

Ilustración 29: Matriz precios

Luego se define la matriz `MM`, la cual es la suma del coste de los kilómetros y el coste de los productos.

```
25      MM<-as.matrix(input$costekm*Matrizkm+B)
```

Ilustración 30: Matriz MM

A continuación se definen la función objetivo y las restricciones para resolver en problema `lp_Solve`.

La función objetivo son los términos de la matriz `MM` definida anteriormente.

```
26 objective.in<-as.numeric(c(MM[1,1],MM[2,1],MM[3,1],MM[4,1],MM[5,1], MM[6,1],MM[7,1],MM[8,1],MM[9,1
27    MM[1,2],MM[2,2],MM[3,2],MM[4,2],MM[5,2], MM[6,2],MM[7,2],MM[8,2],MM[9,2],MM[10,2],MM[11,2],MM[1
28       MM[1,3],MM[2,3],MM[3,3],MM[4,3],MM[5,3], MM[6,3],MM[7,3],MM[8,3],MM[9,3],MM[10,3],MM[11,3],MI
29       MM[1,4],MM[2,4],MM[3,4],MM[4,4],MM[5,4], MM[6,4],MM[7,4],MM[8,4],MM[9,4],MM[10,4],MM[11,4],MI
30       MM[1,5],MM[2,5],MM[3,5],MM[4,5],MM[5,5], MM[6,5],MM[7,5],MM[8,5],MM[9,5],MM[10,5],MM[11,5],MI
31       MM[1,6],MM[2,6],MM[3,6],MM[4,6],MM[5,6], MM[6,6],MM[7,6],MM[8,6],MM[9,6],MM[10,6],MM[11,6],MI
32       MM[1,7],MM[2,7],MM[3,7],MM[4,7],MM[5,7], MM[6,7],MM[7,7],MM[8,7],MM[9,7],MM[10,7],MM[11,7],MI
33       MM[1,8],MM[2,8],MM[3,8],MM[4,8],MM[5,8], MM[6,8],MM[7,8],MM[8,8],MM[9,8],MM[10,8],MM[11,8],MI
34       MM[1,9],MM[2,9],MM[3,9],MM[4,9],MM[5,9], MM[6,9],MM[7,9],MM[8,9],MM[9,9],MM[10,9],MM[11,9],MI
35       MM[1,10],MM[2,10],MM[3,10],MM[4,10],MM[5,10], MM[6,10],MM[7,10],MM[8,10],MM[9,10],MM[10,10],
```

Ilustración 31: objective.in

Las restricciones, en el caso de este problema, son que cada producto solo se tiene que comprar una vez.

```
37    const.mat<-rbind(rep(c(1,0),c(49,49*9)),
38                     c(rep(0,49),rep(1,49),rep(0,49*8)),
39                     c(rep(0,49*2),rep(1,49),rep(0,49*7)),
40                     c(rep(0,49*3),rep(1,49),rep(0,49*6)),
41                     c(rep(0,49*4),rep(1,49),rep(0,49*5)),
42                     c(rep(0,49*5),rep(1,49),rep(0,49*4)),
43                     c(rep(0,49*6),rep(1,49),rep(0,49*3)),
44                     c(rep(0,49*7),rep(1,49),rep(0,49*2)),
45                     c(rep(0,49*8),rep(1,49),rep(0,49*1)),
46                     c(rep(0,49*9),rep(1,49),rep(0,49*0)))
47
48    const.dir<-c("=","=","=","=","=","=","=","=","=","=")
49    const.rhs<-c(1,1,1,1,1,1,1,1,1,1)
```

Ilustración 32: const.mat

Una vez definidos los objetos necesarios se resuelve el problema minimizando el coste total.

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

```
50   sol<-lp (direction = "min", objective.in=objective.in,const.mat=const.ma
51            const.dir=const.dir, const.rhs=const.rhs)
52
53   sol$solution
```

Ilustración 33: sol lp

Después de obtener la solución, la cual se obtiene con ceros y unos, se modifica para que salgan los nombres de las ciudades correspondientes.

```
55   CiudadesReparto1<-which(sol$solution=="1")
56   CiudadesReparto<-which(sol$solution=="1")-c(0,49,49*2,49*3,49*4,49*5,49*6,49*7,49*8,49*9)
57   CiudadesrutaP<-attributes(Distancias)$dimnames[[2]][CiudadesReparto]
58   Ciudadesruta<-as.vector(labels(table(CiudadesrutaP))$CiudadesrutaP)
```

Ilustración 34: Ciudadesruta

Puede existir el problema de que la ciudad elegida como *Origen* aparezca dos veces en la ruta debido a que también puede ser seleccionada en la ruta de compra. Por ello se crea una función que capaz de distinguir si la ciudad *Origen* se encuentra o no.

```
62 - if(input$Origen%in%Ciudadesruta==TRUE){
63     Matriz<-Distancias[Ciudadesruta,Ciudadesruta]}
64 - else { Matriz<-Distancias[c(input$Origen,Ciudadesruta),
65                              c(input$Origen,Ciudadesruta)]}
```

Ilustración 35: Duplicidad Ciudad Origen

También puede ocurrir que la compra de todos los productos se realice en la ciudad de origen y en la ruta de compra solo aparecería su nombre.

```
69   if (length(Ciudadesruta)<2)   paste(input$Origen)
70 - else {
71   CiudadesRepartoTSP<-TSP(Matriz, labels = NULL)
```

Ilustración 36: Ciudad Origen unica en la ruta

Se definen los objetos que hacen referencia a la ciudad *Origen* y al algoritmo, que el usuario elige.

```
74    #Se define ciudad de origen
75    initialtour<-as.integer(which(labels(Matriz)[[1]]==input$Origen[1]))
76
77    #Algoritmo
78    method1<-input$Method
```

Ilustración 37: Objetos Ciudad Origen y Algoritmo

Como la solución debe ser un ciclo hamiltoniano, es necesario incluir una ciudad que actúe como corte y se encuentre a distancia nula del resto de las ciudades de la matriz. Esta función se realiza mediante `insert_dummy`.

```
80    #Hay que insertar la dummy city para hacer el tour hamiltoniano
81    CRtsp <- insert_dummy(CiudadesRepartoTSP, label = "cut")
```

Ilustración 38: dummy city

El hecho de llamar cut a esta ciudad se debe a que ejerce como ciudad de corte y permite optimizar el problema dividiéndolo en racimos y uniendo estos entre sí, tal y como ya hicieran en su día Dantzig, Fulkerson y Johnson. A este hecho se le llama técnica de ramificación y poda. La técnica de ramificación y poda se suele interpretar como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores

(y a la que debe su nombre) es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para «podar» esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.

Teniendo ya todos los ingredientes preparados, se puede obtener las ciudades que forman parte de la ruta, es decir, nuestra solución.

La función `labels()` permite la obtención de los nombres de las ciudades por orden en las que deben visitarse pero debe tenerse en cuenta que existe la ciudad `cut` creada para obtener el ciclo hamiltoniano. Por esta razón debe programarse para que esta ciudad no aparezca en la solución que se muestra.

```
83    #Resultado
84    results1<-solve_TSP((CRtsp),method1,control=list(start=initialtour))
85
86    #función labels ofrece los nombres
87    labels(results1)[labels(results1)!="cut"]
```

Ilustración 39: Resultado nombres de las ciudades

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

El siguiente objeto creado es `LongitudRuta` que es el que permite obtener los kilómetros que tiene la ruta de compra. El código es similar al utilizado para calcular el objeto `RutaCompra` salvo por dos modificaciones.

```
92  #output$distancia
93    LongitudRuta<-reactive({
```

Ilustración 40: outputdistancia

La primera modificación hace referencia a cuando la compra de todos los productos se realice en la ciudad de origen. Si se da esta situación en el campo de *Longitud de la ruta* tiene que salir cero.

```
137    if(length(Ciudadesruta)<2)
138      (0)
139    else {
```

Ilustración 41: Longitud de la ruta igual a cero

La segundo modificación tiene que ver con que este caso se quiere calcular la longitud de la ruta mientras que en el anterior se obtenía la ruta de reparto. Para conseguir la longitud se utiliza la función `tour_length()`.

```
157    #Valor de la longitud del tour
158    V<-as.integer(results1)
159    tour_length(CRtsp, V)
```

Ilustración 42: tour_length

Como se ha nombrado anteriormente también se quiere saber qué producto se compra en cada ciudad. Para ello se crea el objeto `CiudadesProducto`.

```
164  #CiudadesProducto
165  CiudadesProducto<-reactive({
```

Ilustración 43: outputCiudadesProducto

En este caso se utiliza solo la primera parte de los códigos anteriores, hasta el objeto `CiudadesRuta` que es sustituido por `Ciudades` donde se cogen los nombres de las ciudades.

```
202    Ciudades<-attributes(Distancias)$dimnames[[2]][CiudadesReparto]
203    Ciudades
```

Ilustración 44: Ciudades cada producto

Una vez conseguido que los objetos `CiudadesProducto` `LongitudRuta` y `RutaCompra` contengan la información que se quiere, esta tiene que ser llevada al archivo *ui.R*. Para ello se crean salidas que serán llamadas por *ui.R* para sacar la información por pantalla. Estas salidas son las siguientes:

- `output$DatosCostes`

```
208 output$DatosCostes<-renderTable({TablaCostes<-dataset()
209 Costes<-c("ALBACETE","ALICANTE","ALMERIA","AVILA","BADAJOZ","BARCELONA","BILBAO","BURGOS",
210           "CACERES","CADIZ","CASTELLON","CIUDAD.REAL",
211           "CORDOBA","CUENCA","GERONA","GRANADA","GUADALAJARA","HUELVA","HUESCA","JAEN",
212           "LA.CORUNA","LEON","LERIDA","LOGRONO",
213           "LUGO","MADRID","MALAGA","MERIDA",
214           "MURCIA","ORENSE","OVIEDO","PALENCIA",
215           "PAMPLONA","PONTEVEDRA","SALAMANCA","SAN.SEBASTIAN",
216           "SANTANDER","SANTIAGO.DE.COMPOSTELA","SEGOVIA","SEVILLA",
217           "SORIA","TARRAGONA","TERUEL","TOLEDO",
218           "VALENCIA","VALLADOLID","VITORIA","ZAMORA",
219           "ZARAGOZA")
220 dimnames(TablaCostes)[[1]]<-Costes
221 TablaCostes})
```

Ilustración 45: outputDatosCostes

- `output$CiudadesConProducto`

```
224 output$CiudadesConProducto<-renderTable({
225         rbind(CiudadesProducto())
226     })
```

Ilustración 46: outputCiudadesConProducto

- `output$RutaDeCompra`

```
228 output$RutaDeCompra<- renderTable({
229    Tabla<-data.frame(RutaCompra())
230    names(Tabla)<-"Ruta de compra"
231    Tabla
232   })
```

Ilustración 47: outputRutaDeCompra

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

- `output$LongitudDeLaRuta`

```
235 output$LongitudDeLaRuta<- renderPrint({
236   LongitudRuta()
237 })
```

Ilustración 48: outputLongitudDeLaRuta

Las funciones `renderTable`,`renderPrint` y `renderText` son diferentes formas de generar outputs según el tipo de lo que se desea sacar.

10.1.2.Archivo ui.R

Al igual que en el archivo *server.R*, el primer paso es cargar el paquete `shiny` que permite la creación de la aplicación.

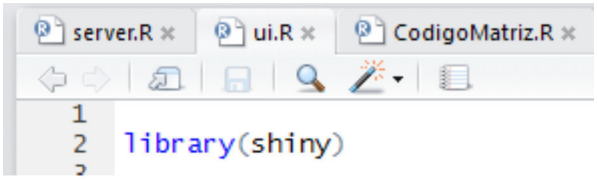


Ilustración 49: Paquetes ui.R

Una vez cargado el paquete *shiny* ya se puede comenzar la construcción del diseño y aspecto de la aplicación. Al igual que el archivo *server. R*, este archivo debe empezar con el código mínimo `shinyUI(fluidPage())`, dentro del cual se encuentra el resto de código del archivo.

```
4 shinyUI(fluidPage(
```

Ilustración 50: shinyUI

La aplicación da comienzo con el título, que en este caso es una imagen con el logo y nombre de la app.

```
6   # Titulo de la aplicacion
7   titlePanel(img(src="SY.png")),
```

Ilustración 51: Título de la aplicación

A continuación se introducen los campos que el cliente tiene que elegir para obtener la ruta que desea. Estos elementos se encuentran en un fondo de color gris y se ordenan de izquierda a derecha y de arriba a abajo.

```
 9    # SidebarPanel con los inputs
10    sidebarLayout(
11      inputPanel(
```

Ilustración 52: Panel con inputs

Lo primero es crear un control de carga de archivos con el cual el usuario cargara el archivo con la lista de precios de los productos. Este apartado tiene el título *Introducir archivos de precios.*

```
13        fileInput('file1', "Introducir archivos de precios",
14                  accept=c('sheetName','header'),multiple=FALSE),
```

Ilustración 53: input archivo de precios

El segundo input es el *Coste de transporte por kilometro*. Se establece un valor predeterminado de 0.05 y se fija que el mínimo es 0 y el máximo 100. Se introduce de la siguiente forma

```
16        numericInput("costekm","Coste transporte por km",min=0,max=100,value=0.05),
17
```

Ilustración 54: input coste kilometro

A continuación se crea una lista con el nombre de todas las ciudades para que el usuario seleccione el origen de la ruta. Es una barra despegable donde aparecen las ciudades en orden alfabético

```
19  selectInput("Origen",label="Origen Ruta",
20              choices=c("ALBACETE","ALICANTE","ALMERÍA"="ALMERIA","ÁVILA"="AVILA",
21                        "BADAJOZ","BARCELONA","BILBAO","BURGOS",
22                        "CÁCERES"="CACERES","CÁDIZ"="CADIZ","CASTELLÓN"="CASTELLON","CIUDAD REAL"="CIUDAD.REAL",
23                        "CÓRDOBA"="CORDOBA","CUENCA","GERONA","GRANADA",
24                        "GUADALAJARA","HUELVA","HUESCA","JAÉN"="JAEN",
25                        "LA CORUÑA"="LA.CORUNA","LEÓN"="LEON","LÉRIDA"="LERIDA","LOGROÑO"="LOGRONO",
26                        "LUGO","MADRID","MÁLAGA"="MALAGA","MÉRIDA"="MERIDA",
27                        "MURCIA","ORENSE","OVIEDO","PALENCIA",
28                        "PAMPLONA","PONTEVEDRA","SALAMANCA","SAN SEBASTIÁN"="SAN.SEBASTIAN",
29                        "SANTANDER","SANTIAGO DE COMPOSTELA"="SANTIAGO.DE.COMPOSTELA","SEGOVIA","SEVILLA",
30                        "SORIA","TARRAGONA","TERUEL","TOLEDO",
31                        "VALENCIA","VALLADOLID","VITORIA","ZAMORA",
32                        "ZARAGOZA"),
33                        selected="MADRID"),
```

Ilustración 55: input origen ruta

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Como se observa al final del código aparece el término `selected` que permite fijar por defecto la ciudad de origen. En este caso se establece que sea Madrid debido a que es la capital del país.

El último input permite seleccionar el algoritmo con el que se quiere resolver el problema y al igual que el anterior, la selección es a través de una barra despegable. Los algoritmos son los siguientes:

- `nearest_insertion`
- `farthest_insertion`
- `cheapest_insertion`
- `arbitrary_insertion`

Se establece como predeterminado el algoritmo `nearest_insertion` debido a que es el que *TSP* propone como tal.

```
37        selectInput("Method",label="Algoritmo",
38                    choices=c("nearest_insertion", "farthest_insertion",
39                    "cheapest_insertion", "arbitrary_insertion"
40                                        ),selected="nearest_insertion"),
41
```

Ilustración 56: input algoritmo

El código de los elementos de entrada se cierra con `width` que determina la anchura de la entrada. El valor es 12 que se ajusta al ancho de una ventana html.

```
43          width=12
44        ),
```

Ilustración 57: width

Una vez programadas las entradas se sigue con la creación de las outputs o salidas que se quiere que aparezcan en la aplicación y todas ellas deben estar incluidas dentro de `mainPanel()` que está destinado a esa función. Dentro de `mainPanel` se utiliza `tabsetPanel` y `tabPanel` para generar diferentes pestañas dentro de la aplicación.

```
47        #mainpanel con los outputs
48        mainPanel(
49          tabsetPanel(
50           tabPanel(
```

Ilustración 58: Panel con outputs

La primera pestaña recibe el nombre de *Introducción* y en ella se da la bienvenida al usuario y se le dan las instrucciones necesarias para que utilice la aplicación de forma correcta.

```
50 tabPanel(strong(p(span("Introducción"))),
51         h3(strong(span("Bienvenido a Purchaser App",style= "color:blue"))),
52         h4(span("Esta aplicación te permite obtener, de forma rápida y simple ,la mejor ruta para
53                 comprar los productos que necesitas.")),
54         h4(span("Solo tienes que seguir estos sencillos pasos:")),
55         h5(span("- Cargar el fichero xls con los precios de los productos.La tabla debe ser
56                 49x10(CiudadesxProductos),todos datos deben ser números y en caso de no haber
57                 producto en alguna ciudad dejar un espacio en blanco. ")),
58         h5(span("- Introducir el coste por kilometro del transporte utilizado .")),
59         h5(span("- Seleccionar la ciudad origen de la ruta.")),
60         h5(span("- Elegir el algoritmo con el que se desea obtener la ruta óptima. ")),
61         h6(em("**nearest_insertion. "),"En cada paso elige la ciudad más cercana a una ciudad de
62             la ruta. "),
63         h6(em("**farthest_insertion."),"En cada paso elige la ciudad más lejana a una ciudad de
64             la ruta. "),
65         h6(em("**cheapest_insertion. "),"En cada paso elige la ciudad que hace minimo el incremento
66             de la ruta. "),
67         h6(em("**arbitrary_insertion. "),"En cada paso elige la ciudad arbitrariamente entre las
68             que quedan por incluir en la ruta."),
69         h4(span("Una vez eligidos los campos anteriores observar la  ruta  y su longitud en
70                 la pestaña",strong(" Ruta de compra",style= "color:blue")))),
```

Ilustración 59: output introducción

A la segunda pestaña se le pone el nombre de *Matriz de Costes* y en ella simplemente se presenta la lista de precios de los productos que el usuario ha cargado. Con la función `tableOutput` se llama a un `renderTable` del archivo *server.R*, en este caso *DatosCostes*.

```
74      tabPanel(strong(p(span("Matriz de costes"))),tableOutput("DatosCostes")),
```

Ilustración 60: output matriz de costes

Por último, la tercera pestaña tiene el nombre de *Ruta de compra* y en ella se ofrecen las ciudades donde se compra cada producto, la longitud de la ruta, el orden de la ruta y un mapa de España. Se utilizan `verbatimTextOutput` que permite el procesamiento de una variable de salida como texto y `tableOutput` que ya se ha descrito en el apartado anterior. La última parte del código es:

```
77      tabPanel(strong(p("Ruta de compra")),
78               h5(strong(span("Ciudades de compra del producto"))),
79               tableOutput("CiudadesConProducto"),
80               h5(strong(span("Longitud de la ruta (Km)"))),
81               verbatimTextOutput("LongitudDeLaRuta"),
82               tableOutput("RutaDeCompra"),
83               h5(strong(span("Mapa de España"))),
84               img(src="RA.png"))
```

Ilustración 61: input ruta de compra

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

Por último es necesario describir algunas funciones, sobretodo enfocadas a editar el tamaño y formato del texto, algunas utilizadas en el código de *ui.R*.

Código	Función
p	Un párrafo de texto
h1	Un primer nivel de cabecera
h2	Un segundo nivel de cabecera
h3	Un tercer nivel de cabecera
h4	Un cuarto nivel de cabecera
h5	Un quinto nivel de cabecera
h6	Un sexto nivel de cabecera
a	Un hipervínculo
br	Un salto de línea(línea en blanco)
div	Una división del texto con un estilo uniforme
span	Una división en la línea de texto con un estilo uniforme
pre	Texto, tal cual, en una fuente de ancho fijo
code	Un bloque de código con formato
img	Una imagen
strong	Texto en negrita
em	Texto en cursiva

Tabla 16: Funciones tamaño y formato de texto

11. INTERFAZ DE LA APLICACIÓN

La aplicación recibe el nombre de *Purcharser App* y tiene un logo ligado a él. Ambos intentan representar la finalidad que tiene dicha aplicación.

Ilustración 62: Logotipo app

Su diseño y apariencia están orientados a que al usuario le resulte fácil de manejar y atractiva a la vista. Esta aplicación se puede dividir en dos partes:

- Inputs o entradas: Datos que el usuario introduce según sus necesidades.
- Outputs o salidas: Resultados obtenidos conforme a los inputs introducidos.

A continuación se describe el contenido de estas dos partes.

11.1. INPUTS

11.1.1. Introducir archivos de precios

Consiste en un botón que permite al usuario cargar el archivo con los precios de los productos que quiere comprar. Una vez cargado aparece el nombre del archivo para comprobar que es el deseado.

Ilustración 63: Introducir archivo de precios

11.1.2.Coste transporte por kilometro

Como el propio nombre indica su función es indiciar el precio por kilometro del transporte que va a utilizar el usuario.

Ilustración 64: Coste transporte por kilometro

11.1.3.Origen Ruta

Es un despegable que posibilita elegir al usuario la ciudad donde quiere comenzar la ruta de compra. Las ciudades se presentan ordenadas alfabéticamente.

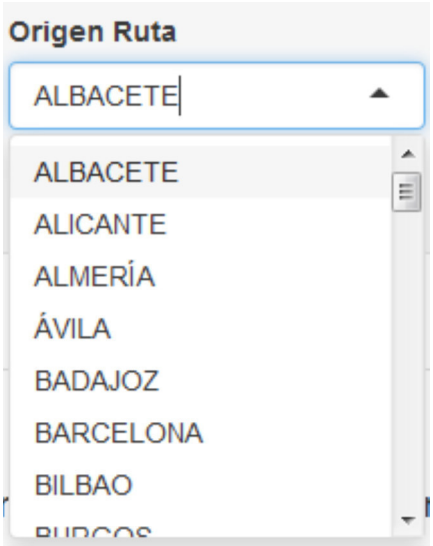


Ilustración 65: Origen Ruta

11.1.4.Algoritmo

Consiste en un despegable que permite seleccionar el algoritmo que el usuario quiere utilizar en la obtención de la solución.

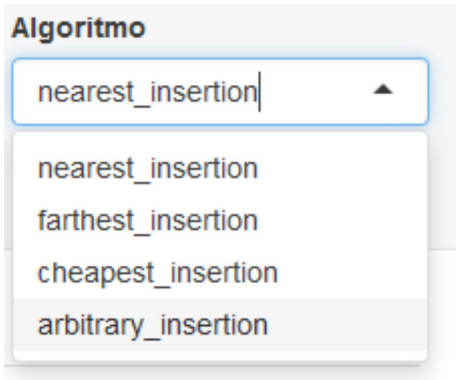


Ilustración 66: Algoritmo

A continuación se muestra una imagen completa de la zona de los inputs.

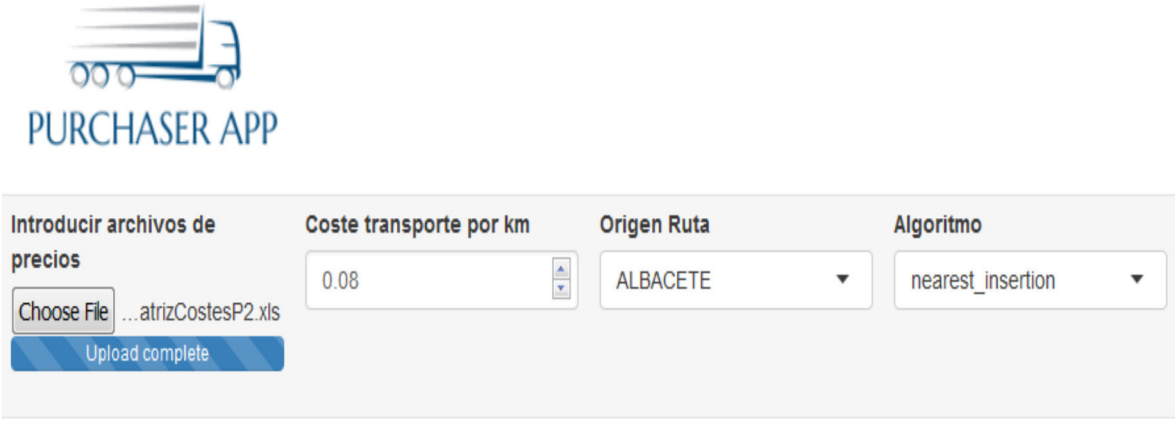


Ilustración 67: Zona inputs

11.2. OUTPUTS

Esta parte se divide en tres pestañas: Introducción, Matriz de Costes y Ruta de Compra.

11.2.1.Introducción

En esta pestaña se da la bienvenida al usuario, se describe la finalidad de la aplicación y se dan las instrucciones para su utilización.

Ilustración 68: Introducción

11.2.2.Matriz de costes

La función de esta pestaña es simplemente mostrar la lista de precios de los productos que el usuario ha cargado.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
ALBACETE	2049.00	2047.00	2011.00	1997.00	2037.00	1960.00	1952.00	2020.00	1953.00	2003.00
ALICANTE	2082.00	2021.00	2030.00	1980.00	2023.00	1987.00	2029.00	2014.00	2031.00	2021.00
ALMERIA	1932.00	1961.00	2070.00	2003.00	1979.00	1997.00	1950.00	2036.00	1949.00	1984.00
AVILA	2016.00	1966.00	2014.00	1959.00	1927.00	1988.00	2067.00	2004.00	1966.00	2015.00
BADAJOZ	1979.00	1987.00	1981.00	1949.00	2101.00	1965.00	1927.00	2054.00	2053.00	1966.00
BARCELONA	2108.00	2069.00	2021.00	1946.00	1899.00	2028.00	2007.00	2046.00	1947.00	2073.00
BILBAO	1962.00	1994.00	2011.00	1942.00	1989.00	2061.00	1971.00	1961.00	1970.00	2018.00
BURGOS	1965.00	1965.00	2028.00	2012.00	1991.00	1970.00	2047.00	2055.00	2029.00	2061.00
CACERES	2039.00	1994.00	1976.00	1962.00	2043.00	1962.00	1955.00	2040.00	1950.00	1936.00
CADIZ	1932.00	1984.00	2021.00	2091.00	2020.00	2014.00	2050.00	1898.00	2007.00	2010.00
CASTELLON	1962.00	2067.00	2029.00	2002.00	2058.00	1962.00	2023.00	1958.00	1947.00	2036.00
CIUDAD.REAL	2045.00	2026.00	1995.00	2051.00	2021.00	1994.00	1968.00	2049.00	1967.00	1968.00

Ilustración 69: Matriz de costes

11.2.3.Ruta de compra

Esta última pestaña es la más importante debido a que es donde se muestran los datos importantes para el usuario. Estos datos son:

- Ciudades de compra del producto
- Longitud de la ruta(Km)
- Ruta de compra

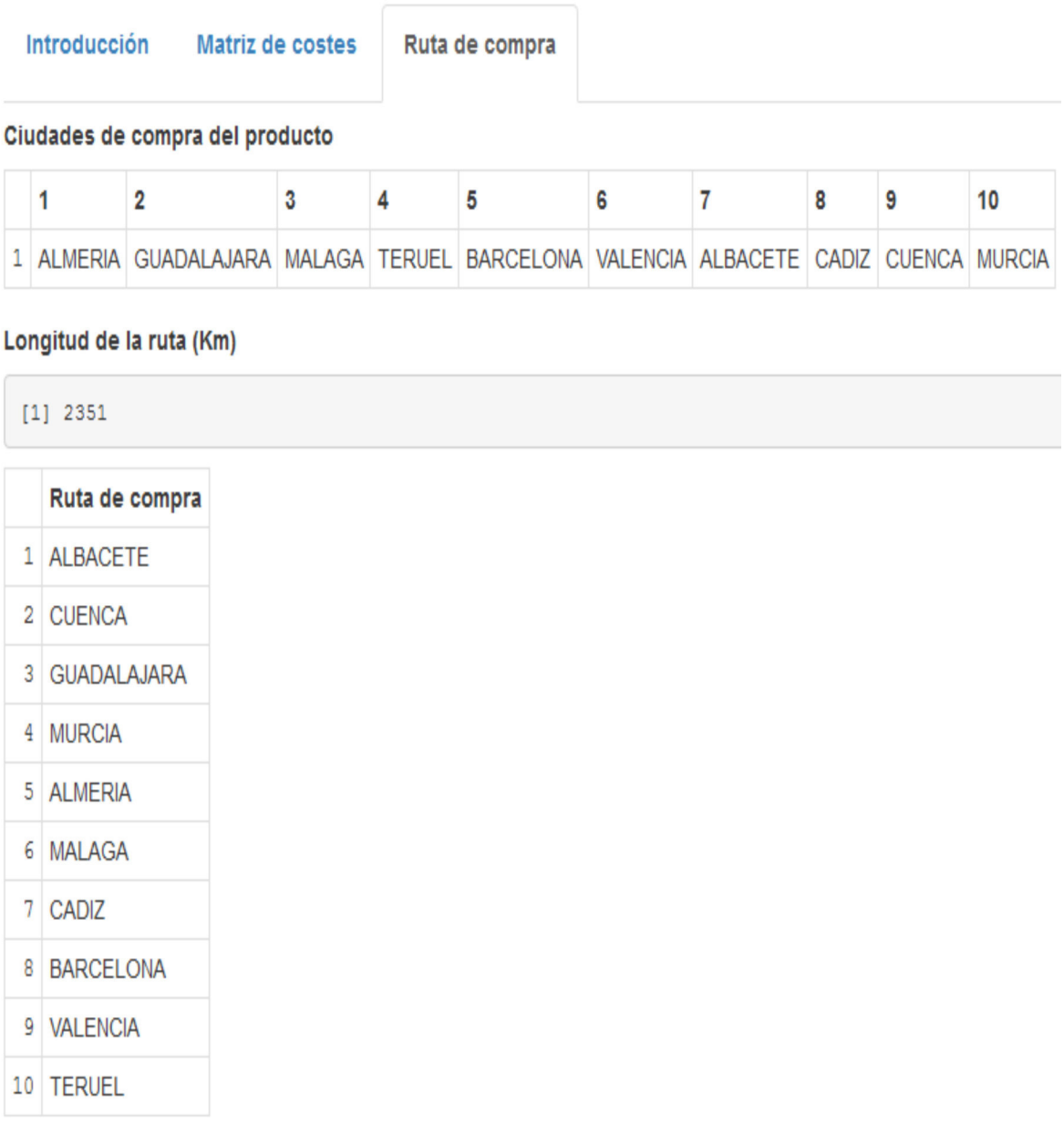


Ilustración 70: Ruta de compra

11.2.4.Información complementaria

Como información complementaria, al final de esta pestaña, se añade un mapa de España con las capitales de provincia y comunidad autónoma.

Ilustración 71: Mapa de España

12. CONCLUSIONES

En este TFG se ha planteado el problema del viajante comprador (TTP, Travelling Purchaser Problem) que es una generalización más compleja del problema del viajero (TSP, Travelling Salesman Problem).

Después de haber profundizado en ellos se puede concluir que los problemas de optimización de rutas siguen siendo un desafío debido a que aun no se es capaz de encontrar la solución óptima a dichos problemas, a pesar de disponer de algoritmos que intentan aproximarse lo máximo posible a esa solución ideal. De hecho estos algoritmos sujetos a los mismos datos dan soluciones distintas cada vez.

Realizar este trabajo ha posibilitado conocer más a fondo el universo de la programación en R, tanto en R-Project como en RStudio, que ofrece una amplia gama de paquetes, cada uno proporcionado una tarea específica dentro de la estadística. Actualmente existen 7234 paquetes fáciles de encontrar y comprender su funcionamiento.

La aplicación que se ha realizado puede ser utilizada por cualquier empresa o particular que necesite ir a comprar los productos que necesita por el territorio peninsular español, de forma rápida y sencilla. Ofrece al usuario la posibilidad de conocer las ciudades donde comprar cada producto, el orden para correr dichas ciudades y la longitud de la ruta simplemente contemplado varios campos.

Concretamente esta aplicación ofrece al usuario la posibilidad de comprar diez productos en 49 ciudades que son las capitales de provincia y de comunidad autónomas. En un futuro se podría mejorar o ampliar la aplicación aumentado la cantidad de productos que se pueden cargar mediante algunos cambios en el código u ofreciendo otras ciudades donde comprar los productos cambiando la matriz de distancias.

Autor: **Alejandro Sarabia Hernando**
Nº TFG: **425.16.108**

13. BIBLIOGRAFÍA

Distancia entre dos ciudades. (s. f.). Recuperado 17 de junio de 2016, a partir de http://www.distanciasentreciudades.com/

lpSolve.pdf. (s. f.). Recuperado a partir de https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf

Palazzesi, A. (2010, agosto 19). El algoritmo voraz. Recuperado 17 de junio de 2016, a partir de http://www.neoteo.com/el-algoritmo-voraz

Problema del agente viajero. (s. f.). Recuperado a partir de http://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n3/e5.html

Problema del viajante. (2016, mayo 21). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=Problema_del_viajante&oldid=91211119

R: Create graphs from adjacency matrices. (s. f.). Recuperado 17 de junio de 2016, a partir de http://finzi.psych.upenn.edu/library/igraph0/html/graph.adjacency.html

R (lenguaje de programación). (2016, junio 2). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=R_(lenguaje_de_programaci%C3%B3n)&oldid=91444799

R Package Snapshot. (s. f.-a). Recuperado a partir de https://cran.r-project.org/web/packages/lpSolve/index.html

R Package Snapshot. (s. f.-b). Recuperado a partir de https://cran.r-project.org/web/packages/TSP/index.html

R Package Snapshot. (s. f.-c). Recuperado a partir de https://cran.r-project.org/web/packages/rJava/index.html

R Package Snapshot. (s. f.-d). Recuperado a partir de https://cran.r-project.org/web/packages/xlsx/index.html

R Package Snapshot. (s. f.-e). Recuperado a partir de https://cran.r-project.org/web/packages/shiny/index.html

R Package Snapshot. (s. f.-f). Recuperado a partir de https://cran.r-project.org/web/packages/igraph/index.html

RStudio. (2015, diciembre 6). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=RStudio&oldid=87601961

R: TSP solver interface. (s. f.). Recuperado 17 de junio de 2016, a partir de http://127.0.0.1:18726/library/TSP/html/solve_TSP.html

Shiny: crear una aplicación web interactiva (apps) desde R. | R blogs / lang. (s. f.). Recuperado a partir de http://www.r-bloggers.com/lang/uncategorized/1459

Stockdale_Lorena.pdf. (s. f.). Recuperado a partir de http://cms.dm.uba.ar/academico/carreras/licenciatura/tesis/2011/Stockdale_Lorena.pdf

Relación de documentos

(X) Memoria .....80 páginas
(_) Anexos .....27 páginas

La Almunia, a 29 de Junio de 2016

Firmado: Alejandro Sarabia Hernando