IES Durango BHI

2026 AVATAR Euskadi
Datuen Herrian Estatistika Aurkitzen

24. KORRIKAren ibilbidearen diseinua TSP probleman oinarrituta: programazio linealaren bidezko optimizazioa

Egileak:

Naroa Aranburu Rementeria
Ander Asensio García
Goiatz Gaztelurrutia Simon
Ane Gonzalez Gil
Izan Saiz Zorrilla

Tutorea:

Unai Mendiguren Martin

2026ko apirilean

Laburpena

Proiektu honetan Korrikaren ibilbidea ikuspegi matematikotik aztertuko dugu. Bi urtean behin, Korrikak Euskal Herri osoa zeharkatzen du euskararen alde, 200 herri eta hiri baino gehiagotik igaroz. Honek erronka interesgarri bat planteatzen du: nola lor daiteke herri kopuru jakin bat lotzen duen ibilbidea ahalik eta laburrena izatea? Optimoa al da Korrikaren ibilbide ofiziala? Eta, hala ez bada, ba al dago hori hobetzeko modurik?

IKERKUNTZA OPERATIBOKO lan honetan, oinarri teoriko gisa hartuko dugu TRAVELLING SALESMAN PROBLEM (TSP) deritzon problema, puntu multzo bat bide laburrenaren bidez lotzea helburu duena. Gure kasuak, baina, badu zailtasun gehigarri bat: puntuak (herriak) ez ditugu lerro zuzenekin lotuko, baizik eta benetako errepideekin. Horrek behartuko gaitu mapa-, nabigazio- eta geolokalizazio-APIak erabiltzen, GOOGLE MAPS bezalako zerbitzuek erabiltzen dituztenen antzekoak.

Erronka honi aurre egiteko R programazio-hizkuntza baliatuko dugu, datuak antolatzeko eta ibilbide posibleak kalkulatzeko. Helburua proposamen matematiko bat garatzea da, herri nagusi guztiak zeharkatzen dituen ibilbide eraginkor bat lortzeko. Azkenik, lortutako emaitza Korrikaren ibilbide ofizialarekin konparatuko dugu, bi proposamenen arteko diferentziak aztertzeko.

Hitz gakoak: Korrika, TSP, Programazio Lineala, HERE TECHNOLOGIES, Optimizazioa, R programazio-lengoaia.

Abstract

Zuentzako.

Keywords: Korrika, TSP, Linear Programming, HERE TECHNOLOGIES, Optimization, R programming-language.

Aurkibidea

Taulak

Irudiak

1. Marko teorikoa

1.1 Korrika

Korrika bi urterik behin Euskal Herrian zehar lasterka egiten den euskararen aldeko ekimen garrantzitsuenetakoa da. Pertsona helduen euskalduntzea eta alfabetatzea eginkizun daukan AEK erakundeak (Alfabetatze Euskalduntze Koordinakundeak) antolatzen du, euskararen aldeko kontzientzia pizteko eta euskaltegietako eguneroko lana indartzeko dirua biltzeko.

11 egun inguru behar izaten dira lurralde osoa zeharkatzeko. Urtero ibilbidea aldatu egiten da, baina 2.300 kilometro ingurukoa izaten da beti. Korrikaren hasieratik bukaeraraino, korrikalariek eskuz esku eta kilometroz kilometro lekuko bat eramaten dute helmugaraino, eta horren barruan gordetzen da Korrikaren amaieran irakurriko den euskaldun ezagun batek idatziriko mezua.

Lehenengo Korrika 1980. urtean burutu zen, diktadura frankista amaitu eta, urte gutxitara, euskaldunak euskaraz alfabetatzeko beharrezkoa zen dirua biltzeko xedea zuen ekimena sortu zen: Korrika. Izan ere, frankismoak erabat zapaldu zuen euskaldun nortasuna.

Aurten, 2026ean, Korrikaren 24. edizioa martxoaren 19tik 29ra kokatuko da, eta Atharratzetik (Zuberoatik, alegia) Bilbora joango da. Izan ere, Korrikako lasterketetako irteera eta helmuga gisa hiri desberdinak izendatzen dira urtetik urtera.

Lan honen helburua ahalik eta herri gehien bilduko dituen ibilbidea sortzea da, 11 eguneko eta 2300 kilometroko bidea osatuz. Bidea Atharratzen hasi eta Bilbo amaituko da.

Hauek dira aurreko urteetako Korrikaren azken hiru edizioak (ibilbideak beti desberdin):

Irudia 1: Korrikaren azken 3 edizioetako ibilbideak

Edizio guztietan Korrikak Euskal Herriko hiri eta herri nagusienak zeharkatzen ditu. Normalean, 3.000 biztanletik gorako herri guztietatik igaro egiten da, baina edizio batzuetan munta handiko herri batzuetatik ez da igaro izan.

Esate baterako 22. edizioa ez zen Oñatitik (11.500 biztanle) edo Legazpitik (8.400 biztanle) igaro, ezta Urduñatik (4.200 biztanle) edo Orozkotik (2.700 biztanle) ere.

Irudia 2: Korrikaren 22. edizioaren ibilbideen zatiak

Ibilbidea optimizatuz gero, posible litzateke herri-kopuru gehiago barne hartzea ibilbidearen kilometro kopurua mantenduz? Horixe izango da, hain zuzen ere, gure lanaren helburua.

1.2 Programazio Lineala eta Programazio Osoa

Programazio lineala optimizazio problema matematiko bat da, horren bidez, helburu funtzio lineal bat minimizatu edo maximizatu egiten da aldagai batzuen menpe baldintza linealez osatutako murriztapen sistema baten barne. Baldintzak grafiko batean irudikatzean, amankomunean daukaten eremuari sistemaren soluzio-eremua edo eskualde egingarria deritzo, eta bertan baldintza guztiak betetzen dira eta inekuazio sistemaren soluzio posible guztiak daude.

Behin soluzio eremua jakinda helburu funtzioa minimizatzeko edo maximizatzeko eskualde egingarriaren erpinak kokatu behar dira, bertan aurkitzen baitira maximoak eta minimoak. Puntu horiek helburu-funtzioan ordezkatuz jakingo da zein den soluzioa, hau da, emaitza optimoa (maximoa edo minimoa), hori da programazio linealaren helburua.

Programazio linealaren enuntziatu estandarra, honako hau da:

$$x_1, x_2, \dots, x_n \text{ aldagaiak izanik,}$$

$$\text{minimizatu } z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \quad \text{non:}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\le b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\le b_m \end{aligned}$$

$$x_k \ge 0 \quad k = 1, 2, \dots, n \text{ izanik}$$

Aldagaiak

Helburu-funtzioa (lineala)

Murriztapen-multzoa (lineala)

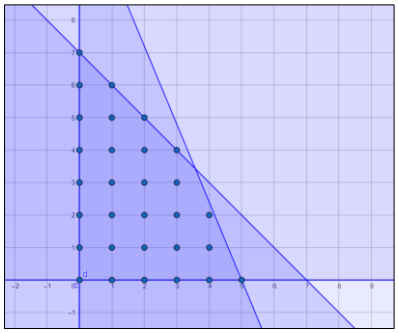
Aldagaien positibotasuna

Taula 1: Programazio Linealaren enuntziatu estandarra

z helburu-funtzioa minimizatu beharrean, maximizatu ere egin daiteke, eta murriztapen-multzoko inekuazioetan "≤" ikurra beharrean, "≥" edo "=" ikurrak ere jar daitezke.

Bi aldagaiko problemak ebazteko aurreko prozedura egokia da, metodo grafiko deritzo eta hau hiru aldagaiko problemak ebazteko erabil daitekeen arren (3 dimentsioko grafikoekin), ez da egokiena horretarako. Aldagai kopurua gehiago handitzen bada metodo desberdin bat erabiltzeko beharra agertzen da, SIMPLEX metodoa. Metodo grafikoa hiru aldagai baino gehiagoko problemetan ez denez egingarria eta gure proiektuan askoz ere aldagai gehiago daudenez, ordenagailua baliatuko dugu Programazio Linealeko problemak ebazteko.

Programazio Osoari dagokionez, programazio linealaren adar bat da, bere eginkizuna aldagaiak zenbaki oso izatera baldintzatzea da ($x_k \in \mathbb{N}$). Honen ondorioz, soluzio kopurua asko murrizten da, hau da, soluzio eremua diskretu izatera pasatzen da, area mugatu bat izatetik area berberan dauden puntu isolatu multzo bat bihurtzen da eskualde egingarria. Hala ere, nahiz eta soluzio-eremua "txikiagoa" izan, ebazpen-metodoa konplikatuagoa da, orain ez dagoelako bermatuta maximoa/minimoa eremuaren erpinetan egongo dela. Gu R programarekin lagunduko gara Programazio Osoko problemak ere ebazteko.



Irudia 3: Programazio Osoan soluzioen multzoa (finitua)

1.3 R programa

Lehen esan bezala, lan honetan aldagai asko izango ditugunez (herri asko izango ditugulako), ordenagailuarekin lagundu beharko gara derrigor. Guk R programa erabiliko dugu. R programatzeko software libre bat da, hau da, edonor erabil dezakeena eta doakoa.

1.3.1 Rko lpSolve liburutegia

Programazio Osoko problemak ebazteko `lpSolve` liburutegiko `lp(…)` funtzioa dago, zeinak, problemaren enuntziatu estandarretik abiatuta (Taula 1), argumentu hauek eskatzen dituen:

- `direction=`: Helburua zein den, *"max"* edo *"min"* idatziko dugu.
- `objective.in=`: Helburu-funtzioaren koefizienteak dituen bektorea idatziko dugu hemen.
- `const.mat=`: Inekuazio baldintzen ezkerreko koefizienteen matrizea idatziko dugu hemen, desberdindu gabe zeintzuk diren "≤", "≥" eta "=" modukoak.
- `const.dir=`: Inekuazioen norabideekin osaturiko bektorea idatziko dugu hemen.
- `const.rhs=`: Inekuazio baldintzen eskumako koefizienteen bektorea idatziko dugu.
- `int.vec=`: Osoak izan behar duten aldagaien zerrenda duen bektorea idatziko dugu.
- `all.int=`: `TRUE` aldagai guztiak osoak badira eta `FALSE` osoak ez badira.

Programazio Osoko problemak ebatz daitezke funtzio horrekin, baldin eta problema bera enuntziatu estandarraren bitartez adierazteko gauza bagara. Askotan zailagoa izaten da problema bera enuntziatzea, behin enuntziatuta dagoela ebaztea baino (ordenagailuak egiten duelako lan hori).

1.3.2 Rko TSP liburutegia

Zorionez, existitzen dira Programazio Linealeko problema konkretuak ebazteko liburutegi espezifikoak. Adibidez, guk TSP problemari helduko diogunez, existitzen da `TSP` izeneko liburutegia, espresuki sortua izan dena TSP problemak ebatzi ahal izateko (problema Programazio Linealeko enuntziatu estandarrean adierazi behar izan gabe). Funtzio honek asko lagunduko digu modu errazean soluzioak lortzen, hurrengo atalean (1.4 atala) ikusiko dugun moduan.

TSPko problemak ebazteko `TSP` liburutegiko `solve_TSP(…)` funtzioa erabiliko dugu, zeinak argumentu hauek eskatzen dituen:

- `distance_matrix=`: Grafoaren distantzia-matrizea TSP objektu gisa.
- `method=`: Zein metodo erabiliko den TSP problema ebasteko (*"nearest_insertion"*, *"cheapest_insertion"*, *"two_opt"*...)

1.4 Saltzaile Ibiltariaren Problema – Travelling Salesman Problem (TSP)

Saltzaile Ibiltariaren Problema (gaztelaniaz, *Problema del Vendedor Ambulante*, eta, ingelesez, *Travelling Salesman Problem - TSP*) optimizazio problema klasiko bat da, eta honetan oinarrituko gara lana aurrera eramateko. Problema honen helburua hiri multzo bat hartu eta haien arteko distantziak erabiliz herri guztietatik baten pasatzen den ahalik eta ibilbide laburrena ezartzea da. Problema honek aplikazio ugari ditu logistika edo elektronika bezalako arloetan.

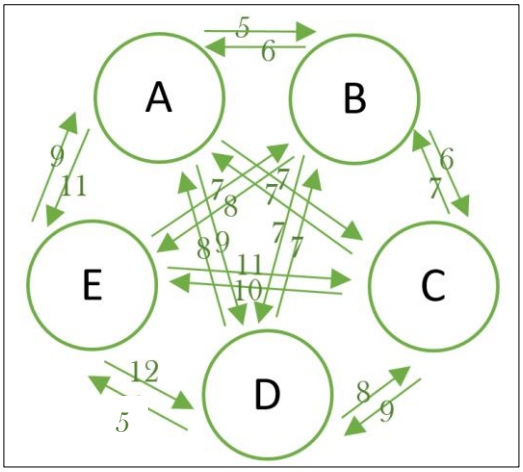
TSP probleman hiriak eta haien arteko kostuak osatzen duenari "grafo" deritzogu. Aldi berean, aurretiaz aipatu dugun bezala, grafoaren mutur bakoitzetik (hiri bakoitzetik) baten soilik igarotzen da. Mota honetako zikloei *"Hamiltondar Ziklo"* esaten zaie.

TSP problemak bitan banatu ditzakegu: simetrikoak eta ez simetrikoak. Problema simetrikoa bada, kostu berdina edukiko du Atik Bra joateak edo Btik Arako bidea egiteak. Problema ez simetrikoa bada, gure kasuan bezala, non hasi eta non bukatzeak kostuan eragingo du; izan ere, ez da berdina izango ibilbide berdina egitea ordena aldatuz gero.

Rko aurreko funtzioak erakusteko, hartuko dugu adibidetzat gure proiektua baino eskala txikiagoa den egoera bat. Adibide honetan bost herriren arteko distantziak emanda, bost herri horiek lotzen dituen ibilbiderik laburrena kalkulatuko dugu, TSP problema, alegia.

Ondoren, adibide honen informazioa adierazten dugu bi modu desberdinetan:

- Grafo baten bitartez
- Distantzia-matrizearen bitartez (`solve_TSP(…)` funtzioak behar duena, alegia)



Irudia 4: 5 hiriko grafo ez-simetrikoa

Distantzia matrizea	**A**	**B**	**C**	**D**	**E**
A	0	5	7	9	11
B	6	0	6	7	8
C	7	7	0	9	10
D	8	7	8	0	5
E	9	7	11	12	0

Taula 2: 5 hiriko grafo ez-simetrikoaren distantzia-matrizea

Honen bidez, zutabeko herri batetik aterata errenkadako herrira zenbateko distantzia behar den adierazi daiteke eta aldrebes. Ikusten den bezala, diagonala zeroz osatuta dago, Atik Ara ez dagoelako distantziarik. Matrize honetan aurretiaz adierazitako simetria eza ikusi daiteke, ez dagoelako distantzia bera Btik Ara eta Atik Bra.

Ebatz dezagun orain TSP problema hau bi modu desberdinetan:

1. Programazio Osoko enuntziatu orokorraren bidez, Rko `lpSolve` liburutegia erabilita (oso astuna eta nekeza).
2. TSP problema gisa, Rko `TSP` liburutegia erabilita (oso erraza eta praktikoa).

1.4.1 Programazio Osoko enuntziatu gisa (Rko lpSolve liburutegia)

Nola adierazi daitezke TSP problemaren baldintzak Programazio Linealaren enuntziatu estandarraren bidez? Zorionez, lan hau dagoeneko aztertuta eta ebatzita dago (Bibliografian [1]).

- Aldagai dikotomikoak: $x_{ij} = \begin{cases} 0, & i-tik\ j-rako\ bidea\ egiten\ ez\ bada \\ 1, & i-tik\ j-rako\ bidea\ egiten\ bada \end{cases}$
- Distantzia-matrizea: c_{ij} $(i-tik\ j-rako\ distantzia)$

x_{ij} aldagaiak (i jatorria, j helmuga)

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \quad non:$$

$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1$ (hiri bakoitzetik behin eta behin bakarrik atera)

$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1$ (hiri bakoitzera behin eta behin bakarrik sartu)

$x_{ij} \in \{0,1\}$ $(\Leftrightarrow x_{ij} \in \mathbb{N}\ eta\ x_{ij} \geq 0, minimizatzen\ gaudelako)$

Taula 3: TSPko problemak enuntziatu estandarren bidez formulatzeko modua

Enuntziatu estandarra gure adibide partikularrera egokituz:

$$Aldagaiak: x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{21}, \dots, x_{55}$$

$$\min z = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij} x_{ij} = 0x_{11} + 5x_{12} + 7x_{13} + \cdots + 12x_{45} + 0x_{55} \quad non:$$

Hiri bakoitzetik behin eta behin bakarrik ateratzeko:

$$\sum_{\substack{j=1 \\ j \neq i}}^{5} x_{1j} = 1 \Leftrightarrow x_{12} + x_{13} + x_{14} + x_{15} = 1 \text{ (lehen hiritik behin eta behin bakarrik atera)}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{5} x_{2j} = 1 \Leftrightarrow x_{21} + x_{23} + x_{24} + x_{25} = 1 \text{ (2. hiritik ...)}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{5} x_{3j} = 1 \Leftrightarrow x_{31} + x_{32} + x_{34} + x_{35} = 1 \text{ (3. hiritik ...)}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{5} x_{4j} = 1 \Leftrightarrow x_{41} + x_{42} + x_{43} + x_{45} = 1 \text{ (4. hiritik ...)}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{5} x_{5j} = 1 \Leftrightarrow x_{51} + x_{52} + x_{53} + x_{54} = 1 \text{ (5. hiritik ...)}$$

Hiri bakoitzera behin eta behin bakarrik heltzeko:

$$\sum_{\substack{i=1 \\ i \neq j}}^{5} x_{i1} = 1 \Leftrightarrow x_{21} + x_{31} + x_{41} + x_{51} = 1 \text{ (lehen hirira behin eta behin bakarrik sartu)}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{5} x_{i2} = 1 \Leftrightarrow x_{12} + x_{32} + x_{42} + x_{52} = 1 \text{ (2. hirira ...)}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{5} x_{i3} = 1 \Leftrightarrow x_{13} + x_{23} + x_{43} + x_{53} = 1 \text{ (3. hirira ...)}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{5} x_{i4} = 1 \Leftrightarrow x_{14} + x_{24} + x_{34} + x_{54} = 1 \text{ (4. hirira ...)}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{5} x_{i5} = 1 \Leftrightarrow x_{15} + x_{25} + x_{35} + x_{45} = 1 \text{ (5. hirira ...)}$$

$$x_{ij} \in \mathbb{N} \; eta \; x_{ij} \geq 0 \; (25 \; baldintza, 25 \; aldagai \; daudelako)$$

Taula 4: 5 hiriko gure adibidearen enuntziatu estandarra

Eta `lpSolve` liburutegiko `lp(…)` funtziorako gure argumentuak prestatzen baditugu:

$$\text{direction} = \text{"}min\text{"}$$

$$\text{objective.in} = (0, 5, 7, 9, 11, 6, \dots, 12, 0)_{1\times 25}$$

$$\text{const.mat} = \overbrace{\begin{matrix} x_{11}, & x_{12}, & x_{13}, & & & \dots & & & , & x_{45}, & x_{55} \end{matrix}}^{} \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}_{35\times 25}$$

$$\text{const.dir} = (=, =, \dots, =, \geq, \geq, \dots, \geq)_{1\times 35}$$

$$\text{const.rhs} = (1, 1, \dots, 1, 0, 0, \dots, 0)_{1\times 35}$$

$$\text{int.vec} = (1, 2, \dots, 25)$$

$$\text{all.int} = TRUE$$

Taula 5: 5 hiriko gure adibidearen `lp(…)` *funtziorako argumentuak*

Hona hemen Rko kodigoa:

```
# 1. TSP eskuz (Programazio Linealeko buruketa baten moduan sartuta)

install.packages("lpSolve") #Instalatuko dugu lpSolve paketea, "Linear Programming Solve". Bertan
daudelako programazio linealeko enuntziatuak ebazteko funtzioak: guk lp() funtzioa erabiliko dugu.
library(lpSolve)            #Kargatzen dugu instalatutako paketea. Instalakuntza behin egitearekin
nahikoa da, baina paketea kargatu, RStudio irekitzen dugun guztietan. Bestela esango dizu lp() funtzioa
ez duela ezagutzen.

#Sortuko ditugu lehendabizi lp() funtzioak eskatuko dizkigun argumentu bakoitzak.
a <- c(0,5,7,9,11,
     6,0,6,7,8,
     7,7,0,9,10,
     8,7,8,0,11,
     9,7,11,5,0)
A1 <- matrix(c(0,1,1,1,1, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,
               0,0,0,0,0, 1,0,1,1,1, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0,
               0,0,0,0,0, 0,0,0,0,0, 1,1,0,1,1, 0,0,0,0,0, 0,0,0,0,0,
               0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 1,1,1,0,1, 0,0,0,0,0,
               0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 1,1,1,1,0,

               0,0,0,0,0, 1,0,0,0,0, 1,0,0,0,0, 1,0,0,0,0, 1,0,0,0,0,
               0,1,0,0,0, 0,0,0,0,0, 0,1,0,0,0, 0,1,0,0,0, 0,1,0,0,0,
               0,0,1,0,0, 0,0,1,0,0, 0,0,0,0,0, 0,0,1,0,0, 0,0,1,0,0,
               0,0,0,1,0, 0,0,0,1,0, 0,0,0,1,0, 0,0,0,0,0, 0,0,0,1,0,
               0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,1, 0,0,0,0,0),
             nrow=10, ncol=25, byrow=T)   #Lehenengo 10 baldintzak (5 atera eta 5 sartu).
A2 <- diag(1, nrow=25)            #Azken 25 baldintzak (aldagai guztiak positiboak).
A <- rbind(A1, A2)
nor <- c("=","=","=","=","=","=","=","=","=","=",">=",">=",">=",">=",">=",">=",
        ">=",">=",">=",">=",">=",">=",">=",">=",">=",">=",">=",">=",">=",">=",
        ">=",">=",">=",">=",">=",">=",">=",">=",">=",">=")
b <- c(1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
osoak <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25)

#Erabiliko dugun funtzioa da: lp(direction, objective.in, const.mat, const.dir, const.rhs, int.vec,
all.int). Zazpi argumentu eskatzen ditu, aldez aurretik definitu ditugu gehienak dagoeneko.
soluzioa <- lp(direction="min", objective.in = a, const.mat = A, const.dir = nor, const.rhs = b, int.vec
= osoak, all.int = TRUE)

#Emaitzak ikusteko:
soluzioa$solution
```

Eta hemen emaitza:

```
> #Emaitzak ikusteko:
> soluzioa$solution
 [1] 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0
```

Hau da, 1 balioarekin atera diren aldagaiak dira: $x_{12}, x_{25}, x_{31}, x_{43}, x_{54}$. Eta, beraz, TSP problema minimizatzen duen ibilbidea da: $1 \longrightarrow 2 \longrightarrow 5 \longrightarrow 4 \longrightarrow 3 \longrightarrow 1$. Eta guztira 33 unitateko distantzia dago.

```
> soluzioa
Success: the objective function is 33
```

1.4.2 TSP problema gisa (Rko TSP liburutegia)

Orain, baliatuko dugu Rko `TSP` liburutegia problema hau ebazteko. Ikusten denez, askoz ere samurragoa da liburutegiko honetako funtzioa erabilita ebaztea.

$$\text{distance\_matrix} = \begin{pmatrix} 0 & 5 & 7 & 9 & 11 \\ 6 & 0 & 6 & 7 & 8 \\ 7 & 7 & 0 & 9 & 10 \\ 8 & 7 & 8 & 0 & 11 \\ 9 & 7 & 11 & 5 & 0 \end{pmatrix}$$
$$\text{method} = \text{"}two\_opt\text{"}$$

Taula 6: 5 hiriko gure adibidearen `solve_TSP(…)` *funtziorako argumentuak*

Hona hemen Rko kodigoa:

```
# 2. TSP zuzenean (TSP buruketak ebazteko espresuki sortutako funtzioa erabiliz)

install.packages("TSP")  #Instalatuko dugu TSP paketea, "Travelling Salesman Problem". Bertan daudelako
TSP problemak ebazteko funtzioak: guk solve_TSP() funtzioa erabiliko dugu.
library(TSP)             #Kargatzen dugu instalatutako paketea. Instalakuntza behin egitearekin nahikoa
da, baina paketea kargatu, RStudio irekitzen dugun guztietan. Bestela esango dizu solve_TSP() funtzioa ez
duela ezagutzen.

distance_matrix <- matrix(c(0,5,7,9,11,
                            6,0,6,7,8,
                            7,7,0,9,10,
                            8,7,8,0,11,
                            9,7,11,5,0),
                          nrow = 5, byrow = TRUE)

#Erabiliko dugun funtzioa da: solve_TSP(tsp, method). Bi argumentu behar ditu: (1) tsp objektua eta (2)
erabiliko den metodoa (hau aukeran).
soluzioa2 <- solve_TSP(ATSP(distance_matrix), method="two_opt")

#Emaitzak ikusteko:
as.integer(soluzioa2)   # Ibilbidea
tour_length(soluzioa2)  # Emaitza (edo ibilbidearen luzera)
```

Eta soluzioa aurrekoaren berdina da:

```
> #Emaitzak ikusteko:
> as.integer(soluzioa2)   # Ibilbidea
[1] 3 1 2 5 4
> tour_length(soluzioa2)  # Emaitza (edo ibilbidearen luzera)
[1] 33
```

Hala ere, metodo heuristikoak erabiltzen dituenez, beti ez du emaitza berdina ematen, eta batzuetan ez du emaitza optimoa aurkitzen (baina bai “oso ona” den bat):

```
> #Emaitzak ikusteko:
> as.integer(soluzioa2)   # Ibilbidea
[1] 2 1 3 5 4
> tour_length(soluzioa2)  # Emaitza (edo ibilbidearen luzera)
[1] 35
```

2. Datu-bilketa

2.1 Datu-basea eraiki

Gure datu-basean Euskal Herriko 2.500 biztanlerik gorako udalerriak hartuko ditugu kontuan. Horretarako estatistika-iturri ofizialetara jo dugu:

- EAEkoak (Bizkaia, Gipuzkoa eta Araba): EUSTAT (Bibliografian [2])
- Nafarroa: NASTAT (Bibliografian [3])
- Iparraldekoak (Lapurdi, Nafarroa Behera eta Zuberoa): INSEE (Bibliografian [4])

Horrez gain, begiratu dugu banan-banan Korrikaren azken 3 edizioak (22. edizioa 2022an, 23. edizioa 2024an eta 24. edizioa aurten) herri bakoitzetik igaro den edo ez.

Herria	Probintzia	Biztanleria	Korrika 22	Korrika 23	Korrika 24	Herria	Probintzia	Biztanleria	Korrika 22	Korrika 23	Korrika 24
Bilbao	Bizkaia	346.933	☑	☑	☑	Pasaia	Gipuzkoa	16.767	☑	☑	☑
Vitoria-Gasteiz	Araba	253.956	☑	☑	☑	Zizur Nagusia	Nafarroa	16.124	☑	☑	☑
Pamplona / Iruña	Nafarroa	207.777	☑	☑	☑	Ermua	Bizkaia	15.558	☑	☑	☑
Donostia/S.Sebastián	Gipuzkoa	183.388	☑	☑	☑	Azpeitia	Gipuzkoa	15.431	☑	☑	☑
Barakaldo	Bizkaia	100.200	☑	☑	☑	Andoain	Gipuzkoa	14.732	☑	☑	☑
Getxo	Bizkaia	75.015	☑	☑	☑	Bergara	Gipuzkoa	14.725	☑	☑	☑
Irun	Gipuzkoa	60.665	☑	☑	☑	Don. Lohizune	Lapurdi	14.690	☑	☑	☑
Baiona	Lapurdi	53.312	☑	☑	☑	Estella-Lizarra	Nafarroa	14.329	☑	☑	☑
Santurtzi	Bizkaia	46.111	☑	☑	☑	Sopelana	Bizkaia	14.017	☑	☑	☑
Portugalete	Bizkaia	45.150	☑	☑	☑	Beasain	Gipuzkoa	13.915	☑	☑	☑
Angelu	Lapurdi	42.288	☑	☑	☑	Aranguren	Nafarroa	12.787	☑	☑	☑
Basauri	Bizkaia	40.323	☑	☑	☑	Azkoitia	Gipuzkoa	12.084	☑	☑	☑
Errenteria	Gipuzkoa	39.779	☑	☑	☑	Etxebarri	Bizkaia	11.901	☑	☑	☑
Tudela	Nafarroa	38.685	☑	☑	☑	Arrigorriaga	Bizkaia	11.680	☑	☑	☑
Leioa	Bizkaia	32.495	☑	☑	☑	Elgoibar	Gipuzkoa	11.659	☑	☑	☑
Durango	Bizkaia	30.165	☑	☑	☑	Trapagaran	Bizkaia	11.630	☑	☑	☑
Sestao	Bizkaia	28.019	☑	☑	☑	Oñati	Gipuzkoa	11.445	☒	☑	☑
Eibar	Gipuzkoa	27.118	☑	☑	☑	Berriozar	Nafarroa	11.201	☑	☑	☑
Biarritz	Lapurdi	25.810	☑	☑	☑	Tafalla	Nafarroa	10.778	☑	☑	☑
Galdakao	Bizkaia	24.788	☑	☑	☑	Antsoain	Nafarroa	10.641	☑	☑	☑
Erandio	Bizkaia	24.209	☑	☑	☑	Urruña	Lapurdi	10.594	☑	☑	☑
Zarautz	Gipuzkoa	23.464	☑	☑	☑	Oiartzun	Gipuzkoa	10.473	☑	☑	☑
V. Egués / Eguesibar	Nafarroa	22.443	☑	☑	☑	Amurrio	Araba	10.442	☑	☑	☑
Arrasate/Mondragón	Gipuzkoa	22.251	☑	☑	☑	Ordizia	Gipuzkoa	10.398	☑	☑	☑
Burlada / Burlata	Nafarroa	21.078	☑	☑	☑	Villava / Atarrabia	Nafarroa	10.016	☑	☑	☑
Hernani	Gipuzkoa	20.428	☑	☑	☑	Zumaia	Gipuzkoa	10.001	☑	☑	☑
Tolosa	Gipuzkoa	20.048	☑	☑	☑	Zumarraga	Gipuzkoa	9.568	☑	☑	☑
Lasarte-Oria	Gipuzkoa	19.808	☑	☑	☑	Abanto Zierbena	Bizkaia	9.353	☑	☑	☑
Barañáin / Barañain	Nafarroa	19.606	☑	☑	☑	Bokale	Lapurdi	8.934	☒	☒	☒
Amorebieta-Etxano	Bizkaia	19.229	☑	☑	☑	Corella	Nafarroa	8.712	☑	☒	☑
Laudio/Llodio	Araba	18.193	☑	☑	☑	Ortuella	Bizkaia	8.644	☑	☑	☑
Hendaia	Lapurdi	18.074	☑	☑	☑	Elortzibar	Nafarroa	8.469	☑	☑	☑
Mungia	Bizkaia	17.772	☑	☑	☑	Berango	Bizkaia	8.458	☑	☑	☑
Bermeo	Bizkaia	17.104	☑	☑	☑	Legazpi	Gipuzkoa	8.377	☒	☒	☒
Hondarribia	Gipuzkoa	16.960	☑	☑	☑	Cintruénigo	Nafarroa	8.333	☑	☒	☑
Gernika-Lumo	Bizkaia	16.889	☑	☑	☑	Zalla	Bizkaia	8.257	☑	☑	☑

Herria	Probintzia	Biztanleria	Korrika 22	Korrika 23	Korrika 24	Herria	Probintzia	Biztanleria	Korrika 22	Korrika 23	Korrika 24
Ondarroa	Bizkaia	8.144	☑	☑	☑	Ugao-Miraballes	Bizkaia	4.181	☑	☑	☑
Baztan	Nafarroa	7.863	☑	☑	☑	Cascante	Nafarroa	4.131	☒	☑	☒
Abadiño	Bizkaia	7.800	☑	☑	☑	Orkoien	Nafarroa	4.097	☒	☒	☑
Balmaseda	Bizkaia	7.773	☑	☑	☑	Olite / Erriberri	Nafarroa	4.087	☒	☒	☑
Huarte / Uharte	Nafarroa	7.737	☑	☑	☑	Ibarra	Gipuzkoa	4.076	☑	☑	☑
Altsasu / Alsasua	Nafarroa	7.710	☑	☑	☑	Cizur	Nafarroa	3.940	☑	☑	☑
Uztaritze	Lapurdi	7.684	☑	☑	☑	Zestoa	Gipuzkoa	3.908	☑	☑	☑
Bidarte	Lapurdi	7.674	☑	☒	☑	Iurreta	Bizkaia	3.830	☑	☑	☑
Astigarraga	Gipuzkoa	7.665	☑	☑	☑	Iruña Oka	Araba	3.780	☑	☑	☑
Berriobeiti	Nafarroa	7.631	☑	☒	☑	Soraluze	Gipuzkoa	3.765	☑	☑	☑
Hazparne	Lapurdi	7.615	☑	☑	☑	Bera	Nafarroa	3.763	☑	☑	☑
Derio	Bizkaia	7.535	☑	☑	☑	Azagra	Nafarroa	3.756	☑	☑	☑
Elorrio	Bizkaia	7.425	☑	☑	☑	Ribaforada	Nafarroa	3.701	☒	☒	☒
Muskiz	Bizkaia	7.282	☑	☑	☑	Oyón-Oion	Araba	3.688	☒	☒	☑
Senpere	Lapurdi	7.238	☑	☑	☑	Lemoa	Bizkaia	3.640	☑	☑	☑
Lekeitio	Bizkaia	7.186	☑	☑	☑	Milagro	Nafarroa	3.639	☑	☑	☑
Aretxabaleta	Gipuzkoa	7.183	☑	☑	☑	Arrangoitze	Lapurdi	3.588	☒	☒	☒
Urretxu	Gipuzkoa	6.795	☑	☑	☑	Mendavia	Nafarroa	3.535	☒	☒	☒
Güeñes	Bizkaia	6.728	☑	☑	☑	Zamudio	Bizkaia	3.384	☑	☑	☑
Kanbo	Lapurdi	6.715	☑	☑	☑	Basusarri	Lapurdi	3.317	☒	☒	☒
Usurbil	Gipuzkoa	6.485	☑	☑	☑	Maule-Lextarre	Zuberoa	3.216	☑	☑	☑
San Adrián	Nafarroa	6.481	☑	☑	☑	Alegría-Dulantzi	Araba	3.172	☑	☑	☑
Urnieta	Gipuzkoa	6.294	☑	☑	☑	Cortes	Nafarroa	3.160	☒	☒	☒
Orio	Gipuzkoa	6.197	☑	☑	☑	Villafranca	Nafarroa	3.077	☑	☑	☒
Lezo	Gipuzkoa	6.188	☑	☑	☑	Zaldibar	Bizkaia	3.051	☑	☑	☑
Gorliz	Bizkaia	6.105	☑	☑	☑	Leitza	Nafarroa	3.011	☑	☑	☒
Urduliz	Bizkaia	6.036	☑	☑	☑	Aoiz / Agoitz	Nafarroa	3.007	☒	☑	☒
Peralta / Azkoien	Nafarroa	6.003	☒	☑	☒	Urketa	Lapurdi	3.006	☒	☒	☒
Ziburu	Lapurdi	5.957	☑	☑	☑	Alonsotegi	Bizkaia	2.970	☒	☒	☒
Lazkao	Gipuzkoa	5.888	☑	☑	☑	Milafranga	Lapurdi	2.957	☑	☒	☑
Hiriburu	Lapurdi	5.779	☑	☑	☒	Getaria	Gipuzkoa	2.931	☑	☑	☑
Villabona	Gipuzkoa	5.778	☑	☑	☑	Gares	Nafarroa	2.930	☑	☑	☑
Agurain/Salvatierra	Araba	5.374	☑	☑	☑	Marcilla	Nafarroa	2.921	☑	☑	☒
Deba	Gipuzkoa	5.364	☑	☑	☑	Beskoitze	Lapurdi	2.920	☒	☒	☒
Mugerre	Lapurdi	5.353	☒	☒	☒	Andosilla	Nafarroa	2.913	☑	☑	☑
Mutriku	Gipuzkoa	5.283	☑	☑	☑	Zizurkil	Gipuzkoa	2.904	☑	☑	☑
Markina-Xemein	Bizkaia	5.056	☑	☑	☑	Ayala/Aiara	Araba	2.882	☑	☑	☑
Lodosa	Nafarroa	4.931	☑	☑	☑	Esteribar	Nafarroa	2.877	☒	☒	☒
Sangüesa / Zangoza	Nafarroa	4.862	☑	☑	☑	Bakio	Bizkaia	2.868	☑	☑	☑
Sondika	Bizkaia	4.651	☑	☑	☑	Caparroso	Nafarroa	2.861	☑	☑	☑
Azkaine	Lapurdi	4.561	☑	☑	☑	Sopuerta	Bizkaia	2.827	☑	☑	☑
Berriz	Bizkaia	4.541	☑	☑	☑	Karrantza Harana	Bizkaia	2.741	☒	☒	☒
Plentzia	Bizkaia	4.528	☑	☑	☑	Sara	Lapurdi	2.736	☑	☑	☑
Castejón	Nafarroa	4.473	☑	☑	☑	Lehuntze	Lapurdi	2.736	☒	☒	☑
Viana	Nafarroa	4.413	☒	☒	☑	Orozko	Bizkaia	2.736	☒	☒	☑
Igorre	Bizkaia	4.252	☑	☑	☑	Lesaka	Nafarroa	2.716	☑	☑	☑
Urduña-Orduña	Bizkaia	4.244	☒	☑	☒	Ablitas	Nafarroa	2.669	☒	☑	☒
Murchante	Nafarroa	4.239	☒	☒	☒	Funes	Nafarroa	2.558	☒	☑	☒
Eskoriatza	Gipuzkoa	4.227	☑	☑	☑	Ayegui / Aiegi	Nafarroa	2.550	☑	☑	☑
Beriáin	Nafarroa	4.197	☒	☒	☑	Etxarri Aranatz	Nafarroa	2.545	☑	☑	☑
						*Atharratze	Zuberoa	611	☒	☒	☑

Taula 7: Euskal Herriko 2.500 biztanletik gorako udalerriak Korrikaren azken hiru edizioetan

2.2 Koordenatuak R bidez

Udalerri bakoitzaren GPS koordenatuak behar izango ditugu, eta horiek lortzeko badira hainbat koordenatu-bilatzaile; esate baterako, honako hau (https://www.coordenadas-gps.com):

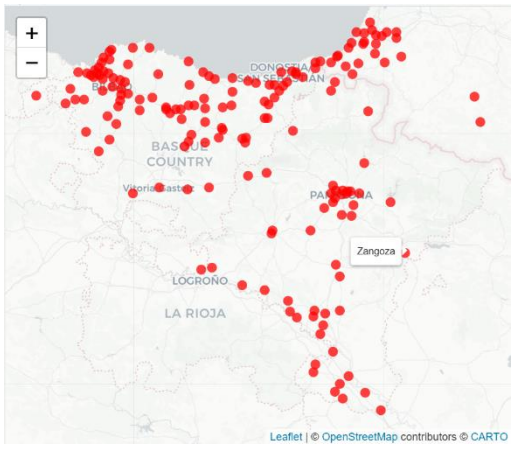
Irudia 5: https://www.coordenadas-gps.com/convertidor-de-coordenadas-gps helbideko adibidea

173 herriren koordenatuak jaso behar ditugunez, nekeza samarra izango litzateke den-denak eskuz kalkulatzea; beraz, adimen artifizialari (Chat GPTri) laguntza eskatu diogu Exceletik jasotako herri-zerrenda batetik, latitude eta longitudea kalkulatzeko.

```
library(xlsx) # Excela erabiltzeko Rko liburutegia.

#read.xlsx(...) funtzioak bi argumentu behar ditu: 1) Ireki nahi den artxiboaren helbidea; 2) Excelaren
zein pestañatik hartu datuak.
herriak <- read.xlsx("C:\\Users\\Hezkuntza\\Desktop\\Herriak_Korrika1.xlsx",1)

#ChatGPTren kodigoa. nominatim.openstreetmap.org webgunetik jasotzen ditu herrien koordenatuak.

library(httr)
library(jsonlite)
library(dplyr)

geocode_nominatim <- function(place){
  url <- paste0(
    "https://nominatim.openstreetmap.org/search?q=", URLencode(place),"&format=json&limit=1")
  res <- fromJSON(url)
  if(length(res) == 0){
    return(data.frame(lat = NA, lon = NA))}
  data.frame(lat = as.numeric(res$lat), lon = as.numeric(res$lon))}

coords <- lapply(herriak$Udalerriak, geocode_nominatim)
coords <- bind_rows(coords)

herriak_geo <- bind_cols(herriak, coords)
```

Eta emaitza jaso dugu **herriak_geo** izeneko aldagaian:

```
> herriak_geo
          Udalerriak      lat        lon
1              Bilbao 43.26300 -2.9350039
2             Gasteiz 42.84651 -2.6724025
3               Iruña 42.81820 -1.6440089
4            Donostia 43.32242 -1.9838889
5           Barakaldo 43.29492 -2.9887476
6               Getxo 43.34312 -3.0078627
...

171             Aiegi 42.65883 -2.0369908
172   Etxarri Aranatz 42.90758 -2.0648822
173       Atharratze 43.11625 -0.8645311
```



Irudia 6: 173 herriak EHko mapan kokatuta (`leaflet(…)` funtzioaren bitartez)

2.3 Distantzia-matrizeak R bidez

1.4.2 atalean (TSP problemak R bidez ebatzi) ikusi dugun moduan, distantzia-matrize batetik abiatuta oso erraza da TSP problemak ebaztea Rko `TSP` liburutegiko `solve_TSP(…)` funtzioaren bitartez.

Kontua da guk behar ditugula bi herriren arteko distantziak, baina ez lerro zuzenean, baizik eta bi herri horiek lotzen dituen errepideetatik eginda. Honek behartzen gaitu nabigazio-rutak sortzen dituen plataforma bat erabiltzen (GOOGLE MAPSen antzekoa) eta guk HERE TECHNOLOGIES baliabidea erabiliko dugu (aurrekoaren alternatiba).

Hau guztia egiteko, oinarritu gara Jindra Lacko-ren "*Travelling Salesman Problem with HERE*" lanean (Bibliografian [5]), eta herrien arteko rutak kalkulatzeko kodigoa fusilatu dugu.

Egin dezagun adibide[1] bat lehenengo datu-basearen 10 herriekin (jendetsuenak, alegia).

```
# Lehendabidizi, lehen hamar herriekin egingo dugu adibide bat.
herriak_geo_10 <- herriak_geo[1:10, ] #Lehen 10 herriak bakarrik hartu.

library(hereR) # hereR liburutegia kargatu.
hereR::set_key("                                          ") # HERE APIko gure kontura konektatuko.

# Nondik nora egin behar duen. Gure kasuan 10 herri daude adibidean. Guztira 100 datu bilatu behar ditu.
indizeak<-expand.grid(from=seq_along(herriak_geo_10$Udalerriak),to=seq_along(herriak_geo_10$Udalerriak))

# Kalkulatu distantzia-matrizea. Jindra Lackoren webgunetik fusilatuta.

      for (i in seq_along(indizeak$from)) {

        active_route <- hereR::route(origin = herriak_geo_10[indizeak$from[i], ],
                                     destination = herriak_geo_10[indizeak$to[i], ],
                                     transport_mode = "pedestrian") %>%                #Ibilgailua: Oinez
          mutate(idx_origin = indizeak$from[i],
                 idx_destination = indizeak$to[i],
                 route_name = paste(herriak_geo_10$Udalerriak[indizeak$from[i]],
                                    ">>",
                                    herriak_geo_10$Udalerriak[indizeak$to[i]])) %>%
          relocate(idx_origin, idx_destination, route_name) %>%
          st_zm()
        if (i == 1) {
          routes <- active_route
        } else {
          routes <- routes %>%
            bind_rows(active_route)}
      }

# Lortutako emaitzarekin distantzia-matrizea sortzeko:

distantzia_matrizea <- matrix(routes$distance,
                              nrow = nrow(herriak_geo_10),
                              ncol = nrow(herriak_geo_10))

rownames(distantzia_matrizea) <- herriak_geo_10$Udalerriak
colnames(distantzia_matrizea) <- herriak_geo_10$Udalerriak
```

Hona distantzia-matrizea (metrotan, eta ez simetrikoa), `solve_TSP(…)` funtzioan sartzeko prest.

```
> distantzia_matrizea
           Bilbao Gasteiz  Iruña Donostia Barakaldo  Getxo   Irun Baiona Santurtzi Portugalete
Bilbao          0   67662 144985   106847      7028  12315 124656 166781     12757       11321
Gasteiz     67662       0  96070   109666     74151  79986 123165 165290     79880       78444
Iruña      144987   96072      0    88812    152032 154770  93803 104594    157761      156325
Donostia   106841  109673  88812        0    113800 116624  18331  60456    119529      118093
Barakaldo    7023   74146 152027   113755         0   7575 131564 173689      6115        4654
Getxo       12309   79980 154764   116626      7575      0 134435 176560      4043        3165
Irun       124650  123160  93803    18322    131609 134433      0  42125    137338      135902
Baiona     166802  165309 104592    60474    173761 176582  42149      0    179487      178051
Santurtzi   12759   79882 157763   119491      6115   4032 137300 179425         0        1659
Portugalete 11321   78444 156325   118053      4654   3165 135862 177987      1659           0
```

[1] HERE APIko pasahitza ezkutatuta dago, tutorearen kontu korronteari esleituta dagoelako.

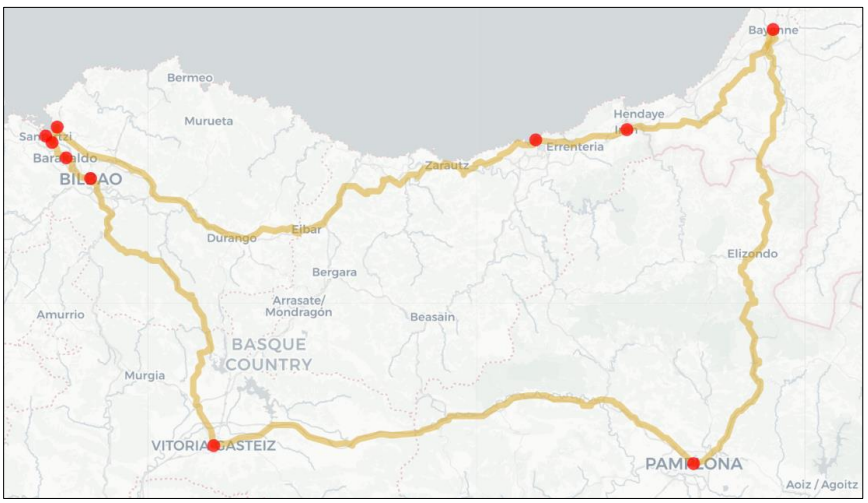
```r
library(TSP) # TSP problemak ebazteko liburutegia.

soluzioa_10 <- solve_TSP(ATSP(distantzia_matrizea))    # solve_TSP(...) funtzioarekin ebatzi.

herriak_geo_10$Udalerriak[as.integer(soluzioa_10)]     # Emaitza ikusteko
tour_length(soluzioa_10)                               # Ibilbidearen luzera ikusteko

# Behin ebatzita, mapan ibilbidea ikusteko kodigoa, Jindra Lackoren webgunetik fusilatuta.

      stops <- as.numeric(soluzioa_10)

      distance_route <- data.frame(idx_origin = stops,
                                   idx_destination = c(stops[2:(nrow(herriak_geo_10))],
                                                       stops[1]))

      distance_result <-  distance_route %>%
        inner_join(routes,
                   by = c("idx_origin", "idx_destination")) %>%
        st_as_sf()

      leaflet(distance_result) %>%
        addProviderTiles("CartoDB.Positron") %>%
        addPolylines(color = "GoldenRod",
                     popup = ~route_name) %>%
        addCircleMarkers(data = herriak_geo_10,
                         fillColor = "red",
                         radius = 5,
                         stroke = F,
                         fillOpacity = .75,
                         label = ~ Udalerriak)
```

Eta hauexek dira erantzunak:

```
> soluzioa_10 <- solve_TSP(ATSP(distantzia_matrizea))
> herriak_geo_10$Udalerriak[as.integer(soluzioa_10)] #Emaitza ikusteko
 [1] "Irun"        "Donostia"    "Getxo"       "Santurtzi"   "Portugalete"
 [6] "Barakaldo"   "Bilbao"      "Gasteiz"     "Iruña"       "Baiona"
> tour_length(soluzioa_10) #Ibilbidearen luzera ikusteko
[1] 462800
```

Ibilbide laburrena (edo laburrenetarikoa, metodo heuristikoekin ebatzita dagoelako) herri horiek guztiak lotzeko 462.800 metro ditu, eta honako hau da ibilbidea:

Irun → Donostia → Getxo[2] → Santurtzi → Portugalete → Barakaldo → Bilbo → Gasteiz → Iruñea → Baiona → Irun



Irudia 7: Euskal Herriko 10 herri jendetsuenak lotzen dituen ibilbiderik laburrena

[2] Zubi eskegia hartzen du, ez dugu modurik aurkitu hori ekiditeko. Ferriak ekidin daitezke, baina HERE APIak emandako *"carShuttleTrain"* izeneko transportea ez.

3. Ibilbidearen eraikuntza

3.1 TSP ez ziklikoak

TSP problema, definizioz, hasierako hirian amaitzen den ibilbide laburrena lortzean datza. Hau da, ibilbide ziklikoak sortzen ditu beti. Guk, ordea, ez dugu ibilbide ziklikorik nahi, baizik eta hiri jakin batetik hasi eta hiri jakin batean bukatzen duen ibilbide laburrena topatu. Zer egingo dugu hori lortzeko? Herri fiktizio bat sortu (*"Dummy node"*) gero hortik zikloa mozteko:

1. Sortuko dugu *Dummy* izeneko hiri fiktizio bat, hasiera eta bukaera herrietatik 0 m-ko distantziara dagoena eta gainontzeko herri guztietatik infinituko distantziara (10^9 m). Modu horretan, TSP problemak halabeharrez kokatuko beharko du *Dummy* hiria hasiera eta bukaera herrien artean (ibilbidearen distantzia minimizatzeko). Adibidez:
    - Hasiera: Gasteiz
    - Bukaera: Donostia

```
# Finkatuko ditugu hasiera eta bukaera herriak:
hasiera <- "Gasteiz"
bukaera   <- "Donostia"

# Dummy herria gehitu (errenkadetan eta zutabeetan),
distantzia_matrizea_hed <- distantzia_matrizea %>%
  cbind(Dummy = 1e9) %>%
  rbind(Dummy  = 1e9)       # Distantzia-matrize hedatua

# Dummy hiria hasiera eta bukaera herriekin lotu
distantzia_matrizea_hed["Dummy", hasiera] <- 0
distantzia_matrizea_hed[hasiera, "Dummy"] <- 0

distantzia_matrizea_hed["Dummy", bukaera] <- 0
distantzia_matrizea_hed[bukaera, "Dummy"] <- 0
```

```
> distantzia_matrizea_hed
             Bilbao Gasteiz      Iruña Donostia  Barakaldo      Getxo       Irun     Baiona  Santurtzi Portugalete Dummy
Bilbao            0   67662     144985   106847       7028      12315     124656     166781      12757      11321 1e+09
Gasteiz       67662       0      96070   109666      74151      79986     123165     165290      79880      78444 0e+00
Iruña        144987   96072          0    88812     152032     154770      93803     104594     157761     156325 1e+09
Donostia     106841  109673      88812        0     113800     116624      18331      60456     119529     118093 0e+00
Barakaldo      7023   74146     152027   113755          0       7575     131564     173689       6115       4654 1e+09
Getxo         12309   79980     154764   116626       7575          0     134435     176560       4043       3165 1e+09
Irun         124650  123160      93803    18322     131609     134433          0      42125     137338     135902 1e+09
Baiona       166802  165309     104592    60474     173761     176582      42149          0     179487     178051 1e+09
Santurtzi     12759   79882     157763   119491       6115       4032     137300     179425          0       1659 1e+09
Portugalete   11321   78444     156325   118053       4654       3165     135862     177987       1659          0 1e+09
Dummy    1000000000       0 1000000000        0 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1e+09
```

2. TSP problema ebatziko dugu baldintza berri hauetan. *Dummy* hiria hasiera eta bukaera herrien artean geratzen da, baina edozein tokitan kokatuta egon daiteke.

```
soluzioa <- solve_TSP(ATSP(distantzia_matrizea_hed)) # Ebatzi TSP problema hedatua
Ibilbidea <- labels(soluzioa) #Emaitza ikusteko
```

```
> Ibilbidea
 [1] "Portugalete" "Barakaldo"   "Bilbao"      "Gasteiz"     "Dummy"       "Donostia"
 [7] "Irun"        "Baiona"      "Iruña"       "Getxo"       "Santurtzi"
```

3. Hurrengo pausua izango da ibilbidea rotatzea Dummy herriaren hurrengo herrian hasi eta buelta osoa egin arte.

```
# Ibilbidea rotatzen dugu, Dummyren hurrengo herritik hasi eta bere aurreko herrira heldu arte.
Ibilbidea_rot <- c(Ibilbidea[(pos_dummy+1):length(Ibilbidea)],
                   Ibilbidea[1:(pos_dummy-1)])
```

```
> Ibilbidea_rot
 [1] "Donostia"    "Irun"        "Baiona"      "Iruña"       "Getxo"       "Santurtzi"
 [7] "Portugalete" "Barakaldo"   "Bilbao"      "Gasteiz"
```

4. Hemen gerta daiteke (gertatu den bezala), hasiera eta bukaera herriak aldrebes agertzea. Izan ere, zikloaren noranzkoa zehazterik ez dago TSP agindua ematen dugunean. Ikusten badugu hori gertatu dela, zikloa inbertitzearekin nahikoa litzateke.

```
> rev(Ibilbidea_rot)
 [1] "Gasteiz"     "Bilbao"      "Barakaldo"   "Portugalete" "Santurtzi"   "Getxo"
 [7] "Iruña"       "Baiona"      "Irun"        "Donostia"
```

Eta, aurreko atalean egin dugun moduan, eskatzen badiogu ibilbide hori lotzeko (azken herritik lehenengo herrirako lotura egin gabe), honako hau jasotzen dugu:

Irudia 8: Gasteizen hasi eta Donostian bukatzen den ibilbiderik laburrena

```
> tour_length(soluzioa)
[1] 404859
```

Eta ibilbide horren distantzia 404.859 m-koa da (2.3 ataleko adibidekoa baino laburragoa, Donostiatik Gasteizerako distantzia 0 delako, *Dummy Nodo*aren eraginez).

3.2 Klusterrak

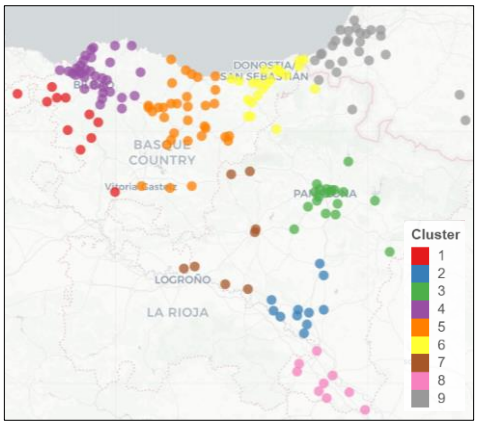
Aurreko ataleko adibidean, 10 herrien arteko TSP problema ebatzi dugu. Hori egiteko 10×10 dimentsioko distantzia-matrize bat egin behar izan dugu, eta horrek eskatzen du HERE APIaren bitartez 100 ruta sortu behar izatea. Hori egiteko ordenagailuak minutu bat inguru denbora behar izan du.

Idealena izango litzateke gure 173 herriekin TSP problema ebaztea, baina horretarako 173×173 dimentsioko distantzia-matrizea sortu beharko genuke, 30.000 ruta desberdin kalkulatuz! Hori egiteko, gutxienez 300 minutu beharko genituzke, eta gainera HERE zerbitzuak dirua kobratu egiten du eskakizun asko egiten direnean.

Gure soluzioa izan da klusterrak erabiltzea. Kluster desberdinak egingo ditugu (9) eta TSP problema bat ebatziko dugu kluster bakoitzerako. Bukaeran denak elkartu ditugu.

```r
k <- 9 # Kluster-kopurua.

set.seed(123) # Beti emaitza berdinak ikusteko (ausazkotasuna kentzeko).
km <- kmeans(st_coordinates(herriak_geo), centers = k) #Herriak klusterretan banatzen ditu kmeans
algoritmoaren bidez.
herriak_geo$cluster <- as.factor(km$cluster) #Zutabe berri bat sortzen du herrien klusterrekin.
```

```
> herriak_geo[, c("Udalerriak", "cluster")]
Simple feature collection with 173 features and 2
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -3.360652 ymin: 41.92353 xmax
Geodetic CRS:  WGS 84
First 10 features:
    Udalerriak cluster                   geometry
1       Bilbao       4   POINT (-2.935004 43.263)
2      Gasteiz       5 POINT (-2.672403 42.84651)
3        Iruña       3  POINT (-1.644009 42.8182)
4     Donostia       6 POINT (-1.983889 43.32242)
5    Barakaldo       4 POINT (-2.988748 43.29492)
6        Getxo       4 POINT (-3.007863 43.34312)
7         Irun       6 POINT (-1.788809 43.33832)
8       Baiona       9 POINT (-1.473666 43.49451)
9    Santurtzi       4 POINT (-3.031877 43.32875)
10 Portugalete       4 POINT (-3.019863 43.31896)
```



Irudia 9: Herriak klusterretan banatuta

Definituko ditugu kluster bakoitzeko hasiera- eta bukaera-herriak, eta kluster bakoitzean sortuko ditugu ibilbideak 3.1 atalean (TSP ez ziklikoak) ikusi dugun moduan:

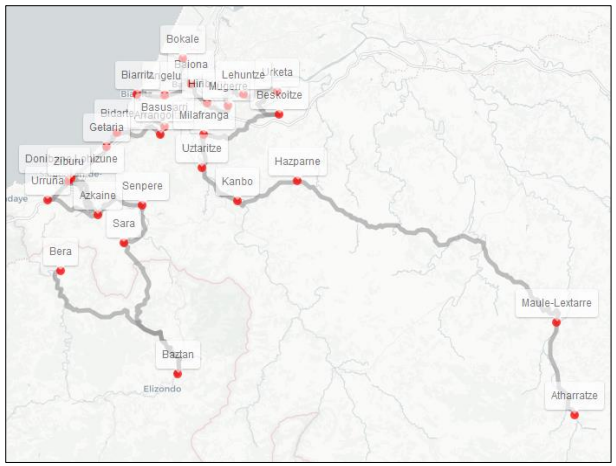
Atharratze → Kluster 9 → Bera → Lesaka → Kluster 6 → Leitza → Esteribar → Kluster 3 → Gares → Tafalla → Kluster 2 → Milagro → Castejón → Kluster 8 → Corella → Lodosa → Kluster 7 → Altsasu → Lazkao → Kluster 5 → Gasteiz → Iruña Oka → Kluster 1 → Sopuerta → Muskiz → Kluster 4 → Bilbo

Hurrengo orrialdean kluster bakoitzean lortutako emaitzak azaltzen dira.

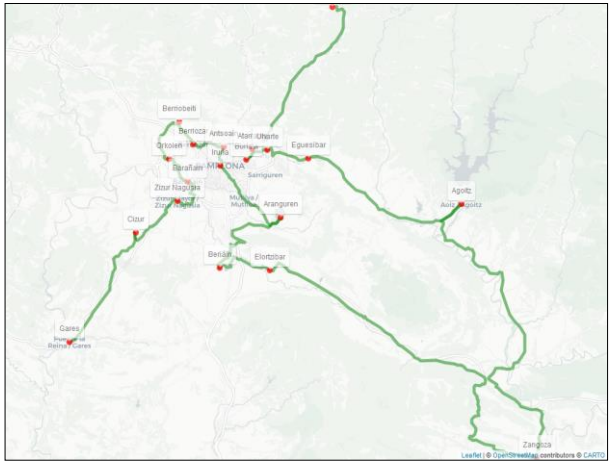
Bukatzeko, klusterren arteko loturak egin behar dira eta ibilbide guztiak elkartu.

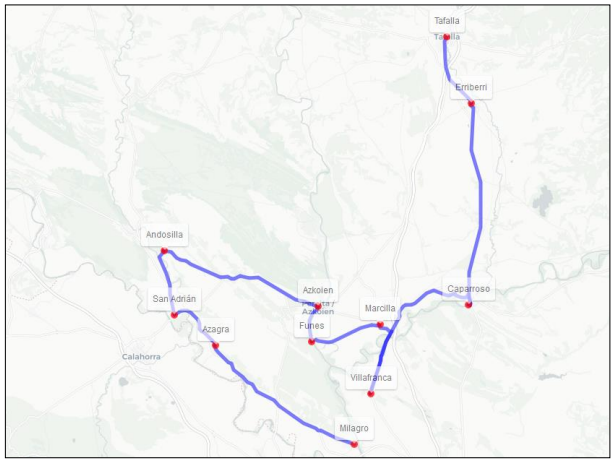
```r
# Klusterren arteko loturak egin.

konexioa_1_2 <- hereR::route(
          origin = herriak_geo[herriak_geo$Udalerriak == "Bera", ],              # Bera
          destination = herriak_geo[herriak_geo$Udalerriak == "Lesaka", ],       # Lesaka
          transport_mode = "pedestrian"
        ) %>% st_zm()
(…)
konexioa_8_9 <- hereR::route(
          origin = herriak_geo[herriak_geo$Udalerriak == "Sopuerta", ],          # Sopuerta
          destination = herriak_geo[herriak_geo$Udalerriak == "Muskiz", ],       # Muskiz
          transport_mode = "pedestrian"
        ) %>% st_zm()

# Ibilbide finala lotu.

ibilbide_finala <- dplyr::bind_rows(distance_result_1, konexioa_1_2, distance_result_2, konexioa_2_3,
      distance result 3, konexioa 3 4, distance result 4, konexioa 4 5, distance result 5, konexioa 5 6,
      distance_result_6, konexioa_6_7, distance_result_7, konexioa_7_8, distance_result_8, konexioa_8_9,
      distance_result_9)
```

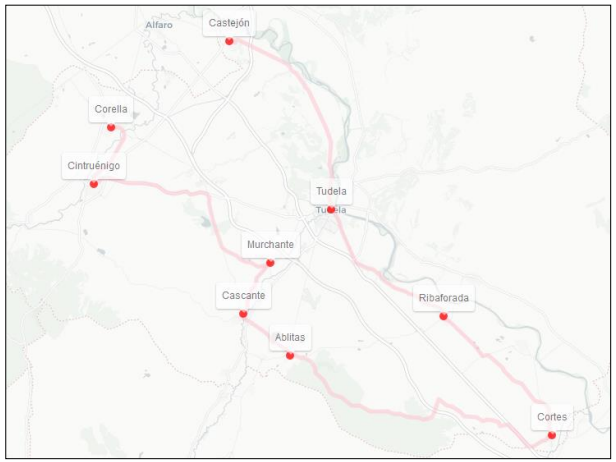
```
> ibilbide_finala
   idx_origin idx_destination                    route_name
1          27             20 Atharratze >> Maule-Lextarre
2          20             10   Maule-Lextarre >> Hazparne
3          10             12             Hazparne >> Kanbo
4          12              8           Kanbo >> Uztaritze
...
175         2             32      Barakaldo >> Alonsotegi
176        32              1         Alonsotegi >> Bilbao
```

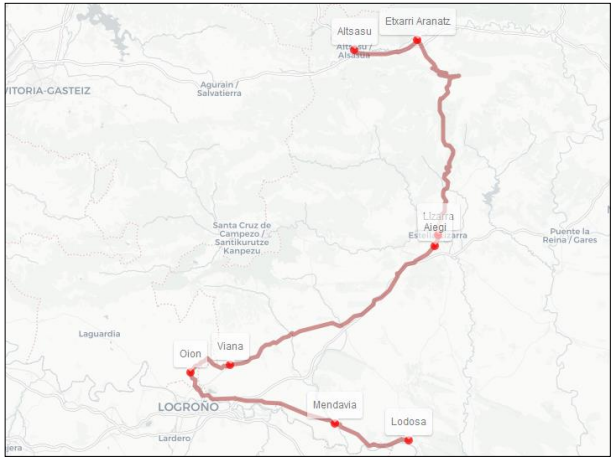

Bokale
Biarritz
Lehuntze
Beskoitze
Getaria
Milafranga
Uztaritze
Hazparne
Urrúña
Senpere
Kanbo
Azkaine
Sara
Bera
Maule-Lextarre
Baztan
Elizondo
Atharratze

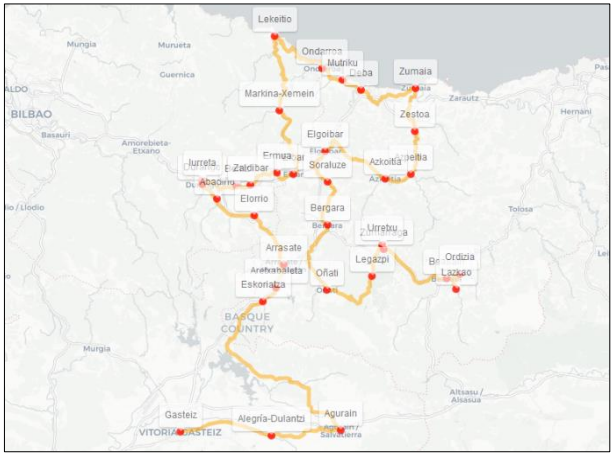

Hondarribia
Irun
Donostia
SAN SEBASTIAN
Pasaia
Lezo
Oiartzun
Zarautz
Orio
Usurbil
Astigarraga
Hernani
Urnieta
Lesaka
Andoain
Zizurkil
Villabona
Ibarra
Leitza

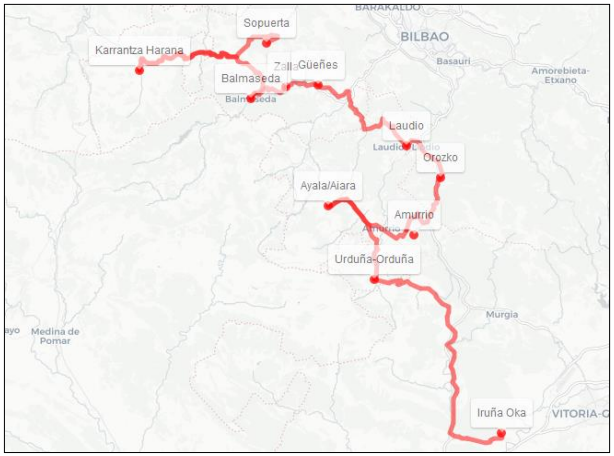

Bermeo
Mungia
Egesibar
Agoitz
Aranguren
Elortzibar
Gares
Zangoza

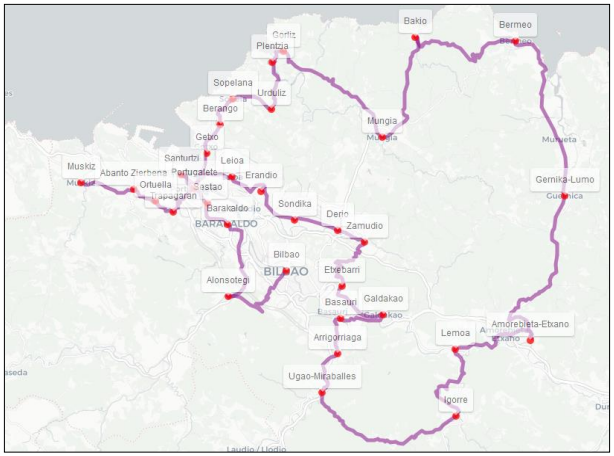

Tafalla
Erriberri
Andosilla
San Adrián
Azkoien
Funes
Marcilla
Caparroso
Azagra
Villafranca
Calahorra
Milagro


Alfaro
Castejón
Corella
Cintruénigo
Tudela
Murchante
Cascante
Ribaforada
Ablitas
Cortes


Altsasu
Etxarri Aranatz
VITORIA-GASTEIZ
Agurain / Salvatierra
Santa Cruz de Campezo / Santikurutze Kanpezu
Puente la Reina / Gares
Laguardia
Oion
Viana
LOGROÑO
Mendavia
Lodosa
Lardero

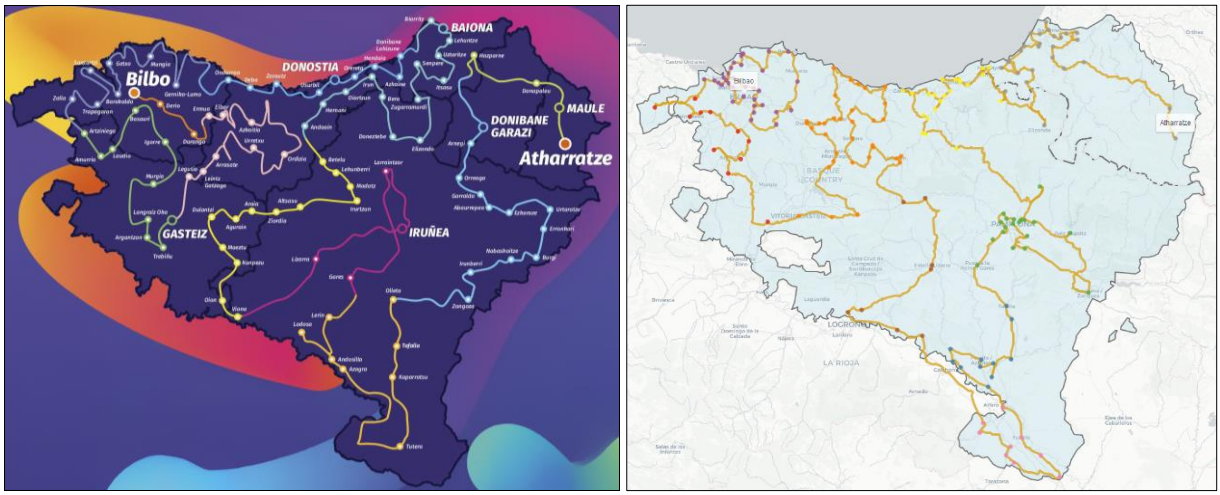

Lekeitio
Ondarroa
Mutriku
Zumaia
Markina-Xemein
Zestoa
BILBAO
Elgoibar
Soraluze
Azkoitia
Eibar
Ermua
Zaldibar
Iurreta
Abadiño
Elorrio
Bergara
Urretxu
Arrasate
Legazpi
Ordizia
Lazkao
Oñati
Eskoriatza
BASQUE COUNTRY
Gasteiz
Alegría-Dulantzi
Agurain
VITORIA-GASTEIZ


Sopuerta
Karrantza Harana
BILBAO
Güeñes
Balmaseda
Laudio
Orozko
Ayala/Aiara
Amurrio
Urduña-Orduña
Iruña Oka
Medina de Pomar
Murgia
VITORIA-G


Bakio
Bermeo
Plentzia
Sopelana
Urduliz
Berango
Getxo
Leioa
Erandio
Santurtzi
Muskiz
Abanto Zierbena
Ortuella
Sestao
Barakaldo
Sondika
Derio
Zamudio
Gernika-Lumo
Bilbao
BILBAO
Etxebarri
Alonsotegi
Basauri
Galdakao
Arrigorriaga
Lemoa
Amorebieta-Etxano
Ugao-Miraballes
Igorre

4. Emaitza

4.1 Gure ekoizpena[3] eta benetakoarekiko konparazioa



Irudia 10: Ezkerrean benetako ibilbidea eta eskuman guk egindakoa

```
Luzera_totala <- Luzera_1_zuz + Luzera_2_zuz + Luzera_3_zuz + Luzera_4_zuz + Luzera_5_zuz + Luzera_6_zuz
               + Luzera_7_zuz + Luzera_8_zuz + Luzera_9_zuz
               + konexioa_1_2$distance + konexioa_2_3$distance + konexioa_3_4$distance
               + konexioa_4_5$distance + konexioa_5_6$distance + konexioa_6_7$distance
               + konexioa_7_8$distance + konexioa_8_9$distance
```

```
> Luzera_totala
[1] 1751500
```

Gure ibilbideak 1.751,5 km ditu eta benetakoak 2,175 km. Gure estimazioen arabera, herri/hiri barruan ibiltzeko ¿? km behar dira, beraz...

4.2 Ondorioak eta hausnarketa

Zuentzako.

Ideiak:

[3] Eranskinetan ekoitzitako ibilbidea handiago dago ikusgai.

5. Bibliografia

[1] El viajante de comerico, Departamento de Estadística e Investigación Operativa, Universidad de Cádiz.
https://knuth.uca.es/moodle/mod/page/view.php?id=3416

[2] Euskal AEko biztanleria, lurralde eremuaren, adin betearen talde handien eta sexuaren arabera . 2001/01/01 - 2025/01/01, EUSTAT.

https://eu.eustat.eus/bankupx/pxweb/eu/DB/-/PX_010154_cepv1_ep06b.px

[3] Población según sexo y municipio de residencia, NASTAT.

https://nastat.navarra.es/es/tablas_mixtas/-/tag/padrones

[4] Statistiques locales, INSEE.

https://statistiques-locales.insee.fr/index.php#c=indicator&selcodgeo=64102&view=map1

[5] Lacko, Jindra. *Travelling Salesman Problem with HERE*. JLA Data.

https://www.jla-data.net/eng/travelling-salesman-problem-with-here/

6. Eranskinak

6.1 Erabilitako kodigoa

```
# 1. Lehendabizi Exceleko datuak irakurri behar ditugu:

    library(xlsx) # Excela erabiltzeko Rko liburutegia.
    herriak_geo <- read.xlsx("C:\\Users\\Hezkuntza\\Desktop\\Herriak_Korrika.xlsx",1)

# 2. Herriak mapan ikusteko:

    library(sf)    # Datu geografikoak lantzeko liburutegia.
    library(leaflet)  # Mapa interaktiboak sortzeko liburutegia.

    herriak_geo <- st_as_sf(
      herriak_geo,
      coords = c("lon", "lat"),  # Lehenik LONGITUDEA, ondoren LATITUDEA.
      crs = 4326                 # WGS84 koordenatu-sistema.
    )

    # Gure herriak mapan ikusteko agindua (leaflet funtzioa).

    leaflet(herriak_geo) %>%                    # Nondik hartu informazioa
      addProviderTiles("CartoDB.Positron") %>%  # Azpiko mapa jartzeko
      addCircleMarkers(fillColor = "red",       # Zein kolorerekin adierazi puntu geometrikoak
                       radius = 5,              # Zenbateko lodiera puntu bakoitza
                       stroke = F,              # Puntuek ez izateko kanpoko ertzik
                       fillOpacity = .75,       # Puntuen opakotasun-maila: %75
                       label = ~ Udalerriak)    # Puntuen izena agertzeko kursorea puntutik igarotakoan

# 3. Klusterrak egin.

    k <- 9 # Kluster-kopurua.

    # Herriak klusteretan banatuko ditugu.

    set.seed(123) # Beti emaitza berdinak ikusteko (ausazkotasuna kentzeko).
    km <- kmeans(st_coordinates(herriak_geo), centers = k) # Herriak klusterretan banatu.
    herriak_geo$cluster <- as.factor(km$cluster) # Zutabe berri bat sortu herriaren klusterrarekin.

    # Herriak mapan ikusteko:

    leaflet(herriak_geo) %>%
      addProviderTiles("CartoDB.Positron") %>%
      addCircleMarkers(fillColor = ~colorFactor(palette = "Set1", domain = cluster)(cluster),
                       radius = 5,
                       stroke = F,
                       fillOpacity = 0.75,
                       label = ~ Udalerriak) %>%
      addLegend("bottomright",     # Legendatxoa agertzeko klusterren koloreekin.
                pal = colorFactor(palette = "Set1", domain = herriak_geo$cluster),
                values = ~cluster,
                title = "Cluster",
                opacity = 1)

# 4. Kluster bakoitzean TSP egin, hasierako eta bukaerako herriak aukeratuta.

    library(hereR) # hereR liburutegia kargatu.
    library(dplyr) # Agindu bereziak emateko funtzioa "%>%"

    # HERE APIn aurretik zabaldutako gure kontura konektatuko gara:

    hereR::set_key("                                          ")

    # 4.1 Lehenengo klusterra (grisa: kluster=9)

        kluster1 <- herriak_geo[herriak_geo$cluster == 9, ] # Lehendabizi klusterreko herriak aukeratu.

        # Kalkulatu distantzia-matrizea. Jindra Lackoren webgunetik fusilatuta. (3 minutu 10 segundo)

        indiz_1 <- expand.grid(from = seq_along(kluster1$Udalerriak),
                               to = seq_along(kluster1$Udalerriak))

        for (i in seq_along(indiz_1$from)) {

          active_route <- hereR::route(origin = kluster1[indiz_1$from[i], ],
                                       destination = kluster1[indiz_1$to[i], ],
                                       transport_mode = "pedestrian") %>%  # Ibilgailua: Oinez
            mutate(idx_origin = indiz_1$from[i],
                   idx_destination = indiz_1$to[i],
                   route_name = paste(kluster1$Udalerriak[indiz_1$from[i]],
                                      ">>",
                                      kluster1$Udalerriak[indiz_1$to[i]])) %>%
```

```r
    relocate(idx_origin, idx_destination, route_name) %>%
    st_zm()

  if (i == 1) {
    routes_1 <- active_route
  } else {
    routes_1 <- routes_1 %>%
      bind_rows(active_route)
  }
}

# Lortutako emaitzarekin distantzia-matrizea sortzeko:

dm1 <- matrix(routes_1$distance,
              nrow = nrow(kluster1),
              ncol = nrow(kluster1))

rownames(dm1) <- kluster1$Udalerriak # Errenkadei izenak eman
colnames(dm1) <- kluster1$Udalerriak # Zutabeei izenak eman

# Finkatuko ditugu hasiera eta bukaera herriak:

hasiera <- "Atharratze"
bukaera <- "Bera"

# 1. Dummy nodoa gehitu (hiri faltsua).

# Dummy herria gehitu (errenkadetan eta zutabeetan):

dm1_hed <- dm1 %>%
  cbind(Dummy = 1e9) %>%   # Balio denak infinito (edo 10^9)
  rbind(Dummy = 1e9)

# Dummy hiria hasiera eta bukaera herriekin lotu:

dm1_hed["Dummy", hasiera] <- 0
dm1_hed[hasiera, "Dummy"] <- 0

dm1_hed["Dummy", bukaera] <- 0
dm1_hed[bukaera, "Dummy"] <- 0

# 2. Ebatzi TSP hedatua

library(TSP) # TSP problemak ebazteko liburutegia.

soluzioa_1 <- solve_TSP(ATSP(dm1_hed)) # Ebatzi TSP problema hedatua (Dummya duena).

Ibilbidea_1 <- labels(soluzioa_1) # Emaitza ikusteko.
Luzera_1 <- tour_length(soluzioa_1) # Ibilbidearen luzera ikusteko.

# 3. Ibilbidea lortu, Dummy agertu gabe.

pos_dummy_1 <- which(Ibilbidea_1 == "Dummy") # Jakiteko zein poiziotan dagoen Dummy herria.

Ibilbidea_1_rot <- c(Ibilbidea_1[(pos_dummy_1+1):length(Ibilbidea_1)],
                     Ibilbidea_1[1:(pos_dummy_1-1)]) # Ibilbidea rotatzen du.

Ibilbidea_1_zuz <- Ibilbidea_1_rot[Ibilbidea_1_rot != "Dummy"] # Dummy herria kentzen du.

# Ibildiearen noranzkoa arbitrario da, eta gerta liteke hasieran hasi beharrean bukaeratik hastea.
  Kasu horretan buelta eman beharko genioke erantzunen bektoreari.

if (Ibilbidea_1_zuz[1] != hasiera) {
  Ibilbidea_1_zuz <- rev(Ibilbidea_1_zuz)
}

indizeak_1 <- match(Ibilbidea_1_zuz, colnames(dm1)) # Jakiteko ibilbidearen posizioak.
Luzera_1_zuz <- sum(dm1[cbind(indizeak_1[-length(indizeak_1)], indizeak_1[-1])]) # Luzera zuzena.

# 4. Behin ebatzita, mapan ibilbidea ikusteko kodigoa, Jindra Lackoren webgunetik fusilatuta.

stops_1 <- indizeak_1

distance_route_1 <- data.frame(
  idx_origin      = stops_1[-length(stops_1)],      # Denak, azkena izan ezik
  idx_destination = stops_1[-1]                     # Denak, lehena izan ezik
)

distance_result_1 <-  distance_route_1 %>%
  inner_join(routes_1,
             by = c("idx_origin", "idx_destination")) %>%
  st_as_sf()

leaflet(distance_result_1) %>%
  addProviderTiles("CartoDB.Positron") %>%
  addPolylines(color = "gray",
               popup = ~route_name) %>%
  addCircleMarkers(data = kluster1,
```

```
                          fillColor = "red",
                          radius = 5,
                          stroke = F,
                          fillOpacity = .75,
                          label = ~ Udalerriak,
                          labelOptions = labelOptions(
                            noHide = TRUE,
                            direction = "top",
                            textsize = "12px",
                            opacity = 0.60))

    # 4.2 Bigarren klusterra (horia: kluster=6)

        (…)[4]

# 5. Klusterren arteko rutak egin.

        konexioa_1_2 <- hereR::route(
          origin = herriak_geo[herriak_geo$Udalerriak == "Bera", ],
          destination = herriak_geo[herriak_geo$Udalerriak == "Lesaka", ],
          transport_mode = "pedestrian"
        ) %>%
          st_zm()

        konexioa_2_3 <- (…)[5]

# 6. Irudikatu dezagun mapa finala.

    # Ibilbide finala lotu.

    ibilbide_finala <- dplyr::bind_rows(
      distance_result_1,
      konexioa_1_2,
      distance_result_2,
      konexioa 2 3,
      distance result_3,
      konexioa 3 4,
      distance_result_4,
      konexioa_4_5,
      distance_result_5,
      konexioa_5_6,
      distance_result_6,
      konexioa_6_7,
      distance_result_7,
      konexioa_7_8,
      distance_result_8,
      konexioa_8_9,
      distance_result_9
    )

    # Irudikatzen dugu mapa finala.

    leaflet(ibilbide_finala) %>%
      addProviderTiles("CartoDB.Positron") %>%
      addPolylines(
        color = "GoldenRod",
        weight = 4,
        opacity = 0.8
      ) %>%
      addCircleMarkers(
        data = herriak_geo,
        fillColor = "red",
        radius = 5,
        stroke = FALSE,
        fillOpacity = 0.75,
        label = ~Udalerriak
      )
```

[4] Dena errepikatu 9 aldiz, bat kluster bakoitzeko.

[5] Dena errepikatu 8 aldiz, bat klusterren arteko lotura bakoitzeko.

6.2 Sortutako ibilbidea (handian)