**Introduction**

Luxembourg, as any proper European country, is full of castles. According to Wikipedia,

“By some optimistic estimates, there are as many as 130 castles in Luxembourg but more realistically  
there are probably just over a hundred, although many of these could be considered large residences  
or manor houses rather than castles”.

I see the editors are probably German or French, calling our castles *manor houses*! They only say  
that because Luxembourg is small, so our castles must be small too, right?

Banter aside, with that many castles, what is the best way to visit them all? And by best way I mean  
shortest way. This is a classical **Travelling salesman problem**. To solve this, I need the following elements:

* A list of castles to visit, with their coordinates
* The distances between these castles to each other
* A program to solve the TSP

Let’s start by loading some packages:

library(tidyverse)

library(magrittr)

library(rvest)

library(curl)

library(brotools)

library(RJSONIO)

library(TSP)

library(ggimage)

First step; scrape the data.

**Scraping the data (that’s the data science part)**

Let’s start by having a list of castles. For this, I go to the French Wikipedia page of  
[Luxembourguish castles](https://fr.wikipedia.org/wiki/Liste_de_ch%C3%A2teaux_luxembourgeois).

The Luxembourguish page is more [exhaustive](https://lb.wikipedia.org/wiki/L%C3%ABscht_vun_de_L%C3%ABtzebuerger_Buergen_a_Schl%C3%A4sser),  
but the names are in Luxembourguish, and I doubt that  
OpenStreetMap, which I’ll use to get the coordinates, understands Luxembourguish.

This list has around 50 castles, a reasonable amount of castles. Scraping the table is quite easy:

page <- read\_html("https://fr.wikipedia.org/wiki/Liste\_de\_ch%C3%A2teaux\_luxembourgeois")

castles <- page %>%

html\_node(".wikitable") %>%

html\_table(fill = TRUE) %>%

select(Nom, Localité) %>%

mutate(query = paste0(Nom, ", ", Localité))

I also add a query column which concatenates the name of the castle (“Nom”) to where it is found  
(“Localité”). The query should be a better choice that simply the castle name to get the coordinates.

Now, I need to add the coordinates to this data frame. For this, I use a function I found online  
that gets the coordinates from OpenStreetMap:

## geocoding function using OSM Nominatim API

## details: http://wiki.openstreetmap.org/wiki/Nominatim

## made by: D.Kisler

#https://datascienceplus.com/osm-nominatim-with-r-getting-locations-geo-coordinates-by-its-address/

nominatim\_osm <- function(address = NULL){

if(suppressWarnings(is.null(address)))

return(data.frame())

tryCatch(

d <- jsonlite::fromJSON(

gsub('\\@addr\\@', gsub('\\s+', '\\%20', address),

'http://nominatim.openstreetmap.org/search/@addr@?format=json&addressdetails=0&limit=1')

), error = function(c) return(data.frame())

)

if(length(d) == 0) return(data.frame())

return(data.frame(lon = as.numeric(d$lon), lat = as.numeric(d$lat)))

}

I can now easily add the coordinates by mapping the nominatim\_osm() function to the  
query column I built before:

castles\_osm <- castles %>%

mutate(geolocation = map(query, nominatim\_osm))

Let’s take a look at castles\_osm:

head(castles\_osm)

## Nom Localité

## 1 Château d'Ansembourg Ansembourg

## 2 Nouveau Château d'Ansembourg Ansembourg

## 3 Château d'Aspelt Aspelt

## 4 Château de Beaufort Beaufort

## 5 Château de Beggen Dommeldange

## 6 Château de Colmar-Berg Colmar-Berg

## query geolocation

## 1 Château d'Ansembourg, Ansembourg 6.046748, 49.700693

## 2 Nouveau Château d'Ansembourg, Ansembourg 6.04760, 49.70085

## 3 Château d'Aspelt, Aspelt 6.222653, 49.524822

## 4 Château de Beaufort, Beaufort 2.757293, 43.297466

## 5 Château de Beggen, Dommeldange 6.137765, 49.643383

## 6 Château de Colmar-Berg, Colmar-Berg 6.087944, 49.814687

I now clean the data. There were several mistakes or castles that where not found, which I added  
manually. I did not notice these mistakes immediately, but when I computed the distances matrix I  
notices several inconsistencies; 0’s in positions other than the diagonal, as well as NAs. So I went  
back to the raw data and corrected what was wrong, this time by looking at Google Maps. Thankfully  
there were not that many mistakes. Below the whole workflow:

# Little helper function to clean the lon and lat columns

extract\_numbers <- function(string){

str\_extract\_all(string, "\\d+", simplify = TRUE) %>%

paste0(collapse = ".")

}

castles <- castles\_osm %>%

mutate(geolocation =

ifelse(Nom == "Château de Wintrange", "6.3517223, 49.5021975", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Septfontaines, Rollingergrund", "6.1028634, 49.6257147", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Septfontaines", "5.9617443, 49.7006292", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Senningen", "6.2342581, 49.6464632", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Schauwenburg", "6.0478341, 49.6110245", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Schuttbourg", "5.8980951, 49.7878706", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Meysembourg", "6.1864882, 49.7704348", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Mamer", "6.0232432, 49.6262397", geolocation)) %>%

mutate(geolocation =

ifelse(Nom == "Château de Born", "6.5125214, 49.7611168", geolocation)) %>%

# Found chateau de Betzdorf in Germany, not Luxembourg:

mutate(geolocation = ifelse(Nom == "Château Betzdorf", "6.330278, 49.694167", geolocation)) %>%

# Found château de Clemency in France, not Luxembourg:

mutate(geolocation = ifelse(Nom == "Château de Clemency", "5.874167, 49.598056", geolocation)) %>%

separate(geolocation, into = c("lon", "lat"), sep = ",") %>%

filter(!is.na(lat)) %>%

mutate(lon = map(lon, extract\_numbers)) %>%

mutate(lat = map(lat, extract\_numbers)) %>%

# Château de Beaufort found is in southern France, not the one in lux

# Château de Dudelange is wrong (same as Bettembourg)

# Château de Pétange is wrong (same as Differdange)

# Château d'Urspelt is wrong (same as Clervaux)

# Château d'Hesperange is wrong (same as Palais Grand-Ducal)

mutate(lon = ifelse(Nom == "Château de Beaufort", "6.2865176", lon),

lat = ifelse(Nom == "Château de Beaufort", "49.8335306", lat)) %>%

mutate(lon = ifelse(Nom == "Château Dudelange", "6.0578438", lon),

lat = ifelse(Nom == "Château Dudelange", "49.4905049", lat)) %>%

mutate(lon = ifelse(Nom == "Château de Pétange", "6.105703", lon),

lat = ifelse(Nom == "Château de Pétange", "49.7704746", lat)) %>%

mutate(lon = ifelse(Nom == "Château d' Urspelt", "6.043375", lon),

lat = ifelse(Nom == "Château d' Urspelt", "50.075342", lat)) %>%

mutate(lon = ifelse(Nom == "Château d'Hesperange", "6.1524302", lon),

lat = ifelse(Nom == "Château d'Hesperange", "49.573071", lat)) %>%

mutate(latlon = paste0(lat, ",", lon)) %>%

mutate(lon = as.numeric(lon), lat = as.numeric(lat))

In the end, I have 48 castles, 2 of them were not found neither by OpenStreetMap nor  
Google Maps.

Now I can get the distances matrix. For this, I opened an account at [Graphhopper](https://www.graphhopper.com/)  
and used their [Matrix API](https://graphhopper.com/api/1/docs/matrix/#matrix-api). When you open  
a free account, you get a standard account for free for two weeks, which was perfect for this little  
exercise.

To use the Matrix API you can make a call with curl from your terminal, like this:

curl "https://graphhopper.com/api/1/matrix?point=49.932707,11.588051&point=50.241935,10.747375&point=50.118817,11.983337&type=json&vehicle=car&debug=true&out\_array=weights&out\_array=times&out\_array=distances&key=[YOUR\_KEY]"

To use this from R, I use the {curl} package and the curl\_download() function to download and  
write the output to disk.

I built the url like this. First, the “points” part:

points <- paste(castles$latlon, collapse = "&point=")

Click if you want to see the “points” string

points

## [1] "49.70069265,6.04674779400653&point=49.7008533,6.04759957386294&point=49.5248216,6.2226525964455&point=49.8335306,6.2865176&point=49.64338295,6.1377647435619&point=49.8146867,6.08794389490417&point=49.5749356,5.9841033&point=49.5173197,6.09641390513718&point=49.8760687,6.22027097982788&point=49.694167,6.330278&point=49.7611168,6.5125214&point=49.70256665,6.21740997690437&point=49.905581,6.07950107769784&point=49.9127745,6.13764166375989&point=49.598056,5.874167&point=50.0544533,6.03028463135369&point=49.75943095,5.82586812555896&point=49.52132545,5.88917535225117&point=49.6345518,6.1386377&point=49.4905049,6.0578438&point=49.8600716,6.11163732377525&point=49.9110418,5.93440053120085&point=49.7475976,6.18681116161273&point=49.61092115,6.13288873913352&point=49.573071,6.1524302&point=49.71207855,6.05156617599082&point=49.6694157,5.9496767&point=49.7704143,6.18888954785334&point=49.6262397,6.0232432&point=49.7478579,6.10315847283333&point=49.7704348,6.1864882&point=49.6328906,6.25941956000154&point=49.7704746,6.105703&point=49.54325715,5.9262570638974&point=49.470114,6.3658507&point=49.719675,6.09334070925783&point=49.7878706,5.8980951&point=49.6110245,6.0478341&point=49.6464632,6.2342581&point=49.7006292,5.9617443&point=49.6257147,6.1028634&point=49.556964,6.380786&point=50.075342,6.043375&point=49.7682266,5.9803414&point=49.9348908,6.20279648757301&point=49.6604088,6.1337864&point=49.9664662,5.93854270968922&point=49.5021975,6.3517223"

Then, I added my key, and pasted these elements together to form the correct url:

my\_key <- "my\_key\_was\_here"

url <- paste0("https://graphhopper.com/api/1/matrix?point=", points, "&type=json&vehicle=car&debug=true&out\_array=weights&out\_array=times&out\_array=distances&key=", my\_key)

Then, I get the matrix like this:

castles\_dist <- "distances\_graphhopper.json"

curl\_download(url, castles\_dist)

Let’s take a look at the object:

distances <- castles\_dist$distances

Click if you want to see the distance object

distances

## [[1]]

## [1] 0 48 46364 38416 16619 20387 19617 31990 31423 46587 60894

## [12] 19171 36961 30701 25734 52929 22843 42618 18138 40015 24860 39395

## [23] 17163 18938 28107 2570 10882 16888 12302 9350 16599 32025 14369

## [34] 40780 56004 6069 17602 16112 31552 8180 14523 49431 53199 13354

## [45] 43769 15868 46237 53617

##

## [[2]]

## [1] 48 0 46412 38464 16667 20435 19665 32038 31471 46635 60942

## [12] 19219 37009 30749 25781 52977 22890 42665 18186 40063 24908 39443

## [23] 17211 18986 28155 2618 10930 16936 12350 9398 16647 32073 14417

## [34] 40828 56052 6116 17650 16160 31599 8228 14571 49478 53247 13402

## [45] 43817 15916 46285 53665

##

## [[3]]

## [1] 46900 46947 0 48698 30281 45548 30424 17056 56584 31187 52215

## [12] 28799 62122 55862 39130 78090 66375 33585 23961 21009 50021 64556

## [23] 35740 25853 10852 49283 43218 43052 37317 36283 42763 17250 39530

## [34] 31748 14513 33919 60798 31885 22872 44629 37023 14605 78360 44602

## [45] 68930 26320 71398 12126

##

## [[4]]

## [1] 38214 38261 48754 0 34949 23582 55661 48848 11274 29579 26880

## [12] 25631 32853 20633 67818 46577 47882 65319 33355 58499 26540 40460

## [23] 24359 38061 43577 37674 61661 21704 55760 29822 25030 36698 27956

## [34] 63482 72862 32945 42596 50328 34302 42495 38740 46782 46847 35141

## [45] 20752 33520 47302 56789

##

## [[5]]

## [1] 16494 16541 25311 35192 0 25375 19477 16687 36411 20939 42124

## [12] 13107 41949 35689 27116 57917 44116 30670 1432 26337 29848 44383

## [23] 18673 5767 11224 18877 20958 23922 15058 16110 23633 13255 19357

## [34] 28833 40700 15310 30392 12024 12782 20050 7178 30661 58187 24429

## [45] 48757 2511 51225 33346

##

## [[6]]

## [1] 18468 18516 43459 23632 29633 0 33496 43553 15352 45965 60272

## [12] 23583 20890 14630 39612 36858 27623 60024 28268 53203 8789 21614

## [23] 17217 32765 38282 17929 29164 13853 27119 8892 15798 31403 7026

## [34] 58187 67567 13199 22337 27919 30929 22750 26330 48809 37128 14882

## [45] 27698 21128 28456 51494

##

## [[7]]

## [1] 19645 19693 30022 55860 21434 35353 0 17740 46389 45070 59377

## [12] 35962 51927 45668 10941 67895 36650 11975 16038 16675 39826 49570

## [23] 42902 13747 19018 22029 13093 32857 7837 24321 32568 30508 29335

## [34] 8659 39662 20998 33544 9053 30034 17958 19337 40306 68165 29296

## [45] 58736 20683 59099 37275

##

## [[8]]

## [1] 33194 33242 17113 48244 16576 45094 16196 0 56130 37454 51761

## [12] 28345 61668 55408 26667 77635 52670 21122 15199 8546 49567 64102

## [23] 35286 12148 11167 35578 29512 42597 23612 35829 42308 22891 39076

## [34] 19285 26753 30159 47093 12671 22418 30923 23317 27397 77906 44148

## [45] 68476 25866 70944 24366

##

## [[9]]

## [1] 34049 34097 59040 11039 45215 18025 49077 59134 0 35131 34289

## [12] 25588 18148 8956 55193 34944 47717 75605 43849 68785 11835 33390

## [23] 19644 48347 53863 33510 44745 17010 42701 26274 22029 46984 22791

## [34] 73768 83148 28780 42477 43501 46511 38331 41912 64390 35215 29033

## [45] 11504 36710 37265 67075

##

## [[10]]

## [1] 40768 40815 31200 29561 26887 47108 44877 38064 35204 0 24550

## [12] 11501 46805 40546 57034 62773 59493 54535 25521 47714 51581 66116

## [23] 18215 27276 32792 40228 50876 23464 44975 37844 23175 16279 41090

## [34] 52698 33741 35479 54253 39544 10472 52287 30906 22876 63043 46162

## [45] 44908 19378 72959 30101

##

## [[11]]

## [1] 54812 54860 52014 26837 40931 61152 58921 52108 34149 24114 0

## [12] 30135 56417 42752 71078 69452 73537 68579 39565 61758 65626 80160

## [23] 33264 41320 46836 54272 64920 34961 59020 51888 38287 30323 55134

## [34] 66742 46672 49523 68297 53588 33981 66331 44951 37998 69722 60206

## [45] 39488 41924 87003 44227

##

## [[12]]

## [1] 19189 19237 28715 23758 13122 25545 35622 28809 25568 11495 27547

## [12] 0 42119 35859 47779 58087 37930 45280 12217 38459 30018 44553

## [23] 8427 18021 23537 18649 41621 13676 35721 16280 13387 16659 19527

## [34] 43443 52822 13901 32690 30289 10679 28544 17602 34064 58357 24599

## [45] 34791 11692 51395 36749

##

## [[13]]

## [1] 36813 36860 61804 32740 47978 20788 51841 61898 18594 46628 54032

## [12] 41927 0 11355 57957 25913 31393 78369 46612 71548 10254 18035

## [23] 33039 51110 56626 36273 47508 29674 45464 29037 31619 49747 26873

## [34] 76531 85911 31544 28499 46264 49274 41094 44675 67153 26183 28043

## [45] 22973 39473 21910 69838

##

## [[14]]

## [1] 30553 30601 55544 20685 41718 14528 45581 55638 9008 40368 42945

## [12] 35668 11355 0 51697 26249 44221 72109 40353 65288 8339 26597

## [23] 26779 44850 50367 30014 41249 23415 39204 22778 25360 43488 20613

## [34] 70272 79652 25284 38981 40004 43014 34835 38415 60894 26519 25537

## [45] 13328 33213 30473 63579

##

## [[15]]

## [1] 25606 25654 38728 67712 27136 41314 10938 26445 52350 56922 71229

## [12] 37346 57888 51628 0 73856 27918 10658 28655 25381 45787 55480

## [23] 42912 31851 30870 27990 14521 38818 16023 30282 38529 42360 35296

## [34] 9692 48367 26996 35203 18968 41886 19386 25040 49012 74126 25555

## [45] 64696 26385 63610 45980

##

## [[16]]

## [1] 52597 52644 77588 46234 63762 36572 67625 77681 34889 62412

## [11] 69484 57711 25729 26142 73741 0 51707 94153 62396 87332

## [21] 30382 29845 48823 66894 72410 52057 63292 45458 61248 44821

## [31] 47403 65531 42657 92315 101695 47328 48813 62048 65058 56878

## [41] 60459 82937 3927 48357 26809 55257 16900 85622

##

## [[17]]

## [1] 22742 22790 66383 47823 44393 27506 36726 52009 45012 59978 74285

## [12] 37595 31384 44290 27751 49817 0 38346 45912 60034 38449 23624

## [23] 32592 49107 48126 21474 20735 31486 33280 24718 31198 45415 24781

## [34] 37380 76023 27212 9559 36224 44942 16608 42297 71343 51639 13101

## [45] 57358 43642 33153 73636

##

## [[18]]

## [1] 43466 43513 32742 64592 35685 61442 12026 20459 72478 53802 68109

## [12] 44693 78016 71756 10711 93983 38600 0 31781 19395 65915 80450

## [23] 51634 28730 27749 45849 20641 58945 19674 52177 58656 39239 55424

## [34] 6452 42381 40430 45885 20422 38766 25506 33588 43026 94253 53116

## [45] 84824 34934 87292 39994

##

## [[19]]

## [1] 17092 17140 23308 33262 1432 30429 15982 15299 41465 25814 40121

## [12] 12039 47003 40743 27715 62970 44714 29283 0 24950 34902 49437

## [23] 17605 4379 9837 19476 21557 22854 15656 21164 22565 11252 24411

## [34] 27446 39313 18800 37574 12622 10778 20648 5791 28657 63241 29483

## [45] 53811 3497 56279 31342

##

## [[20]]

## [1] 40369 40417 21028 58519 26851 55368 16853 5415 66404 47729 62035

## [12] 38620 71942 65683 25560 87910 59845 20015 25473 0 59842 74376

## [23] 45561 22423 21442 42753 36687 52872 30786 46104 52583 33166 49351

## [34] 18178 30668 37333 54268 25355 32693 38098 30492 31312 88180 50019

## [45] 78751 31837 81219 28281

##

## [[21]]

## [1] 25435 25483 50426 23657 36601 9410 40463 50520 9511 52933 67240

## [12] 30550 10254 8470 46579 30697 31384 66991 35235 60171 0 22368

## [23] 19572 39732 45249 24896 36131 16208 34087 17660 18153 38370 15495

## [34] 65154 74534 20166 26099 34887 37897 29717 33298 55776 30967 18643

## [45] 21538 28096 29210 58461

##

## [[22]]

## [1] 39601 39649 64592 40786 50767 21730 50009 64686 34013 67099 81406

## [12] 44716 18356 26775 55533 27246 27163 81157 49401 74337 22735 0

## [23] 35431 53898 59415 39062 38903 32067 42694 31826 34012 52536 26191

## [34] 79320 88700 34332 24269 49053 52063 31939 47464 69942 29069 20615

## [45] 35072 42262 10583 72627

##

## [[23]]

## [1] 17968 18016 35386 24320 18687 19877 42293 35480 19624 18209 33230

## [12] 8427 33133 26873 42921 49101 32654 51951 17782 45130 21032 35141

## [23] 0 21756 30208 17139 28664 5714 26043 8833 5426 23330 10461

## [34] 50114 59493 12123 27413 26843 20737 23268 23168 40735 49371 19323

## [45] 29874 17258 41983 43420

##

## [[24]]

## [1] 17512 17560 15422 36059 4428 33226 13943 13260 44262 27280 41587

## [12] 14836 49800 43540 24703 65768 41703 27243 3050 22910 37699 52234

## [23] 20402 0 7797 19895 18546 25651 12645 23961 25362 12718 27208

## [34] 25406 37273 16329 31410 8506 12244 19956 6213 26147 66038 32280

## [45] 56608 6317 59076 34886

##

## [[25]]

## [1] 27872 27920 10877 44424 10352 41273 18191 11042 52309 33634 47940

## [12] 24525 57847 51588 30348 73815 47348 27849 8975 20692 45747 60281

## [23] 31466 5924 0 30256 24190 38777 18290 32009 38488 19071 35255

## [34] 26012 35056 24837 41771 12858 18598 25601 17995 21917 74085 40327

## [45] 64656 19340 67124 21432

##

## [[26]]

## [1] 2570 2618 48748 37876 19003 19847 22001 34374 30884 46048 60355

## [12] 18632 36421 30162 28117 52389 21574 45001 28350 42399 24321 32360

## [23] 17165 21322 30491 0 13266 16059 14686 8521 15770 31485 13830

## [34] 43164 58387 5529 16334 18496 31012 10564 16906 48891 52659 12086

## [45] 43230 18252 45698 56000

##

## [[27]]

## [1] 10882 10930 43104 61689 21113 31083 13102 28730 42119 50899 65206

## [12] 31324 47657 41397 14188 63624 20735 20742 22632 36755 35556 38436

## [23] 27859 25828 24847 13266 0 27584 10000 20046 27295 36337 25065

## [34] 19286 52743 16764 22410 12945 35863 6736 19017 48063 63894 18161

## [45] 54465 20362 47965 50356

##

## [[28]]

## [1] 16863 16910 42857 21661 23936 13872 32894 42951 16990 23458 34927

## [12] 13676 29769 23509 39010 45736 31548 59422 23031 52601 17668 31777

## [23] 5714 27005 37680 16033 27558 0 26517 7728 640 30801 9331

## [34] 57585 66965 12597 26308 27318 25985 22162 25728 48207 46006 18217

## [45] 26509 22507 38619 50892

##

## [[29]]

## [1] 12254 12302 36928 55514 14937 28982 7761 22554 40018 44724 59031

## [12] 25148 45556 39296 16297 61524 33297 19546 16457 30579 33455 42179

## [23] 25758 19652 18671 14638 10140 26486 0 17950 26197 30161 22964

## [34] 16230 46568 14086 26153 3631 29688 11548 12841 41888 61794 21904

## [45] 52364 14186 51708 44180

##

## [[30]]

## [1] 9350 9398 36159 28322 22333 12073 24364 36253 23110 38665 52972

## [12] 16282 28647 22388 30480 44615 24850 52724 20968 45903 16547 31081

## [23] 8859 25465 30981 8521 20046 7753 17987 0 7465 24103 5802

## [34] 50887 60266 4067 19610 18787 23629 17344 17198 41508 44885 11519

## [45] 35456 13828 37924 44193

##

## [[31]]

## [1] 16574 16622 42568 24986 23647 18483 32605 42662 21985 23169 38253

## [12] 13387 31714 25454 38721 47681 31259 59133 22742 52312 19613 33722

## [23] 5426 26716 37391 15744 27269 640 26229 7439 0 30512 9042

## [34] 57296 66676 12308 26019 27029 25696 21873 25440 47918 47951 17928

## [45] 28454 22218 40564 50603

##

## [[32]]

## [1] 26503 26551 16931 35994 12622 32844 30612 23800 43880 15938 30245

## [12] 16095 49418 43158 42770 65386 45229 40271 11257 33450 37317 51852

## [23] 23036 13012 18528 25964 36612 30348 30711 23579 30059 0 26826

## [34] 38433 28491 21215 39989 25280 5038 38023 16642 16486 65656 31898

## [45] 56226 13616 58694 22401

##

## [[33]]

## [1] 14182 14230 39173 27991 25347 7318 29210 39267 22746 41679 55986

## [12] 19297 27681 21422 35326 43649 24862 55738 23982 48917 15581 26003

## [23] 10461 28479 33996 13643 24878 9331 22833 5790 9042 27117 0

## [34] 53901 63281 8913 19576 23633 26643 18464 22044 44523 43919 12121

## [45] 29215 16842 35531 47208

##

## [[34]]

## [1] 28166 28214 31292 63142 34235 59992 8710 19009 71028 52352 66659

## [12] 43243 76566 70306 9382 92533 36885 6378 30331 17945 64465 64447

## [23] 50184 27280 26299 30550 18965 57495 16357 50727 57206 37789 53974

## [34] 0 40931 38980 44170 17105 37316 23829 32138 41576 92803 34921

## [45] 83374 33484 85842 38544

##

## [[35]]

## [1] 56656 56703 14836 71705 40037 68555 40180 26812 79591 33969

## [11] 46710 51806 85129 78869 48886 101097 76131 43341 38660 30765

## [21] 73028 87563 58747 35609 34628 59039 52974 66059 47073 59291

## [31] 65770 28534 62537 41504 0 53620 70554 41641 45879 54384

## [41] 46778 10804 101367 67609 91937 49327 94405 4470

##

## [[36]]

## [1] 6069 6116 38793 33128 15430 15099 20926 29765 26135 41299 55606

## [12] 13883 31673 25413 27213 47641 27484 40392 23602 37789 19572 34107

## [23] 11875 28099 33615 5529 16764 12602 14123 4067 12314 26737 9081

## [34] 38555 53778 0 22244 14923 26263 14062 13334 44142 47911 14153

## [45] 38481 11296 40949 46827

##

## [[37]]

## [1] 17599 17647 60259 42618 30514 22302 33512 45885 39869 54835 69142

## [12] 32452 28491 39147 35036 46923 9559 56513 37137 53910 33306 20731

## [23] 27449 41635 42002 16331 22406 26343 26197 19575 26054 40272 19576

## [34] 54675 69899 22068 0 30007 39799 15442 28418 57678 48746 7897

## [45] 52215 29998 30260 67512

##

## [[38]]

## [1] 16015 16063 31108 49694 12129 29728 8933 12719 40764 38904 53211

## [12] 29795 46302 40042 19294 62270 36294 27361 13648 24759 34201 48736

## [23] 26504 8314 12851 18398 13137 27231 3723 18696 26943 24341 23710

## [34] 17000 40748 14832 29913 0 23868 14545 10032 36068 62540 25665

## [45] 53110 11378 55578 38360

##

## [[39]]

## [1] 25665 25713 22868 35156 11784 32006 29774 22961 43042 10466 34023

## [12] 15257 48580 42320 41932 64548 44391 39433 10419 32612 36479 51014

## [23] 22198 12174 17690 25126 35774 29510 29873 22741 29221 5081 25988

## [34] 37595 46975 20377 39151 24442 0 37185 15804 21298 64818 31060

## [45] 55388 12778 57856 30902

##

## [[40]]

## [1] 8180 8228 44691 42719 20175 24690 17967 30317 35726 52486 66793

## [12] 28309 41264 35005 19052 57232 16608 25606 21695 38341 29164 31472

## [23] 23306 27415 26433 10564 6736 22201 11607 17344 21912 37924 18672

## [34] 24151 54330 14062 15446 14551 37450 0 18079 49650 57502 11198

## [45] 48073 19424 41001 51943

##

## [[41]]

## [1] 14500 14548 37450 38853 8571 28213 17483 23076 39249 31405 45712

## [12] 17630 44787 38527 25122 60755 42122 33703 5844 31101 32686 47221

## [23] 23196 6938 19193 16883 18965 25716 13064 17181 25428 16842 22195

## [34] 31866 47090 13317 28398 10030 16369 18056 0 34248 61025 24150

## [45] 51595 7820 54063 44703

##

## [[42]]

## [1] 44607 44655 14655 46763 30726 50948 40494 27126 61984 22848 38359

## [12] 34199 67522 61262 49200 83489 76445 43655 29360 31079 55421 69956

## [23] 41140 24844 22014 44067 53288 48451 47387 41683 48162 16486 44930

## [34] 41818 10865 39319 58092 41955 21254 54699 47093 0 83759 50002

## [45] 74330 31720 76798 7225

##

## [[43]]

## [1] 53049 53096 78040 46686 64214 37024 68076 78133 35341 62864

## [11] 69936 58163 26181 26593 74193 3927 51887 94604 62848 87784

## [21] 30834 37251 49275 67346 72862 52509 63744 45910 61700 45273

## [31] 47855 65983 43109 92767 102147 47780 48993 62500 65510 57330

## [41] 60911 83389 0 48032 27261 55709 18722 86074

##

## [[44]]

## [1] 13481 13529 44241 35163 30415 14847 29395 44335 31781 46747 61054

## [12] 24364 28035 31060 30271 46467 13182 52395 29050 49792 25219 20275

## [23] 19361 33547 39063 12213 18288 18256 22079 11488 17967 32185 12121

## [34] 50557 68348 13981 7897 25889 31711 11325 24300 49590 53557 0

## [45] 44128 21910 29803 52275

##

## [[45]]

## [1] 40446 40494 65437 20517 51611 24421 55474 65531 11556 44883 39488

## [12] 34762 24423 13328 61590 26917 54113 82002 50246 75181 18231 34751

## [23] 29845 54743 60259 39906 51142 26480 49097 32670 28426 53381 29188

## [34] 80164 89544 35177 48873 49897 52907 44727 48308 70786 27187 35430

## [45] 0 43106 33848 73471

##

## [[46]]

## [1] 15720 15767 25996 33807 2532 23092 18703 24296 34128 19554 42809

## [12] 11722 39666 33406 26342 55633 43342 34923 4080 32321 27565 42100

## [23] 17288 7632 20818 18103 20185 22537 14284 13827 22248 13940 17074

## [34] 33086 48309 11317 30237 11250 11347 19276 7865 31345 55903 22146

## [45] 46474 0 48942 34030

##

## [[47]]

## [1] 48326 48373 73317 53314 59491 30454 61401 73410 39547 75823 76564

## [12] 53440 23890 33221 63609 16972 35356 74204 58125 83061 30123 12418

## [23] 44155 62623 68139 47786 50295 40791 54086 40550 42736 61260 37582

## [34] 73238 97424 43057 32463 57777 60787 43331 56188 78666 18794 32007

## [45] 33889 50986 0 81351

##

## [[48]]

## [1] 53987 54035 12168 56733 37369 53583 37512 24144 64619 31546 44287

## [12] 36834 70157 63897 46218 86125 73463 40673 35992 28097 58056 72591

## [23] 43775 32941 31960 56371 50305 51087 44405 44319 50798 22413 47565

## [34] 38836 4459 50952 67886 38973 30907 51716 44110 7225 86395 52637

## [45] 76965 34355 79433 0

distances is a list where the first element is the distances from the first castle to all the others.  
Let’s make it a matrix:

distances\_matrix <- distances %>%

reduce(rbind)

Click if you want to see the distance matrix

distances\_matrix

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]

## out 0 48 46364 38416 16619 20387 19617 31990 31423 46587 60894

## 48 0 46412 38464 16667 20435 19665 32038 31471 46635 60942

## 46900 46947 0 48698 30281 45548 30424 17056 56584 31187 52215

## 38214 38261 48754 0 34949 23582 55661 48848 11274 29579 26880

## 16494 16541 25311 35192 0 25375 19477 16687 36411 20939 42124

## 18468 18516 43459 23632 29633 0 33496 43553 15352 45965 60272

## 19645 19693 30022 55860 21434 35353 0 17740 46389 45070 59377

## 33194 33242 17113 48244 16576 45094 16196 0 56130 37454 51761

## 34049 34097 59040 11039 45215 18025 49077 59134 0 35131 34289

## 40768 40815 31200 29561 26887 47108 44877 38064 35204 0 24550

## 54812 54860 52014 26837 40931 61152 58921 52108 34149 24114 0

## 19189 19237 28715 23758 13122 25545 35622 28809 25568 11495 27547

## 36813 36860 61804 32740 47978 20788 51841 61898 18594 46628 54032

## 30553 30601 55544 20685 41718 14528 45581 55638 9008 40368 42945

## 25606 25654 38728 67712 27136 41314 10938 26445 52350 56922 71229

## 52597 52644 77588 46234 63762 36572 67625 77681 34889 62412 69484

## 22742 22790 66383 47823 44393 27506 36726 52009 45012 59978 74285

## 43466 43513 32742 64592 35685 61442 12026 20459 72478 53802 68109

## 17092 17140 23308 33262 1432 30429 15982 15299 41465 25814 40121

## 40369 40417 21028 58519 26851 55368 16853 5415 66404 47729 62035

## 25435 25483 50426 23657 36601 9410 40463 50520 9511 52933 67240

## 39601 39649 64592 40786 50767 21730 50009 64686 34013 67099 81406

## 17968 18016 35386 24320 18687 19877 42293 35480 19624 18209 33230

## 17512 17560 15422 36059 4428 33226 13943 13260 44262 27280 41587

## 27872 27920 10877 44424 10352 41273 18191 11042 52309 33634 47940

## 2570 2618 48748 37876 19003 19847 22001 34374 30884 46048 60355

## 10882 10930 43104 61689 21113 31083 13102 28730 42119 50899 65206

## 16863 16910 42857 21661 23936 13872 32894 42951 16990 23458 34927

## 12254 12302 36928 55514 14937 28982 7761 22554 40018 44724 59031

## 9350 9398 36159 28322 22333 12073 24364 36253 23110 38665 52972

## 16574 16622 42568 24986 23647 18483 32605 42662 21985 23169 38253

## 26503 26551 16931 35994 12622 32844 30612 23800 43880 15938 30245

## 14182 14230 39173 27991 25347 7318 29210 39267 22746 41679 55986

## 28166 28214 31292 63142 34235 59992 8710 19009 71028 52352 66659

## 56656 56703 14836 71705 40037 68555 40180 26812 79591 33969 46710

## 6069 6116 38793 33128 15430 15099 20926 29765 26135 41299 55606

## 17599 17647 60259 42618 30514 22302 33512 45885 39869 54835 69142

## 16015 16063 31108 49694 12129 29728 8933 12719 40764 38904 53211

## 25665 25713 22868 35156 11784 32006 29774 22961 43042 10466 34023

## 8180 8228 44691 42719 20175 24690 17967 30317 35726 52486 66793

## 14500 14548 37450 38853 8571 28213 17483 23076 39249 31405 45712

## 44607 44655 14655 46763 30726 50948 40494 27126 61984 22848 38359

## 53049 53096 78040 46686 64214 37024 68076 78133 35341 62864 69936

## 13481 13529 44241 35163 30415 14847 29395 44335 31781 46747 61054

## 40446 40494 65437 20517 51611 24421 55474 65531 11556 44883 39488

## 15720 15767 25996 33807 2532 23092 18703 24296 34128 19554 42809

## 48326 48373 73317 53314 59491 30454 61401 73410 39547 75823 76564

## 53987 54035 12168 56733 37369 53583 37512 24144 64619 31546 44287

## [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]

## out 19171 36961 30701 25734 52929 22843 42618 18138 40015 24860 39395

## 19219 37009 30749 25781 52977 22890 42665 18186 40063 24908 39443

## 28799 62122 55862 39130 78090 66375 33585 23961 21009 50021 64556

## 25631 32853 20633 67818 46577 47882 65319 33355 58499 26540 40460

## 13107 41949 35689 27116 57917 44116 30670 1432 26337 29848 44383

## 23583 20890 14630 39612 36858 27623 60024 28268 53203 8789 21614

## 35962 51927 45668 10941 67895 36650 11975 16038 16675 39826 49570

## 28345 61668 55408 26667 77635 52670 21122 15199 8546 49567 64102

## 25588 18148 8956 55193 34944 47717 75605 43849 68785 11835 33390

## 11501 46805 40546 57034 62773 59493 54535 25521 47714 51581 66116

## 30135 56417 42752 71078 69452 73537 68579 39565 61758 65626 80160

## 0 42119 35859 47779 58087 37930 45280 12217 38459 30018 44553

## 41927 0 11355 57957 25913 31393 78369 46612 71548 10254 18035

## 35668 11355 0 51697 26249 44221 72109 40353 65288 8339 26597

## 37346 57888 51628 0 73856 27918 10658 28655 25381 45787 55480

## 57711 25729 26142 73741 0 51707 94153 62396 87332 30382 29845

## 37595 31384 44290 27751 49817 0 38346 45912 60034 38449 23624

## 44693 78016 71756 10711 93983 38600 0 31781 19395 65915 80450

## 12039 47003 40743 27715 62970 44714 29283 0 24950 34902 49437

## 38620 71942 65683 25560 87910 59845 20015 25473 0 59842 74376

## 30550 10254 8470 46579 30697 31384 66991 35235 60171 0 22368

## 44716 18356 26775 55533 27246 27163 81157 49401 74337 22735 0

## 8427 33133 26873 42921 49101 32654 51951 17782 45130 21032 35141

## 14836 49800 43540 24703 65768 41703 27243 3050 22910 37699 52234

## 24525 57847 51588 30348 73815 47348 27849 8975 20692 45747 60281

## 18632 36421 30162 28117 52389 21574 45001 28350 42399 24321 32360

## 31324 47657 41397 14188 63624 20735 20742 22632 36755 35556 38436

## 13676 29769 23509 39010 45736 31548 59422 23031 52601 17668 31777

## 25148 45556 39296 16297 61524 33297 19546 16457 30579 33455 42179

## 16282 28647 22388 30480 44615 24850 52724 20968 45903 16547 31081

## 13387 31714 25454 38721 47681 31259 59133 22742 52312 19613 33722

## 16095 49418 43158 42770 65386 45229 40271 11257 33450 37317 51852

## 19297 27681 21422 35326 43649 24862 55738 23982 48917 15581 26003

## 43243 76566 70306 9382 92533 36885 6378 30331 17945 64465 64447

## 51806 85129 78869 48886 101097 76131 43341 38660 30765 73028 87563

## 13883 31673 25413 27213 47641 27484 40392 23602 37789 19572 34107

## 32452 28491 39147 35036 46923 9559 56513 37137 53910 33306 20731

## 29795 46302 40042 19294 62270 36294 27361 13648 24759 34201 48736

## 15257 48580 42320 41932 64548 44391 39433 10419 32612 36479 51014

## 28309 41264 35005 19052 57232 16608 25606 21695 38341 29164 31472

## 17630 44787 38527 25122 60755 42122 33703 5844 31101 32686 47221

## 34199 67522 61262 49200 83489 76445 43655 29360 31079 55421 69956

## 58163 26181 26593 74193 3927 51887 94604 62848 87784 30834 37251

## 24364 28035 31060 30271 46467 13182 52395 29050 49792 25219 20275

## 34762 24423 13328 61590 26917 54113 82002 50246 75181 18231 34751

## 11722 39666 33406 26342 55633 43342 34923 4080 32321 27565 42100

## 53440 23890 33221 63609 16972 35356 74204 58125 83061 30123 12418

## 36834 70157 63897 46218 86125 73463 40673 35992 28097 58056 72591

## [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33]

## out 17163 18938 28107 2570 10882 16888 12302 9350 16599 32025 14369

## 17211 18986 28155 2618 10930 16936 12350 9398 16647 32073 14417

## 35740 25853 10852 49283 43218 43052 37317 36283 42763 17250 39530

## 24359 38061 43577 37674 61661 21704 55760 29822 25030 36698 27956

## 18673 5767 11224 18877 20958 23922 15058 16110 23633 13255 19357

## 17217 32765 38282 17929 29164 13853 27119 8892 15798 31403 7026

## 42902 13747 19018 22029 13093 32857 7837 24321 32568 30508 29335

## 35286 12148 11167 35578 29512 42597 23612 35829 42308 22891 39076

## 19644 48347 53863 33510 44745 17010 42701 26274 22029 46984 22791

## 18215 27276 32792 40228 50876 23464 44975 37844 23175 16279 41090

## 33264 41320 46836 54272 64920 34961 59020 51888 38287 30323 55134

## 8427 18021 23537 18649 41621 13676 35721 16280 13387 16659 19527

## 33039 51110 56626 36273 47508 29674 45464 29037 31619 49747 26873

## 26779 44850 50367 30014 41249 23415 39204 22778 25360 43488 20613

## 42912 31851 30870 27990 14521 38818 16023 30282 38529 42360 35296

## 48823 66894 72410 52057 63292 45458 61248 44821 47403 65531 42657

## 32592 49107 48126 21474 20735 31486 33280 24718 31198 45415 24781

## 51634 28730 27749 45849 20641 58945 19674 52177 58656 39239 55424

## 17605 4379 9837 19476 21557 22854 15656 21164 22565 11252 24411

## 45561 22423 21442 42753 36687 52872 30786 46104 52583 33166 49351

## 19572 39732 45249 24896 36131 16208 34087 17660 18153 38370 15495

## 35431 53898 59415 39062 38903 32067 42694 31826 34012 52536 26191

## 0 21756 30208 17139 28664 5714 26043 8833 5426 23330 10461

## 20402 0 7797 19895 18546 25651 12645 23961 25362 12718 27208

## 31466 5924 0 30256 24190 38777 18290 32009 38488 19071 35255

## 17165 21322 30491 0 13266 16059 14686 8521 15770 31485 13830

## 27859 25828 24847 13266 0 27584 10000 20046 27295 36337 25065

## 5714 27005 37680 16033 27558 0 26517 7728 640 30801 9331

## 25758 19652 18671 14638 10140 26486 0 17950 26197 30161 22964

## 8859 25465 30981 8521 20046 7753 17987 0 7465 24103 5802

## 5426 26716 37391 15744 27269 640 26229 7439 0 30512 9042

## 23036 13012 18528 25964 36612 30348 30711 23579 30059 0 26826

## 10461 28479 33996 13643 24878 9331 22833 5790 9042 27117 0

## 50184 27280 26299 30550 18965 57495 16357 50727 57206 37789 53974

## 58747 35609 34628 59039 52974 66059 47073 59291 65770 28534 62537

## 11875 28099 33615 5529 16764 12602 14123 4067 12314 26737 9081

## 27449 41635 42002 16331 22406 26343 26197 19575 26054 40272 19576

## 26504 8314 12851 18398 13137 27231 3723 18696 26943 24341 23710

## 22198 12174 17690 25126 35774 29510 29873 22741 29221 5081 25988

## 23306 27415 26433 10564 6736 22201 11607 17344 21912 37924 18672

## 23196 6938 19193 16883 18965 25716 13064 17181 25428 16842 22195

## 41140 24844 22014 44067 53288 48451 47387 41683 48162 16486 44930

## 49275 67346 72862 52509 63744 45910 61700 45273 47855 65983 43109

## 19361 33547 39063 12213 18288 18256 22079 11488 17967 32185 12121

## 29845 54743 60259 39906 51142 26480 49097 32670 28426 53381 29188

## 17288 7632 20818 18103 20185 22537 14284 13827 22248 13940 17074

## 44155 62623 68139 47786 50295 40791 54086 40550 42736 61260 37582

## 43775 32941 31960 56371 50305 51087 44405 44319 50798 22413 47565

## [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44]

## out 40780 56004 6069 17602 16112 31552 8180 14523 49431 53199 13354

## 40828 56052 6116 17650 16160 31599 8228 14571 49478 53247 13402

## 31748 14513 33919 60798 31885 22872 44629 37023 14605 78360 44602

## 63482 72862 32945 42596 50328 34302 42495 38740 46782 46847 35141

## 28833 40700 15310 30392 12024 12782 20050 7178 30661 58187 24429

## 58187 67567 13199 22337 27919 30929 22750 26330 48809 37128 14882

## 8659 39662 20998 33544 9053 30034 17958 19337 40306 68165 29296

## 19285 26753 30159 47093 12671 22418 30923 23317 27397 77906 44148

## 73768 83148 28780 42477 43501 46511 38331 41912 64390 35215 29033

## 52698 33741 35479 54253 39544 10472 52287 30906 22876 63043 46162

## 66742 46672 49523 68297 53588 33981 66331 44951 37998 69722 60206

## 43443 52822 13901 32690 30289 10679 28544 17602 34064 58357 24599

## 76531 85911 31544 28499 46264 49274 41094 44675 67153 26183 28043

## 70272 79652 25284 38981 40004 43014 34835 38415 60894 26519 25537

## 9692 48367 26996 35203 18968 41886 19386 25040 49012 74126 25555

## 92315 101695 47328 48813 62048 65058 56878 60459 82937 3927 48357

## 37380 76023 27212 9559 36224 44942 16608 42297 71343 51639 13101

## 6452 42381 40430 45885 20422 38766 25506 33588 43026 94253 53116

## 27446 39313 18800 37574 12622 10778 20648 5791 28657 63241 29483

## 18178 30668 37333 54268 25355 32693 38098 30492 31312 88180 50019

## 65154 74534 20166 26099 34887 37897 29717 33298 55776 30967 18643

## 79320 88700 34332 24269 49053 52063 31939 47464 69942 29069 20615

## 50114 59493 12123 27413 26843 20737 23268 23168 40735 49371 19323

## 25406 37273 16329 31410 8506 12244 19956 6213 26147 66038 32280

## 26012 35056 24837 41771 12858 18598 25601 17995 21917 74085 40327

## 43164 58387 5529 16334 18496 31012 10564 16906 48891 52659 12086

## 19286 52743 16764 22410 12945 35863 6736 19017 48063 63894 18161

## 57585 66965 12597 26308 27318 25985 22162 25728 48207 46006 18217

## 16230 46568 14086 26153 3631 29688 11548 12841 41888 61794 21904

## 50887 60266 4067 19610 18787 23629 17344 17198 41508 44885 11519

## 57296 66676 12308 26019 27029 25696 21873 25440 47918 47951 17928

## 38433 28491 21215 39989 25280 5038 38023 16642 16486 65656 31898

## 53901 63281 8913 19576 23633 26643 18464 22044 44523 43919 12121

## 0 40931 38980 44170 17105 37316 23829 32138 41576 92803 34921

## 41504 0 53620 70554 41641 45879 54384 46778 10804 101367 67609

## 38555 53778 0 22244 14923 26263 14062 13334 44142 47911 14153

## 54675 69899 22068 0 30007 39799 15442 28418 57678 48746 7897

## 17000 40748 14832 29913 0 23868 14545 10032 36068 62540 25665

## 37595 46975 20377 39151 24442 0 37185 15804 21298 64818 31060

## 24151 54330 14062 15446 14551 37450 0 18079 49650 57502 11198

## 31866 47090 13317 28398 10030 16369 18056 0 34248 61025 24150

## 41818 10865 39319 58092 41955 21254 54699 47093 0 83759 50002

## 92767 102147 47780 48993 62500 65510 57330 60911 83389 0 48032

## 50557 68348 13981 7897 25889 31711 11325 24300 49590 53557 0

## 80164 89544 35177 48873 49897 52907 44727 48308 70786 27187 35430

## 33086 48309 11317 30237 11250 11347 19276 7865 31345 55903 22146

## 73238 97424 43057 32463 57777 60787 43331 56188 78666 18794 32007

## 38836 4459 50952 67886 38973 30907 51716 44110 7225 86395 52637

## [,45] [,46] [,47] [,48]

## out 43769 15868 46237 53617

## 43817 15916 46285 53665

## 68930 26320 71398 12126

## 20752 33520 47302 56789

## 48757 2511 51225 33346

## 27698 21128 28456 51494

## 58736 20683 59099 37275

## 68476 25866 70944 24366

## 11504 36710 37265 67075

## 44908 19378 72959 30101

## 39488 41924 87003 44227

## 34791 11692 51395 36749

## 22973 39473 21910 69838

## 13328 33213 30473 63579

## 64696 26385 63610 45980

## 26809 55257 16900 85622

## 57358 43642 33153 73636

## 84824 34934 87292 39994

## 53811 3497 56279 31342

## 78751 31837 81219 28281

## 21538 28096 29210 58461

## 35072 42262 10583 72627

## 29874 17258 41983 43420

## 56608 6317 59076 34886

## 64656 19340 67124 21432

## 43230 18252 45698 56000

## 54465 20362 47965 50356

## 26509 22507 38619 50892

## 52364 14186 51708 44180

## 35456 13828 37924 44193

## 28454 22218 40564 50603

## 56226 13616 58694 22401

## 29215 16842 35531 47208

## 83374 33484 85842 38544

## 91937 49327 94405 4470

## 38481 11296 40949 46827

## 52215 29998 30260 67512

## 53110 11378 55578 38360

## 55388 12778 57856 30902

## 48073 19424 41001 51943

## 51595 7820 54063 44703

## 74330 31720 76798 7225

## 27261 55709 18722 86074

## 44128 21910 29803 52275

## 0 43106 33848 73471

## 46474 0 48942 34030

## 33889 50986 0 81351

## 76965 34355 79433 0

Let’s baptize the rows and columns:

colnames(distances\_matrix) <- castles$Nom

rownames(distances\_matrix) <- castles$Nom

Now that we have the data, we can solve the TSP.

**Solving the Travelling salesman problem (that’s the combinatorial optimization part)**

Let’s first coerce the distances\_matrix to an ATSP object, which is needed for the solver.  
ATSP stands for asymmetrical TSP. Asymmetrical because the distances\_matrix is not symmetric,  
meaning that going from Castle A to Castle B is longer than going from Castle B to Castle A (for  
example).

atsp\_castles <- ATSP(distances\_matrix)

I then define a list of all the available methods:

methods <- c("identity", "random", "nearest\_insertion",

"cheapest\_insertion", "farthest\_insertion", "arbitrary\_insertion",

"nn", "repetitive\_nn", "two\_opt")

And solve the problem with all the methods:

solutions <- map(methods, ~solve\_TSP(x = atsp\_castles, method = ., two\_opt = TRUE, rep = 10, two\_opt\_repetitions = 10)) %>%

set\_names(methods)

## Warning: executing %dopar% sequentially: no parallel backend registered

I do this because the results vary depending on the methods, and I want to be exhaustive (solving  
this problem is quite fast, so there’s no reason not to do it):

solutions\_df <- solutions %>%

map\_df(as.numeric)

solutions\_df is a data frame with the order of the castles to visit in rows and the method used  
in columns.

Click if you want to see the solutions

solutions\_df

## # A tibble: 48 x 9

## identity random nearest\_inserti… cheapest\_insert… farthest\_insert…

##

## 1 1 10 37 44 15

## 2 2 11 17 37 27

## 3 36 4 22 17 29

## 4 33 9 47 40 38

## 5 6 45 16 27 41

## 6 44 43 43 15 19

## 7 37 16 13 18 5

## 8 17 47 21 34 46

## 9 22 22 14 7 12

## 10 47 13 45 20 23

## # ... with 38 more rows, and 4 more variables: arbitrary\_insertion ,

## # nn , repetitive\_nn , two\_opt

Now, let’s extract the tour lengths, see which one is the minimum, then plot it.

tour\_lengths <- solutions %>%

map\_dbl(tour\_length)

which.min(tour\_lengths)

## arbitrary\_insertion

## 6

The total length of the tour is 474 kilometers  
(that’s 295 miles). Before plotting the data, let’s  
re-order it according to the solution:

castles\_to\_visit <- castles[pull(solutions\_df, names(which.min(tour\_lengths))), ]

**Plot the solution**

To plot the solution, I first use a data frame I created with the longitude and latitude of  
Luxembourguish communes, from the geojson file available on the  
[OpenData Portal](https://data.public.lu/en/datasets/limites-administratives-du-grand-duche-de-luxembourg/#resource-39af91a6-9ce4-4c18-8271-313b3ad7c7f5).  
I converted it to a data frame because it is easier to manipulate this way. The code to do that is  
in the appendix of this blog post:

communes\_df <- rea[d](https://www.google.com/maps/search/d+Ansembourg++Ansembourg%0D%0A%23%23+3?entry=gmail&source=g)\_csv("communes\_[d](https://www.google.com/maps/search/d+Aspelt++++++Aspelt%0D%0A%23%23+4?entry=gmail&source=g)f.csv")

## Parsed with column specification:

## cols(

## lon = col\_double(),

## lat = col\_double(),

## commune = col\_character()

## )

Now I can use {ggplot2} to create the map with the tour. I use geom\_polygon() to build the map,  
geom\_point() to add the castles, geom\_path() to connect the points according to the solution I  
found and geom\_point() again to highlight the starting castle:

ggplot() +

geom\_polygon(data = communes\_df, aes(x = lon, y = lat, group = commune), colour = "grey", fill = NA) +

geom\_point(data = castles, aes(x = lon, y = lat), colour = "#82518c", size = 3) +

geom\_path(data = castles\_to\_visit, aes(x = lon, y = lat), colour = "#647e0e") +

geom\_point(data = (slice(castles\_to\_visit, 1)), aes(x = lon, y = lat), colour = "white", size = 5) +

theme\_void() +

ggtitle("The shortest tour to visit 48 Luxembourguish castles") +

theme(legend.position = "bottom",

legend.title = element\_blank(),

legend.text = element\_text(colour = "white"),

plot.background = element\_rect("#272b30"),

plot.title = element\_text(colour = "white"))

The white point is the starting point of the tour. As a bonus, let’s do the same plot without  
points, but castles emojis instead (using the {ggimage} package):

ggplot() +

geom\_polygon(data = communes\_df, aes(x = lon, y = lat, group = commune), colour = "grey", fill = NA) +

geom\_emoji(data = castles, aes(x = lon, y = lat, image = "1f3f0")) + # <- this is the hex code for the "european castle" emoji

geom\_path(data = castles\_to\_visit, aes(x = lon, y = lat), colour = "#647e0e") +

theme\_void() +

ggtitle("The shortest tour to visit 48 Luxembourguish castles") +

theme(legend.position = "bottom",

legend.title = element\_blank(),

legend.text = element\_text(colour = "white"),

plot.background = element\_rect("#272b30"),

plot.title = element\_text(colour = "white"))

## Warning: Ignoring unknown parameters: image\_colour

It’s horrible.

Hope you enjoyed! If you found this blog post useful, you might want to follow  
me on [twitter](https://www.twitter.com/brodriguesco) for blog post updates and  
[buy me an espresso](https://www.buymeacoffee.com/brodriguesco).

[Buy me an Espresso](https://www.buymeacoffee.com/brodriguesco)

**Appendix**

The code below converts the geojson that can be downloaded from the  
[OpenData Portal](https://data.public.lu/en/datasets/limites-administratives-du-grand-duche-de-luxembourg/#resource-39af91a6-9ce4-4c18-8271-313b3ad7c7f5)  
to csv. A csv file is easier to handle. I only focus on the communes.

limadmin <- RJSONIO::fromJSON("limadmin.geojson")

communes <- limadmin$communes

extract\_communes <- function(features){

res <- features$geometry$coordinates %>%

map(lift(rbind)) %>%

as.data.frame() %>%

rename(lon = X1,

lat = X2)

res %>%

mutate(commune = features$properties[1])

}

communes\_df <- map(limadmin$communes$features, extract\_communes)

## Steinfort and Waldbredimus special treatment:

steinfort <- limadmin$communes$features[[5]]$geometry$coordinates[[1]] %>%

map(lift(rbind)) %>%

as.data.frame() %>%

rename(lon = X1,

lat = X2) %>%

mutate(commune = "Steinfort")

waldbredimus <- limadmin$communes$features[[44]]$geometry$coordinates[[1]] %>%

map(lift(rbind)) %>%

as.data.frame() %>%

rename(lon = X1,

lat = X2) %>%

mutate(commune = "Waldbredimus")

communes\_df[[5]] <- NULL

communes\_df[[43]] <- NULL

communes\_df <- bind\_rows(communes\_df, list(steinfort, waldbredimus))