SAPIENZA
UNIVERSITÀ DI ROMA

Rehabilitation of the Parma Airport.

prepared by Arif Huseynov (Matricola 1873471)

under the guidance of Professor Maria Vittoria Corazza

Italy, 2021

Table of Contents.

Figures.

Tables.

Introduction.

The famous engineer K. W. Otto Lilienthal, who was one of the pioneers of worldwide aviation, said: "*To invent an airplane is nothing. To build one is something. But to fly is everything.". Nowadays I would like to add to this quote one more sentence from myself: "To land on time is something more than everything".* Indeed, when we talk about airports, the first thing that is coming to my mind – how they are managing a lot of operations, maintaining sustainable supply, and arranging everything to go like clockwork?... The accuracy of the Airports is the subject of the safety and sustainability, so it must be provided by default. However, the provision of such high service costs money. And at this point we are "landing from the sky on the ground" – airports first and foremost are one of the most important transport hubs, that connects not only cities, but also countries and continents. Airports need air-passengers, who will pay ticket price, cup of coffee before departure or buy nice gift in duty free area of the airport.

The aim of this project is to analyze and if possible, rehabilitate the demand for the Parma Airport. Its annual passenger traffic was continuously falling since 2011 (see ***Figure 1***), when the demand was around 270000 pax/year, till 2020 worldwide air transport crisis because of pandemic alert, when the demand of PA reached 20000 pax/year.[1] But even in 2019, just one year before COVID-19 damaged the lives of the millions and economy of the whole world, the demand of PA was 75000 pax/year.

I will try to understand possible causes of the demand's nose-dive by observing different scenarios and hidden trends in population growth of the Province of Parma and of the alternative choices in face of nearby airports. Then I will try to implement new ways and/or scenarios to increase the future demand.

Most of the computations in this project I had done on JupyterLab, it is an environment for Python language, which allows some useful data analytic tools and web scrapping features.

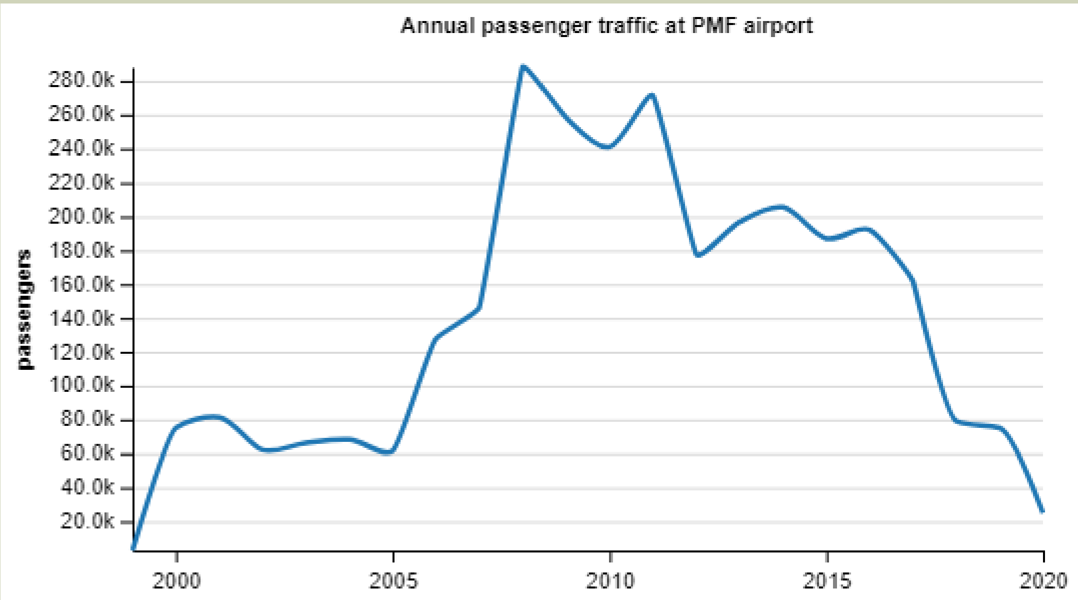


Figure 1. Annual passenger traffic at PMF airport.

[1] Source - https://en.wikipedia.org/wiki/Parma_Airport

Step 1.
OD Matrix and Time Thresholds.

The first goal is to analyze alternative Air Transport services for the inhabitants of the Province of Parma. These alternatives are six airports, situated in the Parma and nearby regions:

- Parma Airport (PMF)[2]
- Milan Malpensa Airport (MXP)
- Verona Airport (VRN)
- Venice Marco Polo Airport (VCE)
- Bologna Guglielmo Marconi Airport (BLQ)
- Pisa Airport (PSA)

The alternatives with different municipalities of the Province of Parma can be represented through Origin Destination Matrix, where airports will be destinations, and municipalities - the origins. Every cell of such matrix may contain different information of OD couple – like time, or distance between them, or even the identifications of the chosen path (Google, for example, is using Shortest Path Algorithm and dynamically calculating the best "cheapest" path between two nodes). To get this valuable information for my project I can choose several solutions. One of the ways is to use Google Maps, and enter step by step each OD couple – but this means, that if I have 44 origins and 6 destinations, I need to repeat this operation 44*6 = 264 times… As engineer I was curious to find easier way, and such way exists – Distance Matrix API which is based on Google Cloud Platform. This API can be easily used with python language, it is very well documented. On the **Code Cell # 1** [3] the first two steps are shown in the following logical order:

- first, we need to prepare our preliminary request and access the API with unique key, which is provided for every user and should not be shared with anyone.
- then we can create these requests in the logical loop, that allow us immediately to collect all coordinates of our origins and destination.

After we prepared all information which is needed to send final requests to Google API, it is time to recap our task: we must get information about travel times between each OD couple, determine thresholds and built Isochrone (in my case, Isoline) map. And in Python we can do it all together just one loop. But to make it easily understandable by my reader, I divided this process on the two loops. My first loop will create TT and Distance matrixes. This is how it looks in logical steps (**Code Cell # 2**):

- Take the list of all origins.
- For every origin in this list find shortest path to all destinations and read TT and distance information
- Create two vectors: TT and distance from current origin to all destinations.
- Append these vectors to the appropriate matrices.

Resulting matrices are shown in ***Table 1*** and ***Table 2***.

---

[2] This and following airport codes are IATA location identifiers.

[3] All code cells are available in the end of this document in the Codes section. Such structure of document built for non-distraction of the reader attention to the code routine, which is not primary objective of this project, rather just a tool.

Municipalities	PARMA AIRPORT	MILAN MALPENSA AIRPORT	Verona Airport, Caselle, VR	Venice Airport (VC	Aeroporto di Bologna	Aeroporto Pisa
Albareto	74,3	232,4	216,0	323,7	165,5	158,2
Bardi, Province of Parma	64,6	189,7	206,2	314,0	155,8	187,3
Bedonia	92,3	250,4	234,0	341,7	183,5	176,2
Berceto	58,0	216,1	199,7	307,4	149,2	134,6
Bore, Province of Parma	60,4	173,0	161,9	286,2	151,7	183,1
Borgo Val di Taro	67,3	225,4	209,0	316,8	158,5	151,2
Busseto	34,1	152,6	124,3	248,5	124,7	200,8
Calestano	34,6	204,2	187,7	295,5	137,3	150,5
Collecchio	14,2	180,5	157,2	265,0	106,8	165,2
Colorno	18,3	185,2	79,6	205,5	101,3	195,7
Compiano	81,4	239,5	223,0	330,8	172,6	165,2
Corniglio	54,8	230,8	161,5	294,4	136,2	147,1
Felino	24,8	193,7	134,5	267,3	109,1	174,9
Fidenza	22,5	156,9	146,0	270,2	112,9	181,8
Fontanellato	17,6	160,9	149,9	274,2	112,9	178,7
Fontevivo	13,1	166,2	158,2	266,0	107,7	173,5
Fornovo di Taro	32,6	190,7	174,3	282,0	123,8	155,3
Langhirano	28,1	204,1	134,8	267,7	109,5	187,7
Lesignano de' Bagni	31,6	206,2	156,8	264,6	106,4	191,2
Medesano	19,8	172,8	165,5	273,3	115,1	159,1
Monchio delle Corti	73,6	249,6	180,4	313,2	155,0	123,7
Montechiarugolo	23,5	193,8	140,7	248,5	90,3	196,8
Neviano degli Arduini	42,4	212,6	158,7	266,5	108,3	159,2
Noceto, Province of Parma	17,3	166,7	161,3	269,1	110,9	165,9
Palanzano	59,4	235,4	166,1	299,0	140,8	133,1
Parma, Province of Parma	5,5	179,9	110,2	227,5	95,9	186,0
Pellegrino Parmense	41,6	176,3	165,2	289,5	140,8	172,3
Polesine Zibello	32,2	156,0	127,9	252,2	131,6	198,3
Roccabianca	26,0	172,0	120,6	244,8	116,9	192,2
Sala Baganza	16,9	186,1	160,0	267,7	109,5	167,2
Salsomaggiore Terme	30,7	161,8	150,8	275,0	124,4	190,0
San Secondo Parmense	16,3	169,8	94,4	220,3	107,2	183,0
Sissa Trecasali	19,4	176,1	88,3	214,1	109,3	190,5
Solignano	42,9	201,1	184,6	292,4	134,2	150,3
Soragna	25,6	159,6	148,6	272,9	115,6	185,3
Sorbolo Mezzani	16,8	184,8	102,9	220,2	91,5	195,3
Terenzo	42,0	200,2	183,7	291,5	133,3	149,7
Tizzano Val Parma	44,0	219,9	150,7	283,6	125,3	162,9
Tornolo	80,8	239,0	222,5	330,3	172,1	140,7
Torrile	12,3	180,6	85,4	211,2	96,7	191,1
Traversetolo	29,4	199,6	145,7	253,5	95,3	189,2
Valmozzola	62,4	220,5	204,0	311,8	153,6	146,2
Varano de' Melegari	36,6	194,7	178,3	286,0	127,8	159,3
Varsi	52,5	210,7	194,2	302,0	143,8	175,3

Table 1. Distances Matrix.

Municipalities	PARMA AIRPORT	MILAN MALPENSA AIRPORT	Verona Airport	Venice Airport (VC	Aeroporto di Bologna	Aeroporto Pisa
Albareto	63,8	154,3	144,1	206,6	112,2	113,2
Bardi, Province of Parma	62,6	151,4	143,0	205,5	111,0	138,0
Bedonia	88,5	179,0	168,9	231,4	136,9	138,0
Berceto	55,4	145,9	135,8	198,3	103,8	96,9
Bore, Province of Parma	60,2	124,5	119,5	189,6	108,6	135,6
Borgo Val di Taro	56,5	147,0	136,8	199,4	104,9	106,0
Busseto	38,8	103,7	88,3	158,4	84,8	139,1
Calestano	37,0	136,7	126,6	189,1	94,6	118,0
Collecchio	14,2	116,9	103,1	165,6	71,1	114,7
Colorno	20,5	119,9	81,7	150,0	68,0	133,4
Compiano	74,4	164,9	154,8	217,3	122,8	123,9
Corniglio	59,9	168,5	147,2	207,9	113,4	117,8
Felino	30,3	137,7	119,9	180,5	86,1	133,1
Fidenza	26,3	101,6	96,9	167,1	72,5	124,9
Fontanellato	19,1	104,6	99,9	170,1	73,5	120,0
Fontevivo	13,9	110,0	100,8	163,3	68,8	115,3
Fornovo di Taro	29,2	119,7	109,6	172,1	77,6	104,6
Langhirano	28,9	137,5	116,2	176,9	82,4	138,3
Lesignano de' Bagni	34,7	141,5	114,3	176,8	82,3	144,2
Medesano	22,2	115,3	107,7	170,3	75,8	108,1
Monchio delle Corti	85,9	194,5	173,2	233,9	139,4	109,7
Montechiarugolo	24,2	128,7	95,1	157,6	63,2	138,8
Neviano degli Arduini	44,8	149,4	116,2	178,7	84,2	157,4
Noceto, Province of Parma	20,5	111,2	104,6	167,1	72,6	116,0
Palanzano	64,8	173,4	152,2	212,9	118,4	120,1
Parma, Province of Parma	12,0	117,5	95,0	162,5	65,6	129,9
Pellegrino Parmense	51,9	124,8	119,8	190,0	99,5	126,5
Polesine Zibello	32,8	109,1	90,6	160,7	89,0	138,9
Roccabianca	27,9	115,3	91,5	161,7	84,3	133,9
Sala Baganza	18,4	124,4	107,2	169,8	75,3	119,8
Salsomaggiore Terme	34,6	111,0	106,0	176,1	83,5	133,2
San Secondo Parmense	18,4	113,0	96,3	164,6	74,9	125,0
Sissa Trecasali	22,2	122,3	92,4	160,7	76,1	134,7
Solignano	42,1	132,6	122,4	185,0	90,5	114,5
Soragna	26,4	103,8	99,1	169,3	74,7	125,9
Sorbolo Mezzani	20,7	121,8	83,3	150,8	65,8	135,3
Terenzo	41,5	132,0	121,8	184,4	89,9	115,0
Tizzano Val Parma	47,3	155,9	134,6	195,3	100,8	142,1
Tornolo	71,8	162,3	152,1	214,7	120,2	121,4
Torrile	13,8	114,9	87,9	156,2	63,1	128,4
Traversetolo	29,9	134,4	101,2	163,7	69,3	142,0
Valmozzola	62,6	153,1	142,9	205,5	111,0	112,1
Varano de' Melegari	32,4	122,9	112,8	175,3	80,8	107,8
Varsi	49,3	139,8	129,6	192,2	97,7	124,6

Table 2. Travel Time matrix (in minutes)

It is time to define travel time thresholds for our users . It seems to be unrealistic for the potential travellers choosing the airport which is really far compare to the nearby Parma Airport. But the travel time is not only criteria of choosing some airport – the uniqueness of provided services can be crucial and push the customer on the long way to the airport, who is providing this service (for example, some flight, which is not provided by Parma Airport). Therefore I will not exclude distant airports, rather than add some negative weight (we will see further) , that will decrease attractiveness of far destinations.

In ***Table 3*** the threshold values are shown:

- TT lower or equal to 30 minutes is represented as 0 and green color cell.
- TT lower or equal to 60 minutes is represented as 1 and blue color cell.
- TT lower or equal to 90 minutes is represented as 2 and yellow color cell.
- TT lower or equal to 120 minutes is represented as 3 and brown color cell.
- TT higher than 120 minutes is represented as 4 and red color cell.

Municipalities	Column2	Column3	Column4	Column5	Column6	Column7
Nodes	PARMA AIRPORT	MILAN AIRPORT	Verona Airport	Venice Airport	Aeroporto di Bologna	Aeroporto Pisa
Albareto	2	4	4	4	3	3
Bardi, Province of Parma	2	4	4	4	3	4
Bedonia	2	4	4	4	4	4
Berceto	1	4	4	4	3	3
Bore, Province of Parma	2	4	3	4	3	4
Borgo Val di Taro	1	4	4	4	3	3
Busseto	1	3	2	4	2	4
Calestano	1	4	4	4	3	3
Collecchio	0	3	3	4	2	3
Colorno	0	3	2	4	2	4
Compiano	2	4	4	4	4	4
Corniglio	1	4	4	4	3	3
Felino	1	4	3	4	2	4
Fidenza	0	3	3	4	2	4
Fontanellato	0	3	3	4	2	3
Fontevivo	0	3	3	4	2	3
Fornovo di Taro	0	3	3	4	2	3
Langhirano	0	4	3	4	2	4
Lesignano de' Bagni	1	4	3	4	2	4
Medesano	0	3	3	4	2	3
Monchio delle Corti	2	4	4	4	4	3
Montechiarugolo	0	4	3	4	2	4
Neviano degli Arduini	1	4	3	4	2	4
Noceto, Province of Parma	0	3	3	4	2	3
Palanzano	2	4	4	4	3	4
Parma, Province of Parma	0	3	3	4	2	4
Pellegrino Parmense	1	4	3	4	3	4
Polesine Zibello	1	3	3	4	2	4
Roccabianca	0	3	3	4	2	4
Sala Baganza	0	4	3	4	2	3
Salsomaggiore Terme	1	3	3	4	2	4
San Secondo Parmense	0	3	3	4	2	4
Sissa Trecasali	0	4	3	4	2	4
Solignano	1	4	4	4	3	3
Soragna	0	3	3	4	2	4
Sorbolo Mezzani	0	4	2	4	2	4
Terenzo	1	4	4	4	2	3
Tizzano Val Parma	1	4	4	4	3	4
Tornolo	2	4	4	4	4	4
Torrile	0	3	2	4	2	4
Traversetolo	0	4	3	4	2	4
Valmozzola	2	4	4	4	3	3
Varano de' Melegari	1	4	3	4	2	3
Varsi	1	4	4	4	3	4

Table 3.Time Thresholds for OD matrix

The last thing we need to do are Isochrone maps. Python has immensely powerful tools for plotting, so we will run **Code Cell # 3** and **Code Cell # 4** to plot all Isolines. We need only appropriate map plots for background, and they are available in the OpenStreetMap service. The results are shown in **Figure 2**, **Figure 3**, **Figure 4**, **Figure 5**, **Figure 6**, **Figure 7** with the same color logics (except brown, which is black colored on map) as table above.

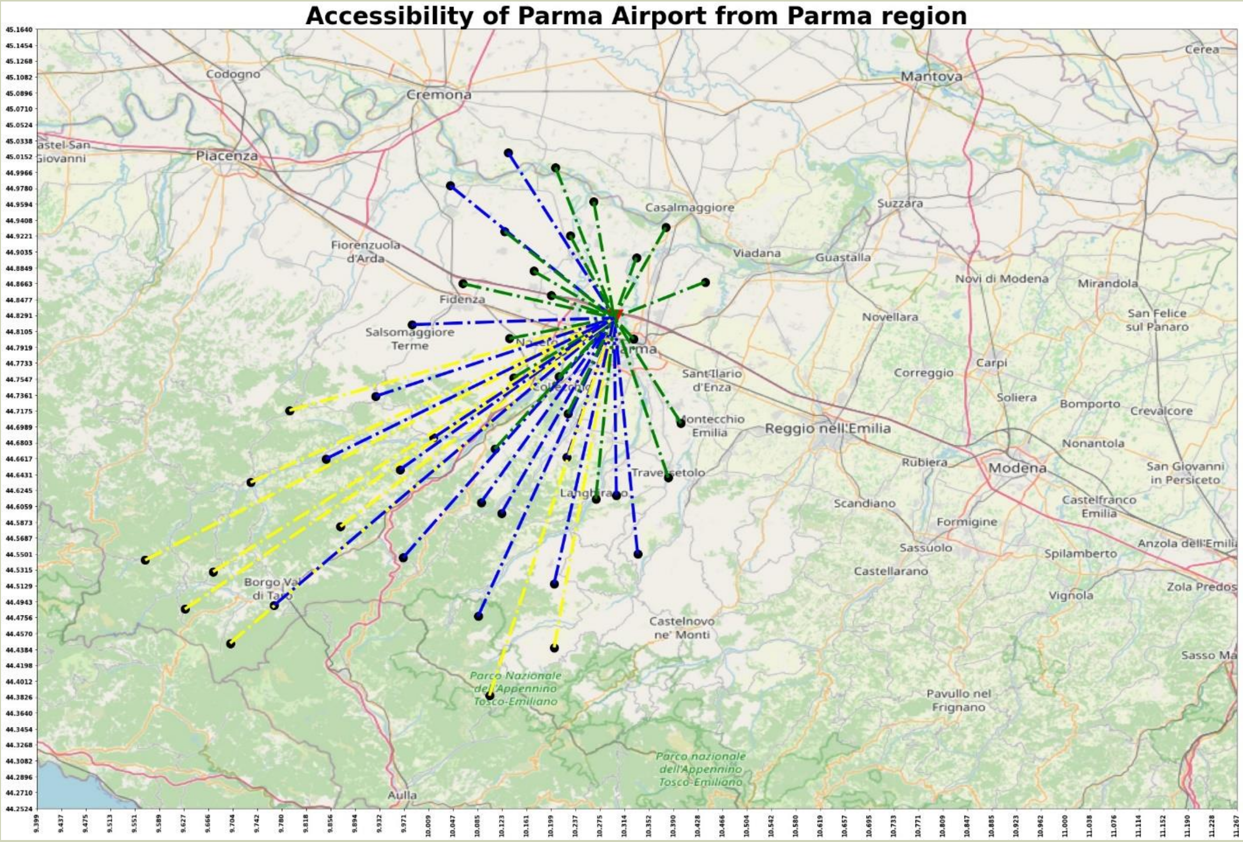


Figure 2. Accessibility of Parma airport.

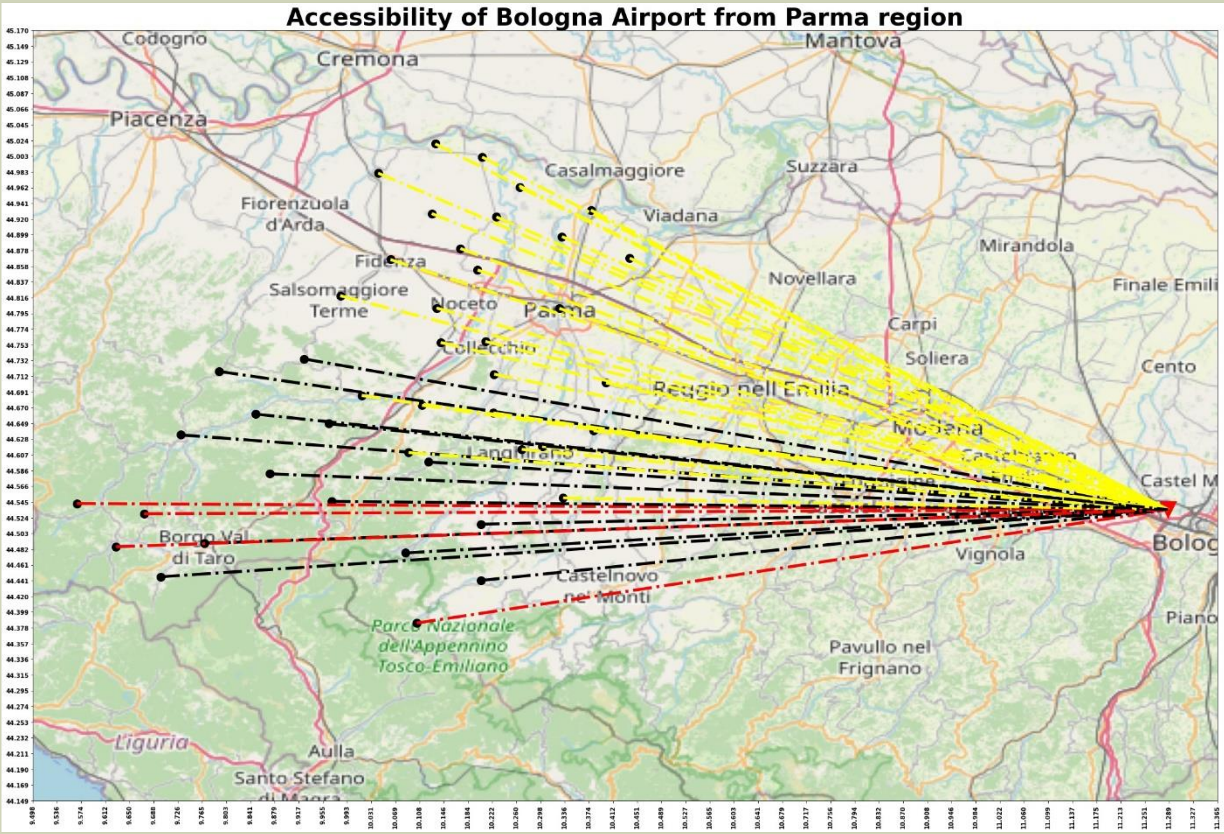


Figure 3. Accessibility of Bologna airport.

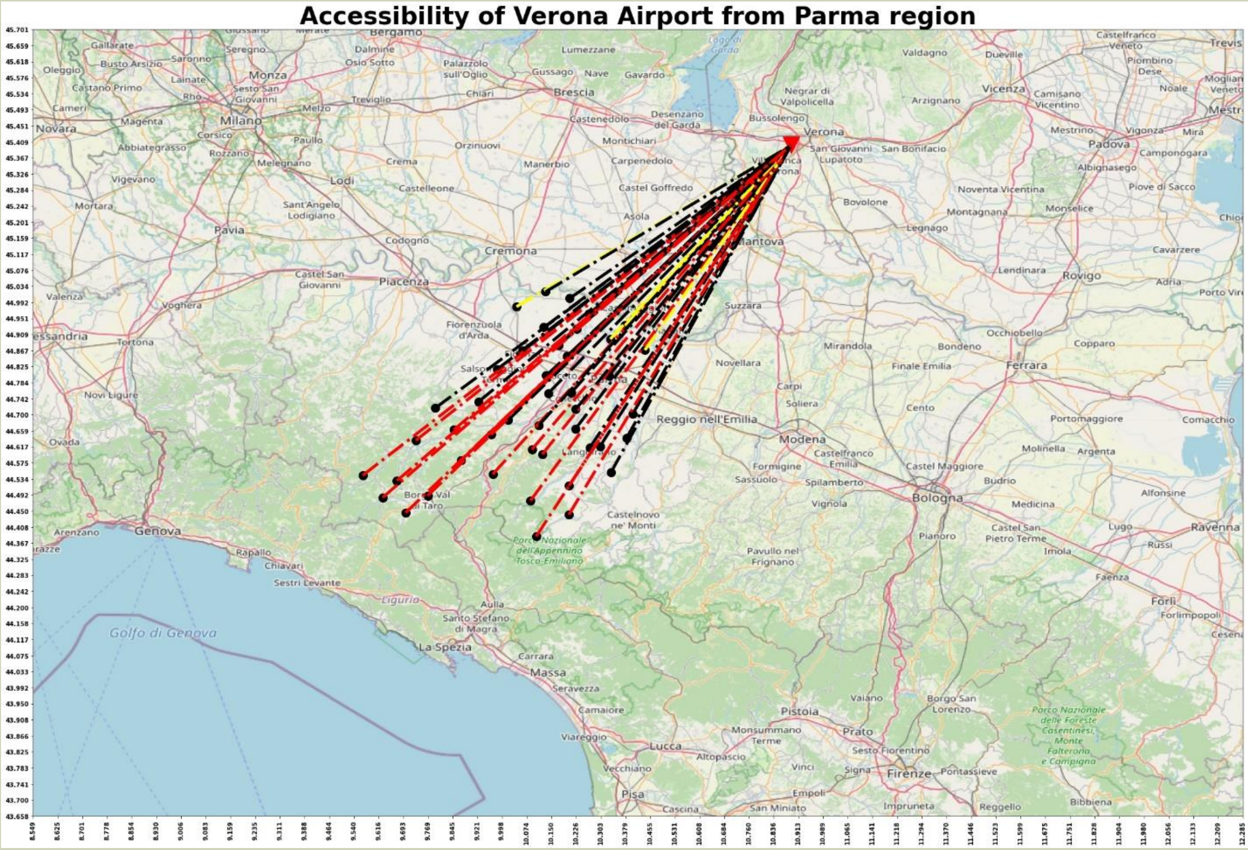


Figure 4. Accessibility of Verona airport.

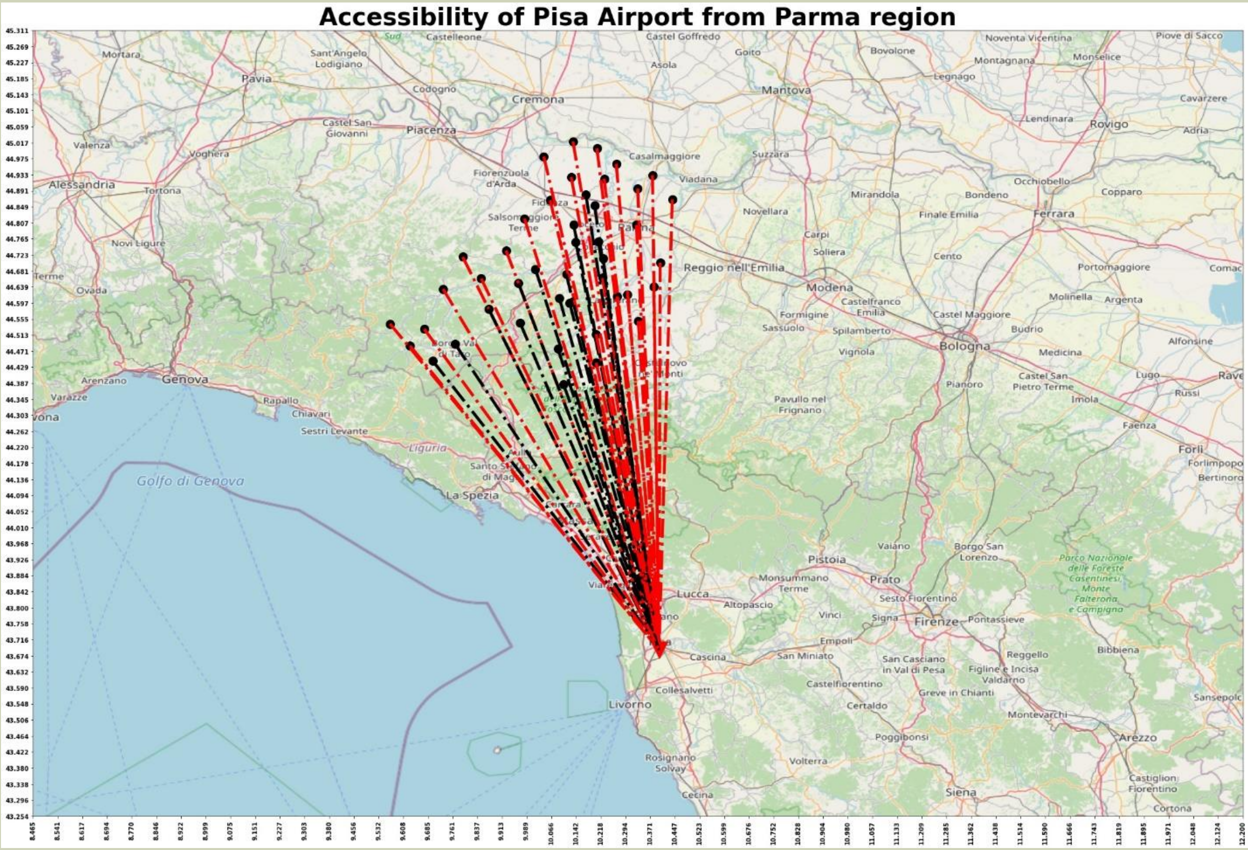


Figure 5. Accessibility of Pisa airport.

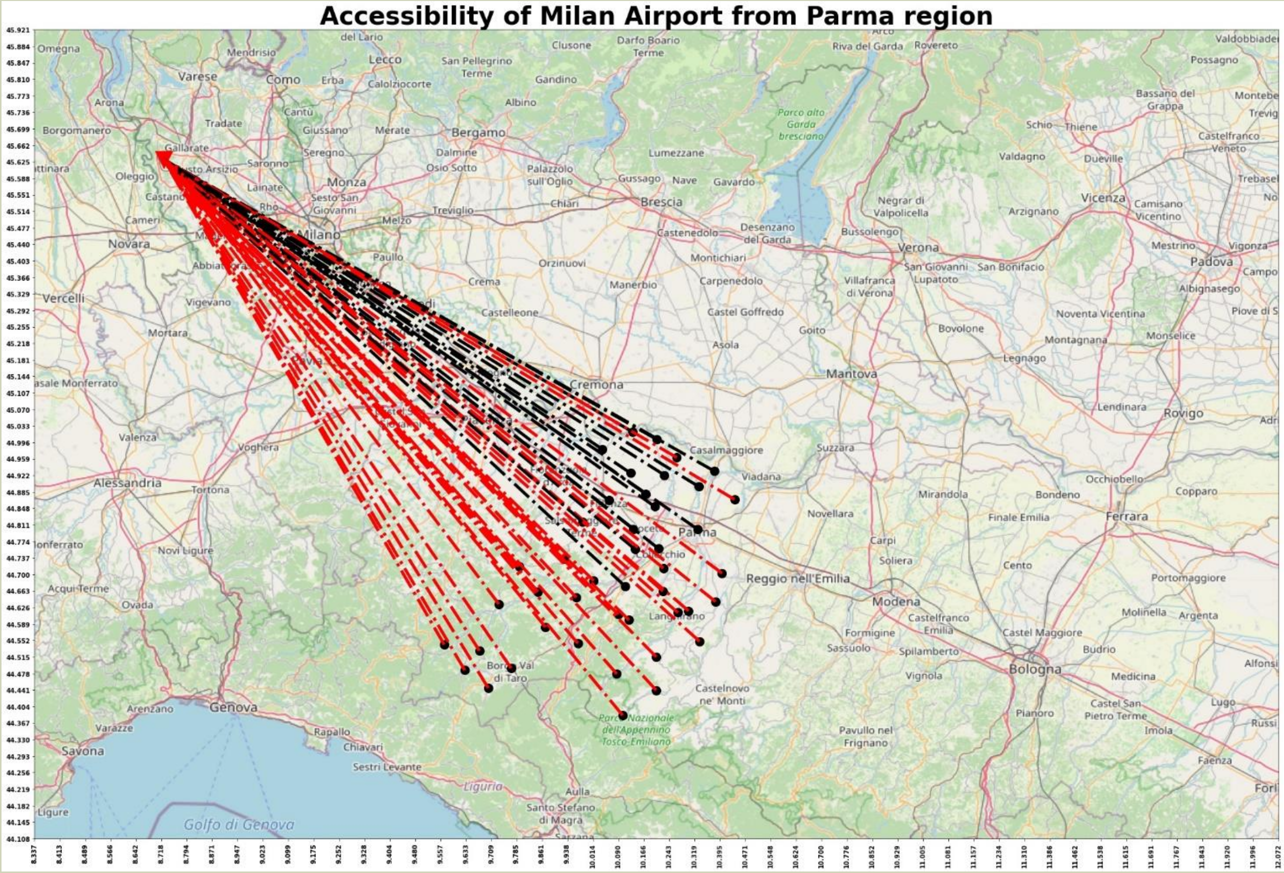


Figure 6. Accessibility of Milan airport.

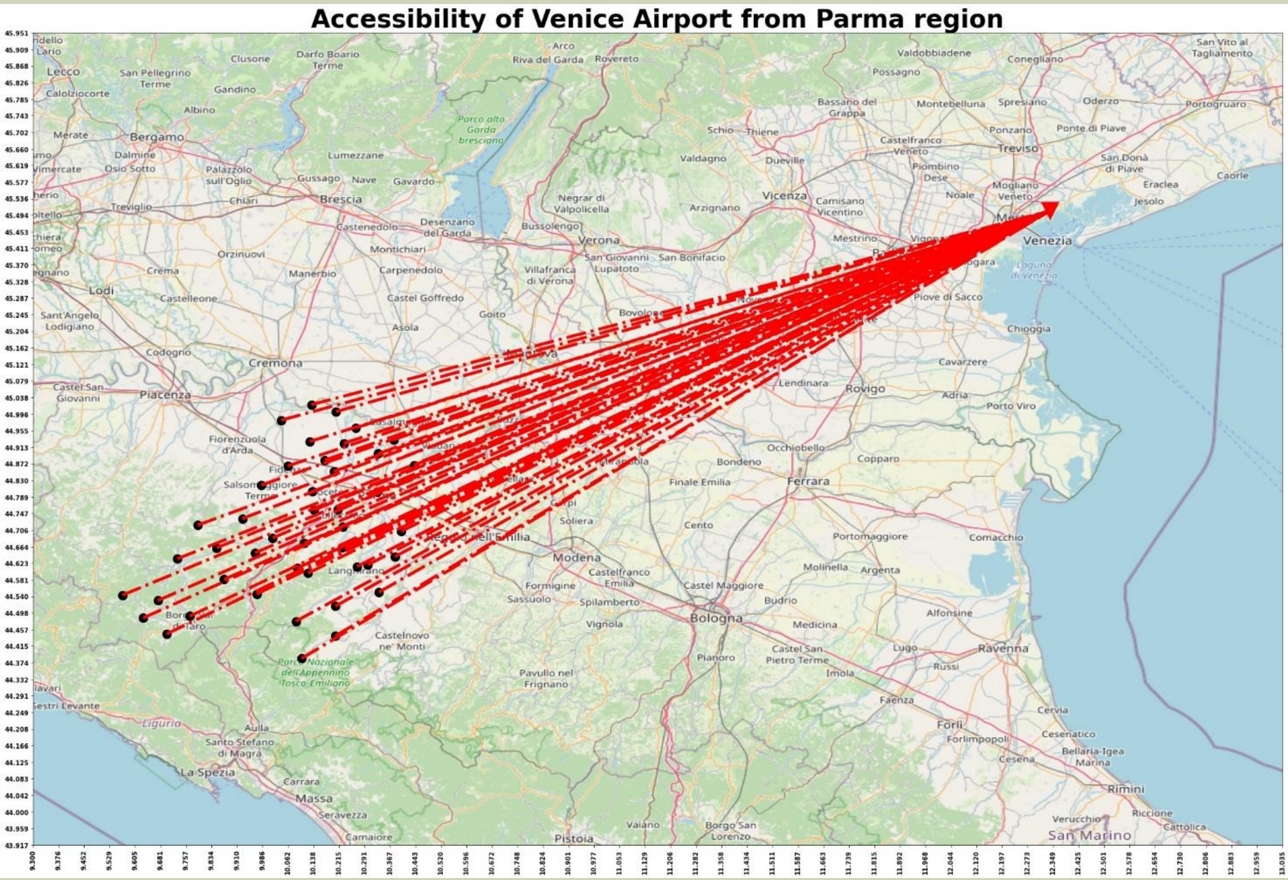


Figure 7. Accessibility of Venice airport.

Step 2
Demand analysis.

Provincial demand

Population statistics are available on istat.it website. This website has databases, which are accessible for querying through standardized SDMX API. This API stands for Statistical Data and Metadata eXchange and allows scientists, engineers, and different professionals to access different statistical sources in Europe under one standard. However, it was incredibly challenging for me to understand documentation of this API, though it is not well documented and requires some more time to understand it. Since we have a limited time, and our objective is rehabilitation of Parma Airport, rather than development of data acquisition for this project, I decided to choose different but still more efficient way to get population data and analyze demand. Instead of fulfilling excel sheet step by step with information obtained from istat.it I downloaded all population datafiles for every municipality separately and built the code, which made it much faster and allowed me to save some additional data for possible insights during this project. Each such datafile contains by default population in every age separately for current municipality and year. After running **Code Cell # 5**, **Code Cell # 6**, **Code Cell # 7** we will get final data + some population indicators, which calculation was also done in these cells. Final matrix is shown in the **Table 4**. Inputs of this table are following:

- Total population of each municipality in the Province of Parma in **2011**, and especially:
  - Population between 26 and 65 including, 2011.
  - Population over 65, 2011.
- Total population of each municipality in the Province of Parma in **2020,** and especially:
  - Population between 26 and 65 including, 2020.
  - Population over 65, 2011.

Indicators, that I used in my analysis are following:

- Population ***difference*** indicators between 2011 and 2020 (= 2020 – 2011):
      - Between population values in 26-65 ages.
      - Between population values in over 65 ages.
      - Between total population values
- Population ***proportion*** indicators (= popul/total_popul):
  - Ratio of population in 26-65 ages to the total population in:
    - 2011
    - 2020
  - Ratio of population in over 65 ages to the total population in:
    - 2011
    - 2020
  - Proportion variation (difference between 2011 and 2020 proportions):
    - In 26-65 ages
    - In over 65 ages
- Population ***variation*** indicators:
  - Ratio of appropriate difference indicator to the population of 2011:
    - In 26-65 ages
    - In over 65 ages
    - Total population

Column1	popul_26_65_2011	popul_over_65_2011	total_2011	popul_26_65_2020	popul_over_65_2020	total_2020	difference_26_65_2011_2020	variation_26_65_2011_2020	proportion_26_65_2011	proportion_26_65_2020	difference_over_65_2011_2020	variation_over_65_2011_2020	proportion_over_65_201	proportion_over_65_2020	propor_variation_26_65_2011_202	propor_variation_over_65_2011_2020	total_difference_2011_2020	total_variation_2011_2020
Albareto	1183	648	2232	1099	641	2116	-84	-7,100591716	53,00179211	51,93761815	-7	-1,080246914	29,03225806	30,29300567	-1,064173967	1,260747607	-116	-5,197132616
Bardi	1163	837	2382	1020	795	2122	-143	-12,29578676	48,82451721	48,06786051	-42	-5,017921147	35,13853904	37,46465598	-0,756656703	2,326116942	-260	-10,91519731
Bedonia	1942	1076	3701	1593	1130	3295	-349	-17,97116375	52,47230478	48,34597876	54	5,018587361	29,07322345	34,29438543	-4,126326027	5,221161979	-406	-10,97000811
Berceto	1195	682	2189	1026	706	2008	-169	-14,14225941	54,59113751	51,09561753	24	3,519061584	31,15577889	35,15936255	-3,495519976	4,003583655	-181	-8,268615806
Bore	367	347	800	274	337	687	-93	-25,34059946	45,875	39,88355167	-10	-2,88184438	43,375	49,05385735	-5,991448326	5,678857351	-113	-14,125
Borgo Val di Taro	3830	1941	7319	3483	1895	6816	-347	-9,060052219	52,32955322	51,10035211	-46	-2,369912416	26,5200164	27,80223005	-1,229201105	1,282213651	-503	-6,872523569
Busseto	3811	1649	7052	3620	1679	6851	-191	-5,011807924	54,04140669	52,83900161	30	1,819284415	23,38343732	24,50737119	-1,202405088	1,123933864	-201	-2,850255247
Calestano	1173	490	2126	1121	516	2108	-52	-4,433077579	55,17403575	53,17836812	26	5,306122449	23,04797742	24,47817837	-1,995667626	1,430200946	-18	-0,846660395
Collecchio	8071	2747	14120	8144	2987	14683	73	0,904472804	57,16005666	55,46550432	240	8,736803786	19,45467422	20,3432541	-1,694552332	0,888579882	563	3,987252125
Colorno	5112	1615	9096	5045	1717	9103	-67	-1,310641628	56,2005277	55,42128968	102	6,315789474	17,75505717	18,86191365	-0,77923802	1,106856487	7	0,076956904
Compiano	603	303	1131	554	319	1096	-49	-8,126036484	53,31564987	50,54744526	16	5,280528053	26,79045093	29,10583942	-2,768204612	2,315388488	-35	-3,094606543
Corniglio	1034	722	2071	930	620	1803	-104	-10,05802708	49,92757122	51,58069884	-102	-14,12742382	34,86238532	34,38713256	1,653127614	-0,475252764	-268	-12,9406084
Felino	4899	1638	8546	5044	1889	9160	145	2,959787712	57,32506436	55,06550218	251	15,32356532	19,16686169	20,62227074	-2,259562174	1,455409053	614	7,184647788
Fidenza	14170	5894	26196	14382	6110	27012	212	1,49611856	54,09222782	53,24300311	216	3,664743807	22,49961826	22,61957648	-0,849224711	0,119958222	816	3,114979386
Fontanellato	4003	1453	7080	3889	1575	7096	-114	-2,847864102	56,53954802	54,80552424	122	8,396421198	20,52259887	22,19560316	-1,734023784	1,673004287	16	0,225988701
Fontevivo	3281	951	5572	3185	1112	5595	-96	-2,925937214	58,88370424	56,92582663	161	16,92954784	17,06748026	19,87488829	-1,957877605	2,807408035	23	0,412778177
Fornovo di Taro	3354	1432	6294	3120	1422	5965	-234	-6,976744186	53,28884652	52,30511316	-10	-0,698324022	22,75182714	23,83906119	-0,98373336	1,087234053	-329	-5,227200508
Langhirano	5555	1851	9842	5785	2072	10581	230	4,140414041	56,44178013	54,67347132	221	11,93949217	18,80715302	19,58227011	-1,768308809	0,775117089	739	7,508636456
Lesignano de' Bagni	2880	734	4795	2917	898	5047	37	1,284722222	60,06256517	57,79671092	164	22,34332425	15,3076121	17,79274817	-2,265854255	2,485136071	252	5,255474453
Medesano	5997	2048	10749	6027	2246	10954	30	0,500250125	55,79123639	55,0209969	198	9,66796875	19,05293516	20,50392551	-0,770239498	1,45099035	205	1,907154154
Monchio delle Corti	520	392	1024	392	377	864	-128	-24,61538462	50,78125	45,37037037	-15	-3,826530612	38,28125	43,63425926	-5,41087963	5,353009259	-160	-15,625
Montechiarugolo	6151	2086	10626	6033	2460	11117	-118	-1,918387254	57,88631658	54,26823783	374	17,92905081	19,63109354	22,12827202	-3,618078748	2,497178472	491	4,620741577
Neviano degli Arduini	1907	1112	3750	1821	1021	3557	-86	-4,509701101	50,85333333	51,1948271	-91	-8,183453237	29,65333333	28,70396401	0,341493768	-0,949369319	-193	-5,146666667
Noceto	7337	2314	12724	7229	2611	12955	-108	-1,471991277	57,66268469	55,80084909	297	12,83491789	18,186105	20,15438055	-1,861835597	1,96827555	231	1,815466834
Palanzano	606	442	1221	535	406	1086	-71	-11,71617162	49,63144963	49,26335175	-36	-8,14479638	36,1998362	37,38489871	-0,368097882	1,185062511	-135	-11,05651106
Parma	106207	39370	186690	111540	43068	200455	5333	5,021326278	56,88949596	55,64341124	3698	9,392938786	21,08843537	21,48512135	-1,246084716	0,396685975	13765	7,373185495
Pellegrino Parmense	558	379	1097	469	343	988	-89	-15,94982079	50,86599818	47,46963563	-36	-9,498680739	34,54876937	34,71659919	-3,396362549	0,167829819	-109	-9,936189608
Roccabianca	1697	713	3110	1537	722	2919	-160	-9,428403064	54,5659164	52,65501884	9	1,26227209	22,92604502	24,73449812	-1,910897557	1,8084531	-191	-6,1414791
Sala Baganza	3151	991	5394	3118	1204	5709	-33	-1,047286576	58,41675936	54,61551936	213	21,49344097	18,37226548	21,08950779	-3,801240007	2,717242315	315	5,839822024
Salsomaggiore Terme	10863	4699	20051	10351	4719	19419	-512	-4,713246801	54,17684903	53,30346568	20	0,425622473	23,43524014	24,30094238	-0,873383357	0,865702238	-632	-3,151962496
San Secondo Parmense	3124	1143	5648	3136	1234	5758	12	0,384122919	55,31161473	54,46335533	91	7,961504812	20,23725212	21,43105245	-0,848259399	1,193800324	110	1,947592068
Solignano	1031	423	1858	930	447	1709	-101	-9,796314258	55,48977395	54,41778818	24	5,673758865	22,7664155	26,15564658	-1,07198577	3,389231076	-149	-8,019375673
Soragna	2737	938	4883	2639	1031	4814	-98	-3,58056266	56,05160762	54,81927711	93	9,914712154	19,20950236	21,41670129	-1,23233051	2,207198933	-69	-1,413065738
Terenzo	661	372	1239	621	363	1187	-40	-6,051437216	53,34947538	52,31676495	-9	-2,419354839	30,02421308	30,58129739	-1,03271043	0,557084313	-52	-4,19693301
Tizzano Val Parma	1127	630	2163	1087	586	2116	-40	-3,549245785	52,10355987	51,3705104	-44	-6,984126984	29,12621359	27,69376181	-0,733049474	-1,432451777	-47	-2,172907998
Tornolo	579	397	1145	459	341	919	-120	-20,7253886	50,56768559	49,94559304	-56	-14,10579345	34,67248908	37,10554951	-0,622092554	2,433060427	-226	-19,73799127
Torrile	4772	1009	7804	4475	1224	7695	-297	-6,223805532	61,14812916	58,15464587	215	21,30822597	12,92926704	15,90643275	-2,993483291	2,977165706	-109	-1,396719631
Traversetolo	5229	1761	9339	5246	2001	9604	17	0,325109964	55,99100546	54,62307372	240	13,6286201	18,85640861	20,83506872	-1,367931742	1,978660112	265	2,837562908
Valmozzola	293	218	585	250	219	528	-43	-14,67576792	50,08547009	47,34848485	1	0,458715596	37,26495726	41,47727273	-2,736985237	4,212315462	-57	-9,743589744
Varano de' Melegari	1508	542	2704	1419	578	2626	-89	-5,901856764	55,76923077	54,0365575	36	6,642066421	20,0443787	22,0106626	-1,732673267	1,966283906	-78	-2,884615385
Varsi	639	479	1300	567	437	1178	-72	-11,26760563	49,15384615	48,13242784	-42	-8,768267223	36,84615385	37,09677419	-1,02141831	0,250620347	-122	-9,384615385
Sissa Trecasali	4459	1581	7990	4225	1688	7788	-234	-5,247813411	55,80725907	54,2501284	107	6,767868438	19,78723404	21,67437083	-1,557130671	1,887136784	-202	-2,5281602
Polesine Zibello	1836	811	3385	1746	775	3182	-90	-4,901960784	54,23929099	54,87115022	-36	-4,438964242	23,95864106	24,3557511	0,63185923	0,397110036	-203	-5,99704579
Sorbolo Mezzani	7524	2332	13097	6980	2492	12602	-544	-7,230196704	57,4482706	55,38803365	160	6,861063465	17,80560434	19,77463895	-2,060236951	1,969034609	-495	-3,779491487

Table 4. Population data and some useful indicators.

I found ***proportion*** indicators especially useful in my analysis, because these indicators allow me to easily access some meaningful trends in population growth (or decrease), such as population aging problem, or hidden urbanization trends, which are happening due to attractiveness of big cities to the young generations. Finally, I propose following threshold limit combinations for further demand generation analysis:

- First negative combination:
    - Population variation of the current municipality in age 26-65 is lower than appropriate ***threshold lower limit***.
    - Proportion variation of the current municipality in age 26-65 between appropriate ***lower*** and ***upper threshold limits.***

    If population in some age range is decreasing during the years, but its ratio to the total population is not changing significantly, this means, that total population is decreasing too. On the other hand, if total population decreasing, this is not necessarily meaning that population in specific age range is decreasing. That is why to put attention on both total populations decrease and precisely decrease in specific age range, I used indicators mentioned above. But why 26-65? Because I assume, that people in this range traveling more often than people over 65. Thus, decrease in this age range with total decrease of population may significantly reduce potential demand.

- Second negative combination (two possible combinations):
    - Population variation of the current municipality in age 26-65 is lower than appropriate ***threshold lower limit***.
    - Proportion variation of the current municipality in age 26-65 is lower than appropriate ***threshold lower limit***.

    or

    - Population variation of the current municipality in age 26-65 is lower than appropriate ***threshold lower limit***.
    - Proportion variation of the current municipality in age over 65 is higher than appropriate ***threshold upper limit***.

    Similar to the previous combination we observe rapid decrease in population in 26-65 range. However, the total population is not rapidly decreasing, which means, that population in age range over 65 is increasing. Thanks to the proportion indicators we can easily catch such situation. Demand in this case will decrease not such significantly as in previous combination, but municipalities with aging trend should be under higher attention of the government authorities and political actors.

- First positive combination:

    - Population variation of the current municipality in age 26-65 is higher than appropriate ***threshold upper limit***.
    - Proportion variation of the current municipality in age 26-65 between appropriate ***lower*** and ***upper threshold limits.***

    Increase in population is generally causing increase in potential demand. Also, we control that this is not simply increase in total population but increase in "likely to travel" age.

- Second positive combination:

    - Population variation of the current municipality in age 26-65 is higher than appropriate ***threshold upper limit***.
    - Proportion variation of the current municipality in age 26-65 is higher than appropriate ***threshold upper limit***.

    or

    - Population variation of the current municipality in age 26-65 is higher than appropriate ***threshold upper limit***.
    - Proportion variation of the current municipality in age over 65 is lower than appropriate ***threshold lower limit***.

    Municipality's population is increasing towards population in the 26-65 range, while old people either increasing with similar trend, or increasing slightly slower than young generation. I expect higher potential demand from such combination.

Note. I call it "threshold combinations" instead of "scenarios", because later I will create scenarios for the post pandemic periods, and I do not want to mix these terms.

Well, we just need to determine our threshold limit values for every indicator. This is done in **Code Cell # 8**, and results are following:

- Population variation threshold limits in 26-65 age range:
    - Lower limit assumed as 25th percentile of all variations and equal to **-9.86%.**

- 75^{th} percentile of all variations equal to -1.43%, since it still shows the decrease in population, the upper limit assumed to be **0%**
- Proportion variation threshold limits in 26-65 age range:
  - Lower limit assumed as 20^{th} percentile of all proportion variations and equal to **-2.74%**
  - 80^{th} percentile of all proportion variations equal to -0.82%, since it still shows the decrease in proportion variation (which means that decrease in population in the current age is very small, but still exists), the upper limit assumed to be **0%**
- Proportion variation threshold limit in over 65 age range:
  - Upper limit assumed as 80^{th} percentile of proportion variations and equal to **2.59%**
  - Lower limit assumed as 20^{th} percentile of proportion variations and equal to **0.69%**

If some municipality will meet conditions of the above metioned threshold combinations, then appropriate adjustment coefficients $\mathbf{T_p}$ will be applied to the potential demand of this municipality:

- First negative combination $\mathbf{T_p}$ = 0.8:

  demand = potential_demand * **0.8**

- Second negative combination $\mathbf{T_p}$ = 0.9:

  demand = potential_demand * **0.9**

- First positive combination $\mathbf{T_p}$ = 1.1:

  demand = potential_demand * **1.1**

- Second positive combination $\mathbf{T_p}$ = 1.2:

  demand = potential_demand * **1.2**

Next step in the trip generation process is determination of the average trip rate made by inhabitant of the Province of Parma abroad and by aircraft. To get these values I used data from http://dati.istat.it/Index.aspx?QueryId=19101&lang=en# . According to this data:

- **1.57** trips per inhabitant made for different purposes in 2019
- **24%** from all trips made abroad
- **68%** from all overseas trips (trips made abroad) made by aircraft:
  - **15%** for business purposes
  - **85%** for holidays and other purposes

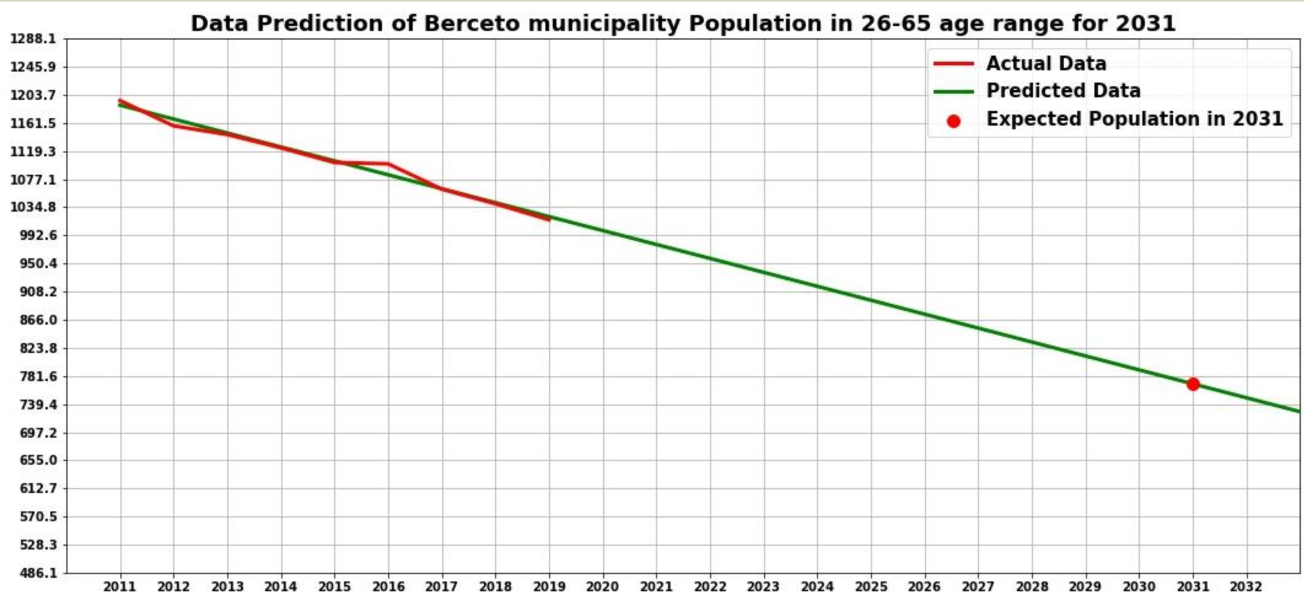
Also, according to UNO news (https://news.un.org/en/story/2021/01/1082302), ***"there was a "dramatic" fall in international air travel due to COVID-19, of around 60 % over the course of last year":***

- For post situation all travels by aircraft will presume this trend first quarters (an assumption), so 60% decrease should be applied compared to 2019 values (multiply at **0.4**).

Before writing about demand scenarios, I need to find expected population in all municipalities and different age groups separately. Thanks to the www.istat.it it was possible to download all population data from 2011 to 2019 in all municipalities and ages separately. This task maybe seems so trivial, but if we see deeply – we need to download 44 (sometime 45 or 46) municipalities * 8 years = 352 files, which is not pleasant task for doing it manually. Luckily, I found a way to download all these files with few lines of code, represented in ***Code Cell # 9***. After acquisition of data, I prepared from all files one datafile and got predicted values through linear regression model in **Code Cell # 10**, ***Code Cell # 11***, ***Code Cell # 12***.

As example, if we take municipality ***Berceto*** and explore its population growth trend, we will see that for population in age 26-65 was only decreasing during these years (see

Figure 8). However, the population in over 65 age range is slowly growing, so it is possible that we observe population aging trend. This situation is not similar for municipalities with higher social activities,



such as Parma, where people in both age ranges are increasing.

Figure 8. Berceto population in 26-65 age range growth trend and expected value for 2031.

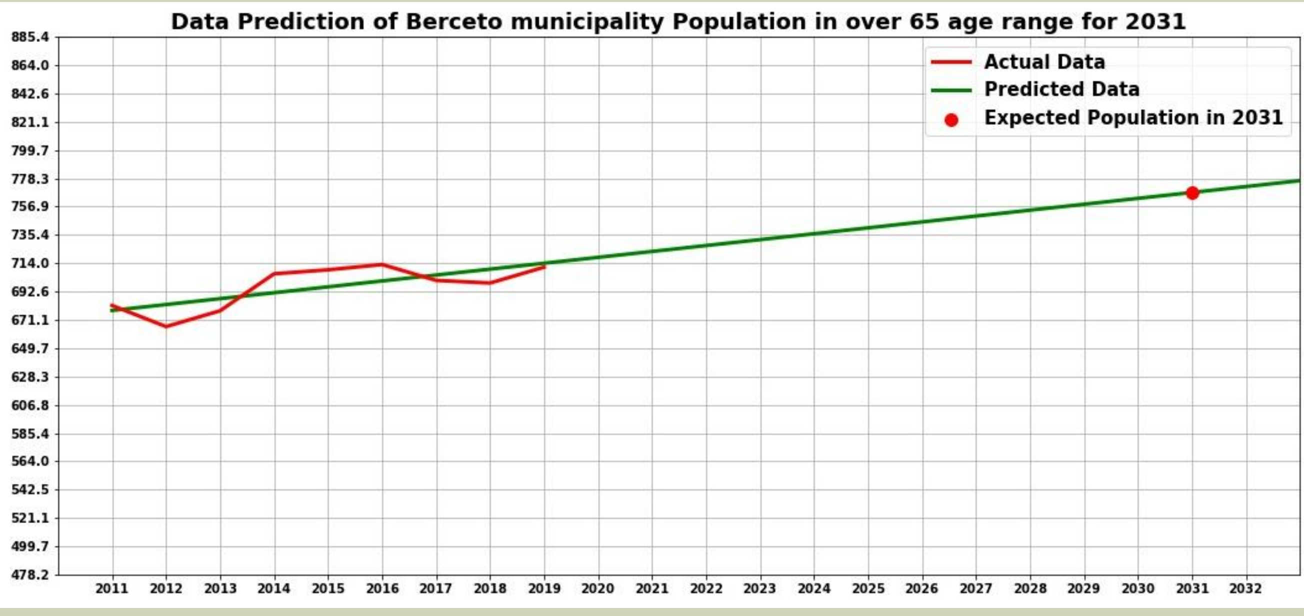


***Figure 9.* Berceto population in over 65 age range growth trend and expected value for 2031.**

Municipalities	age_26_65_2011	age_over_65_2011	age_26_65_2012	age_over_65_2012	age_26_65_2013	age_over_65_2013	age_26_65_2014	age_over_65_2014	age_26_65_2015	age_over_65_2015	age_26_65_2016	age_over_65_2016	age_26_65_2017	age_over_65_2017	age_26_65_2018	age_over_65_2018	age_26_65_2019	age_over_65_2019	age_26_65_2021	age_over_65_2021	age_26_65_2031	age_over_65_2031
Albareto	1183	648	1146	636	1145	641	1156	631	1150	640	1153	622	1139	642	1133	653	1108	650	1110	646	1051	654
Bardi	1163	837	1140	819	1124	832	1121	812	1097	822	1083	808	1046	814	1028	819	1023	805	983	802	801	774
Bedonia	1942	1076	1872	1070	1846	1085	1819	1113	1778	1111	1748	1111	1705	1104	1668	1088	1640	1108	1562	1118	1200	1154
Berceto	1195	682	1157	666	1144	678	1124	706	1102	709	1100	713	1062	701	1040	699	1016	711	979	723	770	768
Bore	367	347	352	341	353	343	337	357	314	362	304	355	297	347	280	344	275	346	247	350	125	352
Borgo Val di Taro	3830	1941	3815	1894	3744	1901	3678	1908	3620	1902	3585	1898	3518	1919	3531	1905	3502	1928	3376	1911	2925	1913
Busseto	3811	1649	3793	1640	3796	1661	3838	1687	3785	1702	3761	1701	3729	1700	3679	1690	3636	1689	3633	1720	3425	1787
Calestano	1173	490	1099	475	1111	478	1163	494	1136	502	1112	516	1125	513	1147	513	1129	508	1127	527	1118	573
Collecchio	8071	2747	7887	2721	8002	2761	8059	2819	8037	2894	8048	2911	8082	2955	8136	2978	8185	2960	8192	3071	8417	3421
Colorno	5112	1615	4975	1584	4971	1594	5042	1629	5023	1659	5000	1650	5018	1660	5061	1683	5043	1709	5031	1725	5036	1863
Compiano	603	303	592	296	602	303	585	313	595	304	588	313	574	302	572	307	558	319	556	316	507	332
Corniglio	1034	722	994	687	987	678	1009	680	1003	685	984	676	966	646	954	639	958	633	939	615	857	520
Felino	4899	1638	4938	1647	4982	1700	4923	1746	4904	1778	4892	1795	4901	1824	4912	1843	4962	1879	4922	1946	4919	2254
Fidenza	14170	5894	13830	5762	13915	5868	14227	5950	14310	6018	14415	6023	14394	6059	14377	6089	14412	6063	14603	6181	15229	6533
Fontanellato	4003	1453	3923	1443	3957	1483	3956	1494	3926	1506	3907	1523	3858	1550	3885	1543	3882	1536	3837	1583	3697	1715
Fontevivo	3281	951	3179	957	3203	973	3255	1005	3241	1011	3232	1033	3239	1057	3224	1086	3168	1097	3198	1136	3153	1330
Fornovo di Taro	3354	1432	3267	1420	3235	1446	3221	1487	3182	1491	3162	1475	3131	1459	3142	1449	3117	1446	3042	1472	2777	1498
Langhirano	5555	1851	5519	1827	5602	1841	5676	1896	5727	1921	5707	1967	5702	2010	5714	2003	5699	2046	5795	2101	6027	2387
Lesignano de' Bagni	2880	734	2857	724	2929	748	2981	774	2939	824	2953	827	2953	836	2921	862	2918	873	2962	920	3023	1120
Medesano	5997	2048	5962	2022	5985	2078	6008	2124	5975	2163	5945	2182	5960	2191	5985	2197	6010	2242	5982	2297	5983	2561
Monchio delle Corti	520	392	495	381	474	388	481	393	462	387	453	384	438	372	433	361	418	369	394	361	279	329
Montechiarugolo	6151	2086	6052	2075	6051	2165	6044	2247	6016	2299	5993	2351	5979	2362	5997	2411	6043	2436	5957	2561	5825	3045
Neviano degli Arduini	1907	1112	1852	1075	1850	1063	1896	1054	1909	1051	1861	1050	1860	1045	1863	1026	1854	1027	1853	1003	1821	915
Noceto	7337	2314	7276	2334	7360	2402	7401	2456	7367	2477	7297	2560	7227	2573	7252	2612	7205	2610	7205	2728	7044	3139
Palanzano	606	442	563	416	538	422	549	431	538	426	545	420	526	412	550	411	541	415	518	406	463	381
Parma	106207	39370	98375	38965	98816	39612	105104	40257	106117	41104	107385	41530	108207	41915	109008	41766	110449	42901	112512	43665	124167	48399
Pellegrino Parmense	558	379	540	369	534	369	535	374	528	367	518	364	508	355	493	358	487	352	473	347	391	317
Roccabianca	1697	713	1663	699	1661	708	1660	720	1639	729	1615	750	1559	761	1546	750	1535	725	1495	762	1287	818
Sala Baganza	3151	991	3159	1003	3194	1038	3130	1082	3133	1106	3094	1124	3071	1155	3070	1173	3083	1190	3038	1254	2902	1518
Salsomaggiore Terme	10863	4699	10478	4616	10615	4721	10539	4748	10471	4750	10552	4806	10481	4828	10507	4804	10266	4704	10275	4827	9849	4970
San Secondo Parmense	3124	1143	3073	1098	3065	1123	3127	1150	3123	1159	3147	1166	3130	1179	3109	1189	3099	1197	3127	1218	3153	1321
Solignano	1031	423	1008	427	1002	425	995	434	1014	448	996	444	995	451	985	444	941	447	952	459	878	494
Soragna	2737	938	2738	918	2751	938	2724	967	2675	983	2683	981	2676	1000	2676	1014	2642	1027	2625	1052	2498	1183
Terenzo	661	372	635	354	619	370	612	378	601	387	616	382	615	380	622	376	619	370	601	383	566	396
Tizzano Val Parma	1127	630	1094	607	1100	592	1108	596	1102	606	1084	584	1082	595	1081	589	1091	587	1072	575	1032	537
Tornolo	579	397	559	380	533	374	525	383	503	386	492	376	488	374	478	361	460	355	429	353	288	314
Torrile	4772	1009	4466	997	4600	1037	4603	1072	4589	1092	4573	1126	4579	1167	4553	1197	4453	1205	4468	1270	4287	1553
Traversetolo	5229	1761	5188	1741	5210	1756	5250	1816	5214	1867	5161	1887	5182	1917	5140	1962	5206	1960	5160	2037	5096	2346
Valmozzola	293	218	283	208	272	215	269	218	264	221	259	219	244	231	243	214	250	216	228	222	169	229
Varano de' Melegari	1508	542	1470	529	1471	538	1466	551	1462	565	1452	566	1448	582	1416	579	1448	571	1414	595	1337	657
Varsi	639	479	630	464	628	457	629	462	619	457	595	454	573	454	575	452	569	439	547	437	449	402
Sissa Trecasali	4459	1581	4463	1578	4464	1595	4418	1593	4364	1620	4323	1636	4281	1654	4281	1653	4253	1663	4184	1691	3879	1810
Polesine Zibello	1836	811	1802	799	1791	801	1791	794	1742	805	1707	816	1695	800	1720	787	1716	778	1655	784	1488	760
Sorbolo Mezzani	7524	2332	7428	2339	7420	2379	7346	2422	7282	2446	7169	2469	7103	2488	7087	2494	7007	2494	6873	2567	6222	2797
Total for every group	**248139**	**94192**	**237587**	**93003**	**238652**	**94580**	**245380**	**96223**	**245578**	**97742**	**246249**	**98544**	**246346**	**99339**	**247074**	**99373**	**247876**	**100586**	**249131**	**102717**	**256410**	**112139**
Total for every year	**342331**		**330590**		**333232**		**341603**		**343320**		**344793**		**345685**		**346447**		**348462**		**351848**		**368549**	

Table 5. Demand prediction based on previous years data.

Finally, we can introduce three demand scenarios in the ***Table 6***, where:

- **T_p** is adjustment coefficient, which is equal to 1, if municipality does not meet any threshold limits combination, otherwise equal to appropriate adjustment coefficient, given above (***see page #18)*** [4].
- **p** is expected population for the specific year (2021 or 2031).
- **d** is resulting demand for the Air Transport in the Province of Parma.
- **g** is travel purpose – we will separate work and non-work to transit the time to the accessibility costs

early post pandemic	**consolidated post pandemic**	**back to normal after 10 years**
2021 (first 2 quarters, pax/year)	**2021 (last 2 quarters, pax/year)**	**2031**
Only business trips allow (if less than p*0,04, then take this limit as value)	All trips allowed, but due to some new regularities and post pandemic effects (like travel is allowed only for vaccinated people or for those who made Covid test maximum 1-2 days before travel) the demand still will not be as usual. So, I assume in this situation update pandemic drop in travel from 0,4 to 0,7 coefficient. (if less than p*0,04, then take this limit as value)	Normal situation, demand will be counted for expected value of population. (if less than p*0,04, then take this limit as value)
$d = p*1,57*0,24*0,68*0,4*g*T_p$	**$d = p*1,57*0,24*0,68*0,7*g*T_p$**	**$p = p*1,57*0,24*0,68*g*T_p$**
1,57 - average trip per inhabitant 0,24 - part of trips made abroad 0,68 - part of overseas trips made by aircraft 0,15 - since only business trips allowed, this is part of trips made for business purposes 0,4 - pandemic decrease in Air Passenger Transportation demand, observed by UNO. Still valid for current scenario g - purpose of travel, in this case - work only, so 15% of all trips. **divide on 2 to get the first half of 2021 (firtst 2 quarters)**	… all values the same, but 0,7 - I am expecting that pandemic decrease in Air Passenger Transportation demand will slow down, but still exist, so I am changing it from 0,4 to 0,7. g - purpose of travel. **divide on 2 to get the first half of 2021 (firtst 2 quarters)**	… all values the same, without "pandemic' adjustments g - purpose of travel.

Table 6. Demand generation scenarios for Air Transport

These scenarios calculated in ***Code Cell # 13***, ***Code Cell # 14***, ***Code Cell # 15*** with following logics:

- Firstly, determine the thresholds.
- For each municipality separetaly:
  - check whether it meet threshold combination or not
  - apply rule of thumb and use it as a limit for demand
  - calculate demand generated by current municipality for every scenario

[4] If you are using Adobe Reader or Microsoft Word, just press **Ctrl + page number** and document will open requested page automatically.

The resulting data is shown in ***Table 7***. As we can see, for the first scenario rule of thumb is applied for every municipality. Which, indeed, seems to be very realistic, since we expect very low demand for the early pandemic situation with travel restrictions. Also, by looking on adjustment coefficient column and compare with coefficients **T_p**, we can understand which municipality meets some threshold combinations.Thus, we got demand generated by each municipality, but now we need to split it in destinations.

Municipalities	adj_coef	d_1_scenario (work only)	rule_of_thumb_applied_1	d_2_work_scenario	d_2_non_work_scenario	rule_of_thumb_applied_2	d_3_work_scenario	d_3_non_work_scenario	rule_of_thumb_applied_3
Albareto	1	70	Yes	47	268	No	66	371	No
Bardi	0.8	71	Yes	38	218	No	48	274	No
Bedonia	0.9	107	Yes	65	368	No	81	461	No
Berceto	0.9	68	Yes	41	234	No	53	301	No
Bore	0.9	24	Yes	14	82	No	16	93	No
Borgo Val di Taro	1	211	Yes	142	806	No	186	1054	No
Busseto	1	214	Yes	144	816	No	200	1135	No
Calestano	1	66	Yes	44	252	No	65	368	No
Collecchio	1.1	451	Yes	333	1889	No	500	2836	No
Colorno	1	270	Yes	182	1030	No	265	1503	No
Compiano	1	35	Yes	23	133	No	32	183	No
Corniglio	1	62	Yes	42	237	No	53	300	No
Felino	1.1	275	Yes	203	1152	No	303	1718	No
Fidenza	1.1	831	Yes	615	3485	No	920	5214	No
Fontanellato	1	217	Yes	146	826	No	208	1179	No
Fontevivo	1	173	Yes	117	661	No	172	976	No
Fornovo di Taro	1	181	Yes	121	688	No	164	931	No
Langhirano	1.1	316	Yes	234	1324	No	356	2016	No
Lesignano de' Bagni	1.1	155	Yes	115	651	No	175	993	No
Medesano	1.1	331	Yes	245	1388	No	361	2047	No
Monchio delle Corti	0.9	30	Yes	18	104	No	21	119	No
Montechiarugolo	1	341	Yes	229	1299	No	341	1932	No
Neviano degli Arduini	1	114	Yes	77	435	No	105	596	No
Noceto	1	397	Yes	267	1514	No	391	2218	No
Palanzano	0.8	37	Yes	20	113	No	26	147	No
Parma	1.1	6247	Yes	4622	26191	No	7296	41342	No
Pellegrino Parmense	0.9	33	Yes	20	113	No	24	139	No
Roccabianca	1	90	Yes	61	344	No	81	458	No
Sala Baganza	1	172	Yes	115	654	No	170	963	No
Salsomaggiore Terme	1	604	Yes	406	2302	No	570	3227	No
San Secondo Parmense	1.1	174	Yes	129	729	No	189	1072	No
Solignano	1	56	Yes	38	215	No	53	299	No
Soragna	1	147	Yes	99	561	No	141	802	No
Terenzo	1	39	Yes	26	150	No	37	210	No
Tizzano Val Parma	1	66	Yes	44	251	No	60	342	No
Tornolo	0.8	31	Yes	17	95	No	19	105	No
Torrile	1	230	Yes	154	875	No	224	1272	No
Traversetolo	1.1	288	Yes	213	1207	No	315	1783	No
Valmozzola	0.8	18	Yes	10	55	No	12	69	No
Varano de' Melegari	1	80	Yes	54	306	No	77	434	No
Varsi	0.8	39	Yes	21	120	No	26	148	No
Sissa Trecasali	1	235	Yes	158	896	No	219	1239	No
Polesine Zibello	1	98	Yes	66	372	No	86	490	No
Sorbolo Mezzani	1	378	Yes	254	1439	No	347	1964	No

Table 7. Demand, generated by each municipality in different scenarios.

As I told earlier, I am not agreeing that airport attractiveness will be represented only through its distance from the municipalities. In fact, most of the users, who are travelling abroad, will choose either Milan Malpensa, Bologna or Pisa airports (Venice is far), because these airports provide much more flights abroad annually, than Parma airport. Not only that, but also uniqueness of provided flights – if you do not have flight in special direction, why would you choose this airport? Asking all these question to myself and remembering my final task of this exercise – distribute demand among all airports, I decided to use famous **Gravity** model for demand distribution.

$$\mathbf{d_{od} = \propto_o^{gen} * \propto_d^{att} * \ d_o^{gen} * \ d_d^{att} * f(c_{od})}$$

Where:

- $\mathbf{d_{od}}$ is demand from origin O to destination D
- $\propto_{\mathbf{o}}^{\mathbf{gen}}, \propto_{\mathbf{d}}^{\mathbf{att}}$ are special demand parameters, used to extract certain demand generated/attracted by certain origin/destination to/from certain destination/origin.
- $\mathbf{d_o^{gen}}$ demand generated by the origin O
- $\mathbf{d_d^{att}}$ demand attracted by destination D
- $\mathbf{f(c_{od})}$ impedance function, which is playing the same role, as distance in gravity mode : higher the value, less the demand (gravity) between the OD couple. I used the following function:

$$\mathbf{f(c_{od})} = e^{-\beta * c_{od}}$$

where:

  - c_{od} is cost from o to d
  - $-\beta$ is calibration parameter, which is equal to 1 by default, but we can change it

In order to perform further analysis, we need to make transition from time values to cost. All data needed for this transformation is provided by Professor Maria Vittoria Corazza. Final surface access costs are represented in the table below:

	2021	Current		
		Resource cost	Perceived cost	Market Price
Avg car driver/ passenger (working)	-	18.76	18.76	22.685
Other (non-working)	-	3.68	4.46	4.46
	Growth 2021-2031	2031		
		Resource cost	Perceived cost	Market Price
Avg car driver/ passenger (working)	1.55	29.078	29.078	35.162
Other (non-working)	1.24	4.5632	5.5304	5.5304

Table 8.
Surface access costs per hour for two class of users: traveling in working and non-working times. Table divided on the current and future scenario.

The transition of the time to cost is performed dynamically while calculating impedance function for the Gravity model and operated in such way, that for every class of user and current scenario appropriate perceived cost was chosen and multiplied on time value. ***Thus, we got direct costs influences on our user choices.***

The problem of the Gravity model is that we need to define two unknowns – $\propto_o^{gen}, \propto_d^{att}$. This is not trivial task, since we have non-linear system of equation which maybe will be solved as doubly constrained problem. However, I decided to go with different solution, which is origin constrained problem.

In the case of origin constrained the d_d^{att} are proxy variables. So, they do not represent real attracted demand, rather than represent the weight of attractor. In our current task as weights, I used number of overseas passenger flights, organized by every airport (destination, so attractor). Data is obtained from https://assaeroporti.com/statistiche_201912/ , and following values of overseas passenger flights per year were applied for every airport:

- Parma: 724 flights
- Milan: 178 460 flights
- Verona: 18 169 flights
- Venice: 76 579 flights
- Bologna: 59 181 flights
- Pisa: 27 274 flights

Such trick allows me to give higher attractiveness to the airport which has higher number of regular overseas flights and thus, has higher number of unique flights. Although the weight can be much higher for big airports, this weight will be reduced by impedance function, if this airport situated far from origin. Accordingly, $\propto_d^{att}$ is equal to 1 for every d. In this case we obtain following formulations:

$$d_{od} = d_o^{gen} * p_{o/d}$$

Where $p_{o/d}$ is probability of choosing certain destination from the current origin and equal to:

$$p_{o/d} = \frac{d_d^{att} * e^{-\beta * c_{od}}}{\sum d_j^{att} * e^{-\beta * c_{oj}}}$$

Now we have all formulations and can count the final demand between each municipality and airport. But before just couple words about those betas, that we applied to adjust our cost function. If choose all betas equal to 1, in the end our results will show that no one is using Parma Airport. This is not realistic, even though the attractiveness of big airports is higher, there are users whose perception of increase in travel time is much more negative, than simply cost of time. Therefore, I adjusted betas in the following way:

- For Parma airport beta is equal to 0.01 since it is closest airport and much more attractive in terms of time.
- For Milan and Venice betas is equal to 0.14 – Milan is quite far, also very crowdy and overloaded (lower service experience probability is high), Venice is even more far, but has higher touristic attractiveness, therefore we compensate it with beta value equal to not 4, but 3.
- For Verona Airport and Pisa Airport betas equal to 0.12 – they are a bit closer than Milan and Venice, therefore less perception of disutility in terms of time.
- Bologna Airport beta equal to 0.1 – extremely hard decision made by me. Bologna Airport is second closest Airport to the Parma region (see ***Table 3***), and incredibly attractive since it has much higher number of operations for passengers. I expected beta equal to 0.08, but results were awful for Parma Airport – the choices were mostly made to the Bologna Airport as being more than 80% percent in average, while only 15% in average and even less of users will choose Parma Airport. *I added some points more to the beta of Bologna, thus decreased its attractiveness. But*

this additional few points should be somehow represented in the future in real life, to increase attractiveness of Parma Airport (this can be level of service, for example).

Total results of demand from Province of Parma to local airport are shown in ***Table 9***. Detailed results are represented in ***Table 10***, ***Table 11***, ***Table 12***.

Demand for the Parma Airport from Province of Parma		
Scenario 1 (first part of 2021)	Scenario 2 (second part of 2021)	Scenario 3 (2031)
4047	**18952**	**27897**

Table 9. Results for demand for the Parma Airport from Province of Parma, pax/year [5].

[5] To get annual demand for 2021 we need to sum first and second scenario and divide it by 2.

Municipalities	parma	milan	verona	venice	bologna	pisa
Albareto	44	0	0	0	22	3
Bardi	46	1	0	0	24	1
Bedonia	90	0	0	0	15	2
Berceto	36	1	0	0	25	6
Bore	14	1	0	0	8	0
Borgo Val di Taro	117	2	1	0	79	12
Busseto	69	18	12	0	113	1
Calestano	32	1	1	0	31	1
Collecchio	112	11	8	0	314	6
Colorno	58	5	14	0	193	1
Compiano	26	0	0	0	8	1
Corniglio	41	0	0	0	19	2
Felino	107	3	3	0	160	2
Fidenza	198	58	20	0	549	6
Fontanellato	56	13	5	0	141	2
Fontevivo	40	6	3	0	123	2
Fornovo di Taro	53	5	3	0	115	5
Langhirano	112	3	4	0	196	2
Lesignano de' Bagni	55	1	2	0	97	1
Medesano	93	11	5	0	215	7
Monchio delle Corti	24	0	0	0	4	2
Montechiarugolo	63	3	7	0	267	1
Neviano degli Arduini	41	1	1	0	71	0
Noceto	101	15	6	0	270	5
Palanzano	26	0	0	0	10	1
Parma	1289	120	136	3	4672	27
Pellegrino Parmense	17	1	1	0	14	1
Roccabianca	34	6	5	0	44	0
Sala Baganza	60	7	8	0	95	1
Salsomaggiore Terme	174	10	9	0	404	7
San Secondo Parmense	59	9	4	0	100	1
Solignano	16	2	2	0	37	0
Soragna	42	3	6	0	96	1
Terenzo	17	1	0	0	21	1
Tizzano Val Parma	17	4	2	0	43	0
Tornolo	6	0	1	0	23	0
Torrile	96	4	2	0	122	6
Traversetolo	158	1	2	0	125	2
Valmozzola	16	0	0	0	2	1
Varano de' Melegari	15	2	2	0	61	0
Varsi	9	0	1	0	29	0
Sissa Trecasali	147	2	1	0	74	11
Polesine Zibello	31	2	1	0	60	3
Sorbolo Mezzani	190	5	3	0	174	6
Total	**4047**	**336**	**284**	**5**	**9261**	**141**

Table 10. Demand for all airports from the Province of Parma, 1st Scenario (pax/year).

Municipalities	parma	milan	verona	venice	bologna	pisa
Albareto	200	2	2	0	98	14
Bardi	165	2	1	0	85	3
Bedonia	364	1	1	0	60	7
Berceto	144	2	2	0	101	25
Bore	57	4	2	0	32	1
Borgo Val di Taro	526	8	6	0	353	55
Busseto	311	83	55	1	506	4
Calestano	142	4	3	0	141	7
Collecchio	554	54	37	1	1548	28
Colorno	258	21	63	1	864	5
Compiano	115	1	1	0	36	5
Corniglio	183	1	1	0	84	10
Felino	528	14	15	0	788	10
Fidenza	978	285	99	1	2708	29
Fontanellato	250	57	21	0	634	9
Fontevivo	177	27	13	0	550	9
Fornovo di Taro	236	20	12	0	518	23
Langhirano	551	14	19	0	966	8
Lesignano de' Bagni	271	6	10	0	476	3
Medesano	460	52	24	1	1060	36
Monchio delle Corti	98	0	0	0	14	10
Montechiarugolo	284	13	31	1	1195	4
Neviano degli Arduini	185	2	7	0	317	1
Noceto	455	67	29	1	1209	22
Palanzano	93	0	0	0	34	4
Parma	6360	590	672	14	23046	132
Pellegrino Parmense	67	5	2	0	57	2
Roccabianca	154	27	23	0	199	2
Sala Baganza	270	31	38	1	425	5
Salsomaggiore Terme	778	47	42	1	1811	30
San Secondo Parmense	292	47	18	0	494	6
Solignano	70	9	7	0	165	2
Soragna	189	13	25	0	429	3
Terenzo	74	3	2	0	93	4
Tizzano Val Parma	77	19	7	0	191	2
Tornolo	22	2	5	0	83	0
Torrile	430	17	11	0	547	25
Traversetolo	779	6	10	0	615	10
Valmozzola	56	0	0	0	7	2
Varano de' Melegari	67	7	11	0	274	2
Varsi	32	1	3	0	105	0
Sissa Trecasali	659	7	5	0	333	49
Polesine Zibello	141	10	6	0	269	12
Sorbolo Mezzani	850	21	14	0	779	29
Total	**18952**	**1601**	**1352**	**26**	**44299**	**647**

Table 11. Demand for all airports from the Province of Parma, 2nd Scenario (pax/year).

Municipalities	parma	milan	verona	venice	bologna	pisa
Albareto	277	3	2	0	136	20
Bardi	208	2	2	0	107	3
Bedonia	456	1	1	0	76	9
Berceto	185	3	2	0	131	33
Bore	65	5	2	0	37	1
Borgo Val di Taro	687	11	8	0	462	72
Busseto	433	115	77	1	704	6
Calestano	208	6	4	0	206	10
Collecchio	832	81	56	1	2325	43
Colorno	377	31	92	2	1260	7
Compiano	158	1	1	0	49	6
Corniglio	232	1	2	0	106	13
Felino	788	21	22	0	1176	15
Fidenza	1463	426	148	2	4051	43
Fontanellato	357	82	29	0	905	14
Fontevivo	262	41	20	0	813	13
Fornovo di Taro	320	28	16	0	700	32
Langhirano	839	22	28	1	1470	12
Lesignano de' Bagni	413	8	16	0	726	4
Medesano	679	77	36	1	1563	53
Monchio delle Corti	112	0	0	0	17	11
Montechiarugolo	423	19	46	1	1778	6
Neviano degli Arduini	253	3	9	0	434	1
Noceto	666	97	42	1	1770	33
Palanzano	121	0	1	0	45	6
Parma	10039	931	1061	21	36378	208
Pellegrino Parmense	83	6	2	0	70	3
Roccabianca	205	36	30	0	265	3
Sala Baganza	397	46	56	1	626	7
Salsomaggiore Terme	1091	65	58	1	2539	42
San Secondo Parmense	430	68	27	0	727	8
Solignano	97	13	10	0	229	3
Soragna	270	19	35	1	614	4
Terenzo	104	4	3	0	130	6
Tizzano Val Parma	104	26	9	0	259	3
Tornolo	25	2	5	0	91	0
Torrile	625	25	16	0	795	36
Traversetolo	1151	9	15	0	908	14
Valmozzola	71	0	0	0	8	2
Varano de' Melegari	94	10	16	0	388	2
Varsi	39	1	3	0	130	0
Sissa Trecasali	912	9	7	0	461	67
Polesine Zibello	186	13	8	0	354	15
Sorbolo Mezzani	1160	28	20	0	1063	40
Total	**27897**	**2395**	**2041**	**39**	**67079**	**928**

Table 12. Demand for all airports from the Province of Parma, 3rd Scenario (pax/year).

Regional demand.

It is easily seen that demand for Parma Airport is generated not only by population of the Parma region, though we have all data about its real demand during last years in ***Figure 1***. Indeed, Parma Airport is acceptable from all other provinces of Emilia-Romagna. It could better to perform further analysis exactly in the same way as we did before – use precise population data for every municipality of every province and then apply this all for the Gravity Model. However, for this project it will not be time efficiently, which means that we can apply lest time-consuming method with enough demand prediction accuracy.

In order to forecast demand for Parma Airport in different scenarios I decided to use ratios of predicted demand scenarios of the Parma province to its total population as reference ratios. Thus, I expect approximately similar proportions of demand from other provinces, which is not very right, since we know that there is influence of the impedance function in terms of travel time (access costs) - so, speaking more generally, as far province from Parma Airport as less should we expect the demand. But such uncertainty is covering with other factors that may increase demand for observing airport and was not considering in this project. The final results are shown in table below.

Municipalities	2021 population	2031 population	sc_1	sc_2	sc_3
Piacenza	286548	285530	2527	11835	16430
Parma	458851	484808	4047	18952	27897
Reggio nell'Emilia	538740	552952	4752	22252	31818
Modena	707392	722074	6239	29218	41550
Bologna	1025890	1067552	9048	42373	61429
Ferrara	343521	329415	3030	14189	18955
Ravenna	393533	398655	3471	16254	22940
ForlГ¬-Cesena	395258	396739	3486	16325	22829
Rimini	344271	362903	3036	14219	20882
Total	13482012	13801884	**118909**	**556850**	**794193**

Table 13. Final total regional demand for all scenarios for the Parma Airport (pax/year)

As a conclusion I want to mention some moments that can improve demand analysis and make it appropriate for the real-life application. Firstly, it could be much better to have and analyse data of every municipality from the neighbourhood provinces, which may have access to this airport, instead of using values proportional to the Province of Parma. Secondly, in the Gravity model we rely so much on the beta calibration parameters, which give a meaningful sensitiveness to the impedance function, however estimated directly from the perception of modeller (me in this current case). Betas can be identified from historical data of the real demand for each alternative (airport), but unfortunately this data was not available in our case.

Step 3.
Service configuration.

The objective of the current step is establishing appropriate supply that will not only meet current demand, but also will allow to reassign different demand flows to the Parma Airport if possible through transfers, although some restrictions will be also considered.

Generally, the scheme where we want to allocate our airport consists 9 locations, as it is shown in ***Figure 10***. Arrows in this figure represents demand patterns, each demand pattern will have appropriate value according to the post pandemic scenarios that we calculated before. These values will be proportional to the given pattern demand, which are following:

1 - 130 pax/day
2 - 140 pax/day
3 - 60 pax/day
4 - 250 pax/day
5 - 110 pax/day

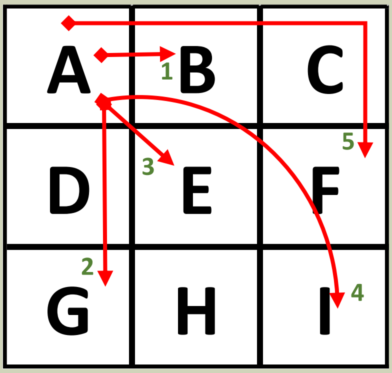


Figure 10. Airports allocation and demand patterns.

Overall, in the current analysis we considered three configurations:

1. Point-to-point service exists for all locations, so we will have from every location 8 routes.
2. Point-to-point adapted - the direct services will be offered only on the routes, where daily service frequency higher than a minimum value.
3. Hub-&-spoke – the central airport becomes the only place, where all flights will originate or terminate.

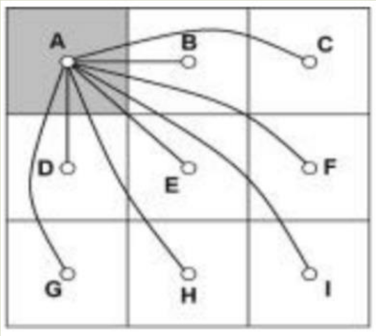
General suggestions – 40 seats aircraft, 70% load factor with increase up to 80% in next 10 years. Point-to-point adapted – demand for pattern 4 and 5 reduced by 35%. Hub-&-spoke - demand pattern 1 reduced by 40%.

Before diving into analysis, I want to mention that it may seem so trivial while we want to rehabilitate some airport the solution is based on putting it in the centre with hub-&-spoke scenario, thus forcing all passengers to come through this point. However, reality with 40-seats available aircrafts and dramatically increased activities dictates its own drawbacks, which we will see further.

Demand patterns	Pattern Shape	Reference values pax/day	Demand distribution		
			1 scenario	2 scenario	3 scenario
1		130	61	288	410
2		140	66	310	442
3		60	28	133	189
4		250	118	553	788
5		110	52	243	347
total pax/day		690	326	1526	2176

Table 14. Demand patterns for the 1st, 2nd and 3rd scenario without any adjustments.

Point-to-point.



Scenario 1.

	corner (for example, A)	demand distribution		central edge (for example, B)	demand distribution		centre	demand distribution
	A-B	61		B-A	61		E-A	28
	A-C	66		B-C	61		E-B	61
	A-D	61		B-D	28		E-C	28
	A-E	28		B-E	61		E-D	61
	A-F	52		B-F	28		E-F	61
	A-G	66		B-G	52		E-G	28
	A-H	52		B-H	66		E-H	61
	A-I	118		B-I	52		E-I	28
sum		506			411			359
locations		4			4			1
Total				4025				

Table 15. P2P. Demand distribution according to the given patterns, 1st scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		61		3	120	61
2		66		3	120	66
3		28	70.00%	2	80	28
4		118		5	200	118
5		52		2	80	52

Table 16. P2P. Supply generated for each pattern type, 1st scenario.

		1	2	3	4	5		
Demand pax/day		61	66	118	28	52		
Supply units		3		5	2			
Position	Location	Frequency Matrix					Total Flights	
corner	4	2	2	1	1	2	13	
central edge	4	3	1	0	2	2	13	
centre	1	4	0	0	4	0	20	
Position	Location	Service matrix						
corner	4	8	8	4	4	8	2	flights /day
central edge	4	12	4	0	8	8	3	and more
centre	1	4	0	0	4	0		
Position	Locations	Supply					Total Flights	
corner	4	24	0	20	8	0	52	
central edge	4	36	0	0	16	0	52	
centre	1	12	0	0	8	0	20	
							124	*Flights*
							32	*pax/flight*

Table 17. P2P. Frequency, service and supply matrices for the 1st Scenario.

For the current configuration and 1st scenario we have followings:

1. In every corner airport there are 13 flights originating on 8 routes, edge airport – same situation, centre airport – 20 flights on 8 routes.
2. 32 services have 2 flights/day, 40 services – 3 and more flights/day, overall there are 72 service.
3. Overall 124 flights with average 32 passengers/flight, this value **exceeds** given load factor.

Scenario 2.

	corner (for example, A)	demand distribution		central edge (for example, B)	demand distribution		centre	demand distribution
	A-B	288		B-A	288		E-A	133
	A-C	310		B-C	288		E-B	288
	A-D	288		B-D	133		E-C	133
	A-E	133		B-E	288		E-D	288
	A-F	243		B-F	133		E-F	288
	A-G	310		B-G	243		E-G	133
	A-H	243		B-H	310		E-H	288
	A-I	553		B-I	243		E-I	133
sum		2366			1924			1681
locations		4			4			1
Total	18843							

Table 18. P2P. Demand distribution according to the given patterns, 2nd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		288	70.00%	11	440	288
2		310		12	480	310
3		133		5	200	133
4		553		20	800	553
5		243		9	360	243

Table 19. P2P. Supply generated for each pattern type, 2nd scenario.

		1	2	3	4	5		
Demand pax/day		288	310	553	133	243		
Supply units		11	12	20	5	9		
Position	**Location**	**Frequency Matrix**					Total Flights	
corner	4	2	2	1	1	2	89	
central edge	4	3	1	0	2	2	73	
centre	1	4	0	0	4	0	64	
Position	**Location**	**Service matrix**						
corner	4	8	8	4	4	8	5	flights /da
central edge	4	12	4	0	8	8	9	and more
centre	1	4	0	0	4	0		
Position	**Locations**	**Supply**					**Total Flights**	
corner	4	88	96	80	20	72	356	
central edge	4	132	48	0	40	72	292	
centre	1	44	0	0	20	0	64	
							712	*Flights*
							26	*pax/flight*

Table 20. P2P. Frequency, service and supply matrices for the 2nd Scenario.

For the current configuration and 2nd scenario we have followings:

1. In every corner airport there are 89 flights originating on 8 routes, edge airport – 73 flights on 8 routes, centre airport – 64 flights on 8 routes.
2. 16 services have 5 flights/day, 56 services – 9 and more flights/day, overall there are 72 service.
3. Overall 712 flights with average 26 passengers/flight.

Scenario 3.

	corner (for example, A)	demand distribution		central edge (for example, B)	demand distribution		centre	demand distribution
	A-B	410		B-A	410		E-A	189
	A-C	442		B-C	410		E-B	410
	A-D	410		B-D	189		E-C	189
	A-E	189		B-E	410		E-D	410
	A-F	347		B-F	189		E-F	410
	A-G	442		B-G	347		E-G	189
	A-H	347		B-H	442		E-H	410
	A-I	788		B-I	347		E-I	189
sum		3374			2744			2397
locations		4			4			1
Total	26869							

Table 21. P2P. Demand distribution according to the given patterns, 3rd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		410	80.00%	13	520	410
2		442		14	560	442
3		189		6	240	189
4		788		25	1000	788
5		347		11	440	347

Table 22. P2P. Supply generated for each pattern type, 3rd scenario.

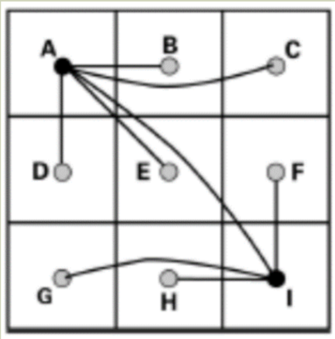
		1	2	3	4	5		
Demand pax/day		410	442	788	189	347		
Supply units		13	14	25	6	11		
Position	**Location**	**Frequency Matrix**					Total Flights	
corner	4	2	2	1	1	2	107	
central edge	4	3	1	0	2	2	87	
centre	1	4	0	0	4	0	76	
Position	**Location**	**Service matrix**						
corner	4	8	8	4	4	8	6	flights /day
central edge	4	12	4	0	8	8	11	and more
centre	1	4	0	0	4	0		
Position	**Locations**	**Supply**					**Total Flights**	
corner	4	104	112	100	24	88	428	
central edge	4	156	56	0	48	88	348	
centre	1	52	0	0	24	0	76	
							852	*Flights*
							32	*pax/flight*

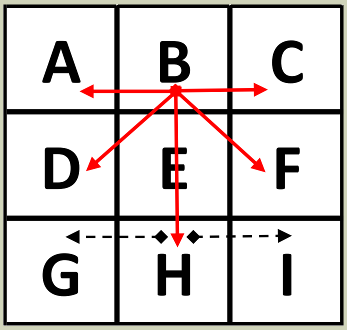
Table 23. P2P. Frequency, service and supply matrices for the 3rd Scenario.

For the current configuration and 3rd scenario we have followings:

1. In every corner airport there are 107 flights originating on 8 routes, edge airport – 87 flights on 8 routes, centre airport – 76 flights on 8 routes.
2. 16 services have 6 flights/day, 56 services – 11 and more flights/day, overall there are 72 services.
3. Overall 852 flights with average 32 pax/flight.

Point-to-point adapted.





Scenario 1.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	61		B-A	61		E-A	28	
	A-C	66		B-C	61		E-B	61	
	A-D	61		B-D	28		E-C	28	
	A-E	28		B-E	61		E-D	61	
	A-F	-	134	B-F	28	68	E-F	61	0
	A-G	-		B-G	-		E-G	28	
	A-H	-		B-H	134		E-H	61	
	A-I	210		B-I	-		E-I	28	
sum		428			375			359	
locations		4			4			1	
Total	3569								

Table 24. P2P adapted. Demand distribution according to the given patterns, 1st scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		61		3	120	61
2		66		3	120	66
3		28	70.00%	2	80	28
4		210		8	320	210
5		134		5	200	134

Table 25. P2P adapted. Supply generated for each pattern type, 1st scenario.

		1	2	3		4		
Demand pax/day		61	66	28	210	134		
Supply units		3		2	8	5		
Position	Location	Frequency Matrix					Total Flights	
corner	4	2	1	1	1	0	16	
central edge	4	3	0	2	0	1	18	
centre	1	4	0	4	0	0	20	
Position	Location	Service matrix						
corner	4	8	4	4	4	0	3	and more
central edge	4	12	0	8	0	4	2	flights /da
centre	1	4	0	4	0	0		
Position	Locations	Supply					Total Flights	
corner	4	24	12	8	32	0	76	
central edge	4	36	0	16	0	20	72	
centre	1	12	0	8	0	0	20	
							168	*Flights*
							21	*pax/flight*

Table 26. P2P adapted. Frequency, service and supply matrices for the 1st Scenario.

For the current configuration and 1st scenario we have followings:

1. In every corner airport there are 16 flights originating on 5 routes, edge airport – 18 flights on 6 routes, centre airport – 20 flights on 8 routes.
2. 16 services have 2 flights/day, 36 services – 3 and more flights/day, overall there are 52 services.
3. Overall 168 flights with average 21 pax/flight.

Scenario 2.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	288		B-A	288		E-A	133	
	A-C	310		B-C	288		E-B	288	
	A-D	288		B-D	133		E-C	133	
	A-E	133		B-E	288		E-D	288	
	A-F	-	626	B-F	133	316	E-F	288	0
	A-G	-		B-G	-		E-G	133	
	A-H	-		B-H	626		E-H	288	
	A-I	985		B-I	-		E-I	133	
sum		2003			1754			1681	
locations		4			4			1	
Total	16706								

Table 27. P2P adapted. Demand distribution according to the given patterns, 2nd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		288		11	440	288
2		310		12	480	310
3		133	70.00%	5	200	133
4		985		36	1440	985
5		626		23	920	626

Table 28. P2P adapted. Supply generated for each pattern type, 2nd scenario.

		1	2	3		4		
Demand pax/day		288	310	133	985	626		
Supply units		11	12	5	36	23		
Position	**Location**	**Frequency Matrix**					Total Flights	
corner	4	2	1	1	1	0	75	
central edge	4	3	0	2	0	1	66	
centre	1	4	0	4	0	0	64	
Position	**Location**	**Service matrix**						
corner	4	8	4	4	4	0	11	and more
central edge	4	12	0	8	0	4	5	flights /da
centre	1	4	0	4	0	0		
Position	**Locations**	**Supply**					Total Flights	
corner	4	88	48	20	144	0	300	
central edge	4	132	0	40	0	92	264	
centre	1	44	0	20	0	0	64	
							628	*Flights*
							27	*pax/flight*

Table 29. P2P adapted. Frequency, service and supply matrices for the 2nd Scenario.

For the current configuration and 2nd scenario we have followings:

1. In every corner airport there are 75 flights originating on 5 routes, edge airport – 66 flights on 6 routes, centre airport – 64 flights on 8 routes.
2. 16 services have 5 flights/day, 36 services – 11 and more flights/day, overall there are 52 services.
3. Overall 628 flights daily with average 21 pax/flight.

Scenario 3.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	410		B-A	410		E-A	189	
	A-C	442		B-C	410		E-B	410	
	A-D	410		B-D	189		E-C	189	
	A-E	189		B-E	410		E-D	410	
	A-F	-	892	B-F	189	451	E-F	410	0
	A-G	-		B-G	-		E-G	189	
	A-H	-		B-H	892		E-H	410	
	A-I	1405		B-I	-		E-I	189	
sum		2856			2501			2397	
locations		4			4			1	
Total	23822								

Table 30. P2P adapted. Demand distribution according to the given patterns, 3rd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		410		13	520	410
2		442		14	560	442
3		189	80.00%	6	240	189
4		1405		44	1760	1405
5		892		28	1120	892

Table 31. P2P adapted. Supply generated for each pattern type, 3rd scenario.

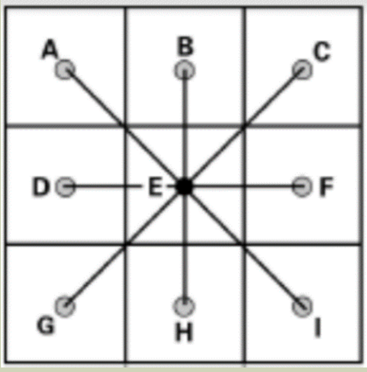
		1	2	3		4		
Demand pax/day		410	442	189	1405	892		
Supply units		13	14	6	44	28		
Position	**Location**	**Frequency Matrix**					Total Flights	
corner	4	2	1	1	1	0	90	
central edge	4	3	0	2	0	1	79	
centre	1	4	0	4	0	0	76	
Position	**Location**	**Service matrix**						
corner	4	8	4	4	4	0	13	and more
central edge	4	12	0	8	0	4	6	flights /day
centre	1	4	0	4	0	0		
Position	**Locations**	**Supply**					Total Flights	
corner	4	104	56	24	176	0	360	
central edge	4	156	0	48	0	112	316	
centre	1	52	0	24	0	0	76	
							752	*Flights*
							32	*pax/flight*

Table 32. P2P adapted. Frequency, service and supply matrices for the 3rd Scenario.

For the current configuration and 3rd scenario we have followings:

1. In every corner airport there are 90 flights originating on 5 routes, edge airport – 79 flights on 6 routes, centre airport – 76 flights on 8 routes.
2. 16 services have 6 flights/day, 36 services – 13 and more flights/day, overall there are 52 services.
3. Overall 752 flights daily with average 32 pax/flight.

Hub & spoke.



Scenario 1.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	-		B-A	-		E-A	456	
	A-C	-		B-C	-		E-B	337	
	A-D	-		B-D	-		E-C	456	
	A-E	456		B-E	337		E-D	337	
	A-F	-	428	B-F	-	300	E-F	337	2914
	A-G	-		B-G	-		E-G	456	
	A-H	-		B-H	-		E-H	337	
	A-I	-		B-I	-		E-I	456	
sum		456			337			3175	
locations		4			4			1	
Total	6350								

Table 33.H&S. Demand distribution according to the given patterns, 1st scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		337	70.00%	13	520	337
2		456		17	680	456

Table 34. H&S. Supply generated for each pattern type, 1st scenario.

		1	2		
Demand pax/day		337	456		
Supply units		13	17		
Position	Location	Frequency Matrix		Total Flights	
corner	4	0	1	17	
central edge	4	1	0	13	
centre	1	4	4	120	
Position	Location	Service matrix			
corner	4	0	4	13	flights/day
central edge	4	4	0	17	flights /day
centre	1	4	4		
Position	Locations	Supply		Total Flights	
corner	4	0	68	68	
central edge	4	52	0	52	
centre	1	52	68	120	
				240	*Flights*
				26	*pax/flight*

Table 35.H&S. Frequency, service and supply matrices for the 1st Scenario.

For the current configuration and 1st scenario we have followings:

1. In every corner airport there are 17 flights originating on 1 route, edge airport – 13 flights on 1 route, centre airport – 120 flights on 8 routes.
2. 8 services have 13 flights/day, 8 services – 17 flights/day, overall there are 16 services.
3. Overall 240 flights daily with average 26 pax/flight.

Scenario 2.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	-		B-A	-		E-A	2136	
	A-C	-		B-C	-		E-B	1579	
	A-D	-		B-D	-		E-C	2136	
	A-E	2136		B-E	1579		E-D	1579	
	A-F	-	2004	B-F	-	1407	E-F	1579	13641
	A-G	-		B-G	-		E-G	2136	
	A-H	-		B-H	-		E-H	1579	
	A-I	-		B-I	-		E-I	2136	
sum		2136			1579			14862	
locations		4			4			1	
Total	29724								

Table 36. H&S. Demand distribution according to the given patterns, 2nd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		1579	70.00%	57	2280	1579
2		2136		77	3080	2136

Table 37. H&S. Supply generated for each pattern type, 2nd scenario.

		1	2		
Demand pax/day		1579	2136		
Supply units		57	77		
Position	**Location**	**Frequency Matrix**		**Total Flights**	
corner	4	0	1	77	
central edge	4	1	0	57	
centre	1	4	4	536	
Position	**Location**	**Service matrix**			
corner	4	0	4	57	flights/day
central edge	4	4	0	77	flights /day
centre	1	4	4		
Position	**Locations**	**Supply**		**Total Flights**	
corner	4	0	308	308	
central edge	4	228	0	228	
centre	1	228	308	536	
				1072	*Flights*
				28	*pax/flight*

Table 38. H&S. Frequency, service and supply matrices for the 2nd Scenario.

For the current configuration and 2nd scenario we have followings:

1. In every corner airport there are 77 flights originating on 1 route, edge airport – 57 flights on 1 route, centre airport – 536 flights on 8 routes.
2. 8 services have 57 flights/day, 8 services – 77 flights/day, overall there are 16 services.
3. Overall 1072 flights daily with average 28 pax/flight.

Scenario 3.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	-		B-A	-		E-A	3046	
	A-C	-		B-C	-		E-B	2252	
	A-D	-		B-D	-		E-C	3046	
	A-E	3046		B-E	2252		E-D	2252	
	A-F	-	2857	B-F	-	2006	E-F	2252	19452
	A-G	-		B-G	-		E-G	3046	
	A-H	-		B-H	-		E-H	2252	
	A-I	-		B-I	-		E-I	3046	
sum		3046			2252			21192	
locations		4			4			1	
Total	42385								

Table 39. H&S. Demand distribution according to the given patterns, 3rd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		2252	80.00%	71	2840	2252
2		3046		96	3840	3046

Table 40. H&S. Supply generated for each pattern type, 3rd scenario.

		1	2		
Demand pax/day		2252	3046		
Supply units		71	96		
Position	**Location**	**Frequency Matrix**		**Total Flights**	
corner	4	0	1	96	
central edge	4	1	0	71	
centre	1	4	4	668	
Position	**Location**	**Service matrix**			
corner	4	0	4	71	flights/day
central edge	4	4	0	96	flights /day
centre	1	4	4		
Position	**Locations**	**Supply**		**Total Flights**	
corner	4	0	384	384	
central edge	4	284	0	284	
centre	1	284	384	668	
				1336	*Flights*
				32	*pax/flight*

Table 41. H&S. Frequency, service and supply matrices for the 3rd Scenario.

For the current configuration and 3rd scenario we have followings:

1. In every corner airport there are 96 flights originating on 1 route, edge airport – 71 flights on 1 route, centre airport – 668 flights on 8 routes.
2. 8 services have 71 flights/day, 8 services – 96 flights/day, overall there are 16 services.
3. Overall 1336 flights daily with average 28 pax/flight.

General comparison of service configuration attributes.

It could be better to have a complex look on all configurations, and such opportunity provided in the table below:

Daily performance	service configuration								
	scenario 1			scenario 2			scenario 3		
	p2p	p2p adapted	hub&spoke	p2p	p2p adapted	hub&spoke	p2p	p2p adapted	hub&spoke
Number of direct services	72	52	16	72	52	16	72	52	16
Range of frequences	2-5	2-8	13-17	5-12	5-36	57-77	6-25	6-44	71-96
Trip passengers	4025	3569	6350	18843	16706	29724	26869	23822	42385
Average pass per flight	32	21	26	26	27	28	32	32	32
Corner airports									
Passengers originating per airport	506	428	456	2366	2003	2136	3374	2856	3046
Transfer passengers (total)	0	535	428	0	2504	2004	0	3570	19452
Total passengers	2022	1711	1826	9466	8010	8546	13498	11422	12186
Routes	8	5	1	8	5	1	8	5	1
Departures per day	13	16	17	89	75	77	107	90	96
Centre edge airports									
Passengers originating per airport	411	375	337	1924	1754	1579	2744	2501	2252
Transfer passengers (total)	0	270	1202	0	1265	5626	0	1804	8023
Total passengers	1644	1499	1349	7696	7015	6316	10975	10003	9007
Routes	8	6	1	8	6	1	8	6	1
Departures per day	13	18	13	73	66	57	87	79	71
Centre airports									
Passengers originating per airport	359	359	3175	1681	1681	14862	2397	2397	21192
Transfer passengers (total)	0	0	2914	0	0	13641	0	0	19452
Total passengers	359	359	3175	1681	1681	14862	2397	2397	21192
Routes	8	8	8	8	8	8	8	8	8
Departures per day	20	20	120	64	64	536	76	76	668

Table 42. Table of attributes for all service configurations.

As I told before, hub airports in the configurations, where transfer is allowed, becomes busiest airports with highest flight frequencies. Such airports can be characterized as international or regional airports. However, in total hub situation we saw, that other airports on the corners and edges became less effective with dramatical drop in the number of passengers originating per airport. This can be unfavourable demand distribution, of course, depends on the actual case, but generally…

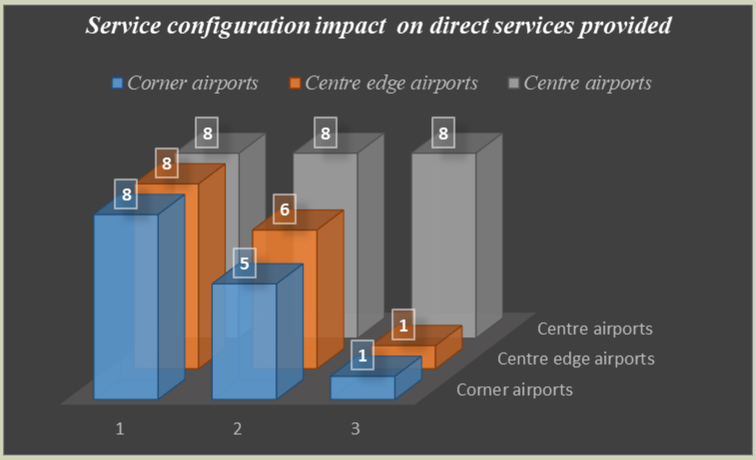


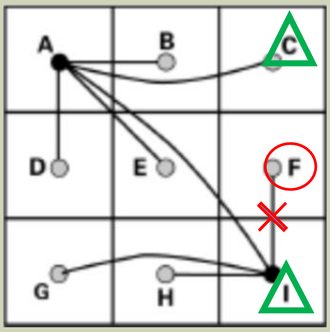
Figure 11. Direct services within different service configuration.

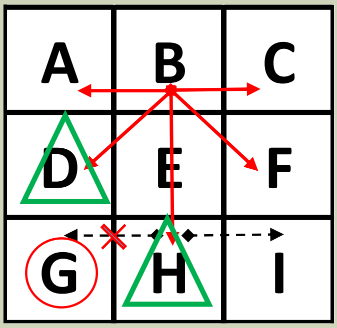
Meanwhile, in the first configuration (point-to-point), we saw that all airports have almost similar demand per airport, thus we obtain some equity of demand distribution. But such configuration does not meet our goal, since we will not influence on demand increase, which we really need for Parma airport, on the one hand. On the other hand, the average pax/flight also exceeds basic requirement with 40-seats aircraft and payload value of 70 % as a start. What we should choose and will our choice guarantee “the best” solution?

The answer for the second question is definitely “no”, because this is very theoretical analysis and indeed, we need more data and attributes for making better approximation. But with all information we already obtained, it is clearly visible that solution is between “point-to-point adapted” and “hub & spoke” configurations with airport “hub” allocation. In order to finish with the 3rd step, we can take a look on the situation, when demand will be partly shifted. This may happen if some of the airports will be out of service. I will analyse only for future demand pattern, when world is expected to return to normal life after pandemic.

“Partly shifted demand” situation.

Point-to-point adapted, 3rd scenario.





From every hub except central airport one route is not available now. This caused decrease in demand by 40%. Passengers, who are originated in corner or edge airports will choose between two alternatives and then will continue their way by car. For example, passengers from airport A traveling to airport F will choose to arrive either in airport C or airport I, then they will travel by car.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	410		B-A	410		E-A	189	
	A-C	509		B-C	410		E-B	410	
	A-D	410		B-D	257		E-C	189	
	A-E	189		B-E	410		E-D	410	
	A-F	-	735	B-F	189	293	E-F	410	0
	A-G	-		B-G	-		E-G	189	
	A-H	-		B-H	735		E-H	410	
	A-I	1247		B-I	-		E-I	189	
sum		2765			2411			2397	
locations		4			4			1	
Total	23101								

Table 43. P2P adapted, shifted demand. Demand distribution according to the given patterns, 3rd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		410		13	520	410
2		509		16	640	509
3		189		6	240	189
4		1247	80.00%	39	1560	1247
5		257		9	360	257
6		735		23	920	735

Table 44. P2P adapted, shifted demand. Supply generated for each pattern type, 3rd scenario.

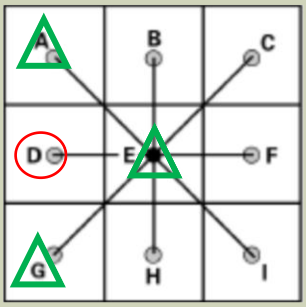
		1	2	3		4			
Demand pax/day		410	509	189	1247	257	735		
Supply units		13	16	6	39	9	23		
Position	Location	Frequency Matrix						Total Flights	
corner	4	2	1	1	1	0	0	87	
central edge	4	3	0	1	0	1	1	77	
centre	1	4	0	4	0	0	0	76	
Position	Location	Service matrix							
corner	4	8	4	4	4	0	0	9	and more
central edge	4	12	0	4	0	4	4	6	flights /day
centre	1	4	0	4	0	0	0		
Position	Locations	Supply						Total Flights	
corner	4	104	64	24	156	0	0	348	
central edge	4	156	0	24	0	36	92	308	
centre	1	52	0	24	0	0	0	76	
								732	*Flights*
								32	*pax/flight*

Table 45. P2P adapted, shifted demand. Frequency, service and supply matrices for the 3rd Scenario.

For the current configuration with shifted demand of the 3rd scenario we have followings:

1. In every corner airport there are 87 flights originating on 5 routes, edge airport – 77 flights on 6 routes, centre airport – 76 flights on 8 routes.
2. 12 services have 6 flights/day, 40 services – 9 and more flights/day, overall there are 52 services.
3. Overall 732 flights daily with average 32 pax/flight.

Hub & Spoke, 3rd scenario.



Before all passengers was travelling through central airport. Now one route of this airport is not functioning. This caused decrease in demand by 40%. Remaining part of demand will choose between one from two corner airports and continue their way by car. For example, passengers are originated on airport F, traveling to airport D, will choose between A and G, then continue with car to D.

	corner (for example, A)	demand distribution	transfer	central edge (for example, B)	demand distribution	transfer	centre	demand distribution	transfer
	A-B	-		B-A	-		E-A	2908	
	A-C	-		B-C	-		E-B	2075	
	A-D	-		B-D	-		E-C	2908	
	A-E	2908		B-E	2075		E-D	2075	
	A-F	-	2718	B-F	-	1829	E-F	2075	18190
	A-G	-		B-G	-		E-G	3945	
	A-H	-		B-H	-		E-H	-	
	A-I	-		B-I	-		E-I	3945	
sum		2908			2075			19931	
locations		4			4			1	
Total	39862								

Table 46. Hub & Spoke, shifted demand. Demand distribution according to the given patterns, 3rd scenario.

Demand patterns	Pattern Shape	Pax/day per pattern	Load Factor	Supply per pattern (units)	Seats generated	Payload (units)
1		2075	80.00%	65	2600	2075
2		2908		91	3640	2908
3		3945		124	4960	3945

Table 47. Hub & Spoke, shifted demand. Supply generated for each pattern type, 3rd scenario.

		1	2	3		
Demand pax/day		2075	2908	3945		
Supply units		65	91	124		
Position	**Location**	**Frequency Matrix**			**Total Flights**	
corner	4	0	1	0	91	
central edge	4	1	0	0	65	
centre	1	3	2	2	625	
Position	**Location**	**Service matrix**				
corner	4	0	4	0	65	flights/day
central edge	4	4	0	0	91	flights /day
centre	1	3	2	2		
Position	**Locations**	**Supply**			**Total Flights**	
corner	4	0	364	0	364	
central edge	4	260	0	0	260	
centre	1	195	182	248	625	
					1249	*Flights*
					32	*pax/flight*

Table 48. Hub & Spoke, shifted demand. Frequency, service and supply matrices for the 3rd Scenario.

For the current configuration with shifted demand of the 3rd scenario we have followings:

1. In every corner airport there are 91 flights originating on 1 route, edge airport – 65 flights on 1 route, centre airport – 625 flights on 7 routes.
2. 6 services have 65 flights/day, 8 services – 91 and more flights/day, overall there are 15 services.
3. Overall 1249 flights daily with average 32 pax/flight.

Airport allocation.

From the previous analysis with partly shifted demand we saw, that although pax/flight keeps stable value for both configurations, the number of routes decreased in the hub center airport of “Hub & Spoke configuration” but did not change at all in the “Point-to-Point adapted” configuration. Moreover, disfunction of just one route caused rapid decrease in demand for “Hub & Spoke”, where “P2P adapted” shows better results with slight decrease in demand. And finally, for the corner airport of “P2P adapted” configuration demand increased!

To conclude, the best allocation according to final results would be corner airport of the “P2P adapted” configuration with very stable service and higher demand potential. Final results of all types of configurations and situations are shown in the ***Table 49***.

Daily performance	service configuration											
	scenario 1			*scenario 2*			*scenario 3*			*sc3 - demand partly shifted*		
	p2p	*p2p adapted*	*hub&spoke*	*p2p*	*p2p adapted*	*hub&spoke*	*p2p*	*p2p adapted*	*hub&spoke*	*p2p*	*p2p adapted*	*hub&spoke*
Number of direct services	72	52	16	72	52	16	72	52	16		52	15
Range of frequences	2-5	2-8	13-17	5-12	5-36	57-77	6-25	6-44	71-96		6-39	65-124
Trip passengers	4025	3569	6350	18843	16706	29724	26869	23822	42385		23101	39862
Average pass per flight	32	21	26	26	27	28	32	32	32		32	32
Corner airports												
Passengers originating per airport	506	428	456	2366	2003	2136	3374	2856	3046		2765	2908
Transfer passengers (total)	0	535	428	0	2504	2004	0	3570	19452		2939	2718
Total passengers	2022	1711	1826	9466	8010	8546	13498	11422	12186		11754	11631
Routes	8	5	1	8	5	1	8	5	1		5	1
Departures per day	13	16	17	89	75	77	107	90	96		87	91
Centre edge airports												
Passengers originating per airport	411	375	337	1924	1754	1579	2744	2501	2252		2411	2075
Transfer passengers (total)	0	270	1202	0	1265	5626	0	1804	8023		1173	7316
Total passengers	1644	1499	1349	7696	7015	6316	10975	10003	9007		9643	8300
Routes	8	6	1	8	6	1	8	6	1		6	1
Departures per day	13	18	13	73	66	57	87	79	71		77	65
Centre airports												
Passengers originating per airport	359	359	3175	1681	1681	14862	2397	2397	21192		2397	19931
Transfer passengers (total)	0	0	2914	0	0	13641	0	0	19452		0	18190
Total passengers	359	359	3175	1681	1681	14862	2397	2397	21192		2397	19931
Routes	8	8	8	8	8	8	8	8	8		8	7
Departures per day	20	20	120	64	64	536	76	76	668		76	625

Table 49. Final table of all configurations and different scenarios.

Step 4.
Airport capacity.

Our last step will be directed for the establishment of the airport capacity, which is a function of so many factors. However, in the current analysis we will take into consideration only runway capacity and number of gates. We will assume that this is single runway, its length (all declared distances), aerodrome areas, obstacle safeguarding are already predetermined and known.

Ultimate capacity is the maximum number of movements per hour that a runway can achieve. It is obtained from mathematical process that considers the mix of traffic. Overall, we will have different scenarios of traffic mix, but below I will give all types of aircrafts used in our analysis:

- Super (1)
- Heavy (2)
- B757 (3) (this and all above aircrafts require "wake-vortex" separation)
- Large (4) (all jet airlines and business jets, and some large propeller types)
- Small (5) (single engine, typical up to six-seat, general aviation types)
- Light (6) (all propeller aircraft, excluding single-engine propeller types)

The separation minima will be based on Instrument Flight Rules. Also, it will be assumed, that arrival-departures are 50% each, and they are conducted alternatively. It means that departing time needed to enter and backtrack the runway will be absorbed in time taken by the arrival to pass its point of entry and leave the runway. Length of final approach (n) is 8 n.mi. Parameters for gates quantity determination: T = 45 min, s = 15 min.

Matrices of separation requirements are following (in nautical miles):

		1	2	3	4
(Heavy)	1	4	5	6	8
(Large)	2	3	3	4	4
(Small)	3	3	3	3	3
(Light)	4	3	3	3	2.5

		1	2	3	4	5	6
(Super)	1	6	6	8	8	10	10
(Heavy)	2	4	4	5	5	6	6
(B757)	3	4	4	4	4	4	5
(Large)	4	3	3	3	3	3	4
(Small)	5	3	3	3	3	3	3
(Light)	6	2.5	2.5	2.5	2.5	2.5	3

Table 50. Matrices of separation requirements for A-D scenarios (on the left) and scenario E (on the right).

As it was written above, we have 5 scenarios with given separation requirements, ROT values (runway occupancy times), mix of traffic and approach speeds. In order to determine final saturation capacity, we need to find average separation between couples of aircraft in the following way:

1) T_{ij} – min separation between i and j ar runway, depends wether leading aircraft with higher speed or lower (opening case and closing case accordingly) :

$$T_{ij} = max\left(\frac{n+s_{ij}}{v_j} - \frac{n}{v_i}, o_i\right)$$

$$T_{ij} = \max\left(\frac{s_{ij}}{v_j}, o_i\right)$$

2) t_{ij} - average time interval for all possible aircraft class pairs i, j with buffer time equal to 10 sec.:

$$tij = Tij + b$$

3) P_{ij} matrix (probabilities)
4) Then $E(t) = \sum_i \sum_j t_{ij}\, p_{ij}$, and Maximum Throughput Rate: **3600sec/ E(t) = *n* - aircraft**

Table 51. Scenario A.

S_{ij} =

	2	3	4
2	3	4	4
3	3	3	3
4	3	3	2.5

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
2	0.15	120	55
3	0.7	100	50
4	0.15	85	45

		2	3	4
Tij	2	90	192	268
	3	90	108	178
	4	90	108	106

		2	3	4
tij	2	100	202	278
	3	100	118	188
	4	100	118	116

		2	3	4
pij	2	0.02	0.11	0.02
	3	0.11	0.49	0.11
	4	0.02	0.11	0.02

		2	3	4
E	2	2.25	21.21	6.26
	3	10.50	57.82	19.73
	4	2.25	12.39	2.61

E =	135.02		27	arrivals		54	arrivals+departures

Table 52. Scenario B.

S_{ij} =

	1	2	3
1	4	5	6
2	3	3	4
3	3	3	3

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
1	0.1	140	60
2	0.2	120	55
3	0.7	100	50

		1	2	3
Tij	1	103	184	298
	2	77	90	192
	3	77	90	108

		1	2	3
tij	1	113	194	308
	2	87	100	202
	3	87	100	118

		1	2	3
pij	1	0.01	0.02	0.07
	2	0.02	0.04	0.14
	3	0.07	0.14	0.49

		1	2	3
E	1	1.13	3.89	21.58
	2	1.74	4.00	28.28
	3	6.10	14.00	57.82

E =	138.54		26	arrivals		52	arrivals+departures

Table 53. Scenario C.

S_{ij} =

	1	2	3	4
1	4	5	6	8
2	3	3	4	4
3	3	3	3	3
4	3	3	3	2.5

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
1	0.01	140	60
2	0.3	120	55
3	0.5	100	50
4	0.19	85	45

		1	2	3	4
Tij	1	103	184	298	472
	2	77	90	192	268
	3	77	90	108	178
	4	77	90	108	106

		1	2	3	4
tij	1	113	194	308	482
	2	87	100	202	278
	3	87	100	118	188
	4	87	100	118	116

		1	2	3	4
pij	1	0.00	0.00	0.01	0.00
	2	0.00	0.09	0.15	0.06
	3	0.01	0.15	0.25	0.10
	4	0.00	0.06	0.10	0.04

		1	2	3	4
E	1	0.01	0.58	1.54	0.92
	2	0.26	9.00	30.30	15.86
	3	0.44	15.00	29.50	17.85
	4	0.17	5.70	11.21	4.18

E =	142.52		25	arrivals	50	arrivals+departures

Table 54. Scenario D.

S_{ij} =

	1	3	4
1	4	6	8
3	3	3	3
4	3	3	2.5

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
1	0.45	135	70
3	0.05	120	60
4	0.5	100	45

		1	3	4
Tij	1	107	207	363
	3	80	90	156
	4	80	90	90

		1	3	4
tij	1	117	217	373
	3	90	100	166
	4	90	100	100

		1	3	4
pij	1	0.20	0.02	0.23
	3	0.02	0.00	0.03
	4	0.23	0.03	0.25

		1	3	4
E	1	23.63	4.88	83.85
	3	2.03	0.25	4.15
	4	20.25	2.50	25.00

E =	166.53		22	arrivals	44	arrivals+departures

Table 55. Scenario E.

$S_{ij} =$

	1	2	3	4	5	6
1	6	6	8	8	10	10
2	4	4	5	5	6	6
3	4	4	4	4	4	5
4	3	3	3	3	3	4
5	3	3	3	3	3	3
6	2.5	2.5	2.5	2.5	2.5	3

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
1	0.15	145	65
2	0.25	140	60
3	0.1	135	55
4	0.15	120	45
5	0.15	110	45
6	0.2	100	45

		1	2	3	4	5	6
Tij	1	149	161	228	281	390	449
	2	99	103	141	184	252	298
	3	99	103	107	147	179	255
	4	74	77	80	90	120	192
	5	74	77	80	90	98	134
	6	62	64	67	75	82	108

		1	2	3	4	5	6
tij	1	159	171	238	291	400	459
	2	109	113	151	194	262	308
	3	109	113	117	157	189	265
	4	84	87	90	100	130	202
	5	84	87	90	100	108	144
	6	72	74	77	85	92	118

		1	2	3	4	5	6
pij	1	0.02	0.04	0.02	0.02	0.02	0.03
	2	0.04	0.06	0.03	0.04	0.04	0.05
	3	0.02	0.03	0.01	0.02	0.02	0.02
	4	0.02	0.04	0.02	0.02	0.02	0.03
	5	0.02	0.04	0.02	0.02	0.02	0.03
	6	0.03	0.05	0.02	0.03	0.03	0.04

		1	2	3	4	5	6
E	1	3.58	6.43	3.57	6.56	9.01	13.78
	2	4.10	7.05	3.77	7.29	9.84	15.41
	3	1.64	2.82	1.17	2.35	2.84	5.29
	4	1.90	3.27	1.35	2.25	2.93	6.06
	5	1.90	3.27	1.35	2.25	2.43	4.33
	6	2.16	3.71	1.53	2.55	2.75	4.72

E =	157.22		23	arrivals	46	arrivals+departures

Table 56. Scenario E – another design with buffer times specifically calculated.

t_{max} = 30 sec.

S_{ij} =

	1	2	3	4	5	6
1	6	6	8	8	10	10
2	4	4	5	5	6	6
3	4	4	4	4	4	5
4	3	3	3	3	3	4
5	3	3	3	3	3	3
6	2.5	2.5	2.5	2.5	2.5	3

	Mix (%)	Approach speed (kts)	Runway time occupancy ROT (sec)
1	0.15	145	65
2	0.25	140	60
3	0.1	135	55
4	0.15	120	45
5	0.15	110	45
6	0.2	100	45

		1	2	3	4	5	6
bijop	1	33	25	15	-11	-49	-82
	2	33	33	25	9	-12	-32
	3	33	33	33	17	6	-17
	4	33	33	33	33	22	6
	5	33	33	33	33	33	20
	6	33	33	33	33	33	33

		1	2	3	4	5	6
Tij	1	149	161	228	281	390	449
	2	99	103	141	184	252	298
	3	99	103	107	147	179	255
	4	74	77	80	90	120	192
	5	74	77	80	90	98	134
	6	62	64	67	75	82	108

		1	2	3	4	5	6
tij	1	182	186	243	270	341	368
	2	132	136	166	193	240	267
	3	132	136	140	163	185	238
	4	107	110	113	123	142	198
	5	107	110	113	123	131	154
	6	95	97	100	108	115	141

		1	2	3	4	5	6
pij	1	0.02	0.04	0.02	0.02	0.02	0.03
	2	0.04	0.06	0.03	0.04	0.04	0.05
	3	0.02	0.03	0.01	0.02	0.02	0.02
	4	0.02	0.04	0.02	0.02	0.02	0.03
	5	0.02	0.04	0.02	0.02	0.02	0.03
	6	0.03	0.05	0.02	0.03	0.03	0.04

		1	2	3	4	5	6
E	1	4.09	6.98	3.65	6.08	7.68	11.03
	2	4.96	8.49	4.15	7.23	9.01	13.33
	3	1.98	3.40	1.40	2.45	2.78	4.76
	4	2.42	4.13	1.70	2.77	3.19	5.94
	5	2.42	4.13	1.70	2.77	2.95	4.63
	6	2.85	4.86	1.99	3.24	3.44	5.64

E =	164.23		22 arrivals		44 arrivals+departures

Now, when we got general picture for capacity of all available scenarios, there is a possibility to compare it with supply, provided in the step 3. We assume that 50% of traffic will be between 8 and 9 o'clock at morning, then all other time till 21:00 remaining demand (other 50%) will be uniformly distributed. Thus, with peak 1 hour and 11 other hours we can count capacity of our airport. ***Please, kindly notice that in supply analysis we considered only departures.*** According to supply scenarios of the previous steps, supply per peak hour will exactly equal to number of departures, since this value is 50% of daiyly operations:

number of estimated operations per peak hour for differen configurations									
	sc 1			sc 2			sc 3		
loc	p2p	p2p_ad	h&s	p2p	p2p_ad	h&s	p2p	p2p_ad	h&s
corner	13	16	17	89	75	77	107	90	96
edge	13	18	13	73	66	57	87	79	71
centre	20	20	120	64	64	536	76	76	668

Table 57. Number of operations per peak hour for the estimated demand patterns in different configurations.

Now we can observe the capacity of our airport with single runway:

time of day	sc. A	sc. B	sc. C	sc. D	sc. E	sc. E - 2
peak hour	54	52	50	44	46	44
non-peak hour	4	4	4	3	3	3
total	98	96	94	77	79	77

Table 58. Number of operations under capacity constraints with different scenarios of traffic mix.

All airport service configurations will exceed capacity constrains in the future.

Conclusion.

As we saw from supply-capacity analysis from the previous section, there is no traffic mix configuration that will meet any supply configuration, and versus. On the one hand, supply is the core value, since it is created according to the estimated demand. On the other hand, capacity constraints are physical barrier of the Parma Airport rehabilitation. Considering all those theses my final decision will be reconstruction of Parma Airport single runway to the independent parallel runways, thus I will increase possible number of movements per hour up to 99-119 mov/h. In such case I will be able to use "Point-to-Point Adapted" configuration with allocating Parma Airport at the corner hub position. The final number of gates airport will need to meet all constraints and satisfy its demand will be based on peak hour arrival rate with 90 movements or 45 arrivals per hour:

$$G = A * (T + S) = 45 \text{ arr/h} * (45 \text{ min} + 15 \text{ min}) = 45$$

Codes.

```
import pandas as pd
import googlemaps
from googlemaps import distance_matrix
import numpy as np

df = pd.read_excel('/Air_Transport/Project/Exer_1.xlsx')
df.drop(list(df.columns[9:92]), axis = 1, inplace = True)

origins = list(df.NAME_Municipality)
destinations =  list(df.columns[3:9])

my_api_key = [redacted]
gmaps = googlemaps.Client(key=my_api_key)


origins_id = []
origins_lat = []
origins_lng = []

for i in range(len(origins)):
    current_location = gmaps.geocode(origins[i])[0]
    origins_id.append('place_id:'+ current_location['place_id'])
    origins_lat.append(current_location['geometry']['location']['lat'])
    origins_lng.append(current_location['geometry']['location']['lng'])


destinations_id = []
destinations_lat = []
destinations_lng = []

for i in range(len(destinations)):
    current_location = gmaps.geocode(destinations[i])[0]
    destinations_id.append('place_id:'+ current_location['place_id'])
    destinations_lat.append(current_location['geometry']['location']['lat'])
    destinations_lng.append(current_location['geometry']['location']['lng'])
```

Code Cell # 1. Collecting location coordinates of our nodes.

```
distances_matrix = []
travel_time_matrix = []
undefined_origins = []


for origin in range(len(origins_id)):
    distances_vector = []
    travel_time_vector = []
    current_output = distance_matrix.distance_matrix(gmaps, origins_id[origin],␣
 ↪destinations_id, mode = 'driving')['rows'][0]['elements']
    try:
        for parameter in range(len(current_output)):
            distances_vector.
 ↪append(current_output[parameter]['distance']['value'])
            travel_time_vector.
 ↪append(current_output[parameter]['duration']['value'])
    except :
        for parameter in range(len(current_output)):
            distances_vector.append(0)
            travel_time_vector.append(0)
        undefined_origins.append(origins[origin])

    distances_matrix.append(distances_vector)
    travel_time_matrix.append(travel_time_vector)

if len(undefined_origins)>0:
    print(f'Undefined origins are {undefined_origins}')

distances_data = pd.DataFrame(distances_matrix, columns = destinations, index =␣
 ↪origins)
traveltime_data = pd.DataFrame(travel_time_matrix, columns = destinations,␣
 ↪index = origins)


origins_loc = pd.DataFrame(np.transpose([origins_lat, origins_lng]),
                           columns = ['Latitude', 'Longitude'], index = origins)
destinations_loc = pd.DataFrame(np.transpose([destinations_lat,␣
 ↪destinations_lng]),
                                columns = ['Latitude', 'Longitude'], index =␣
 ↪destinations)
locations = pd.concat([origins_loc, destinations_loc])
locations.to_csv('/Air_Transport/Project/locations.csv')

distances_data.to_csv('/Air_Transport/Project/distances.csv')
traveltime_data.to_csv('/Air_Transport/Project/traveltime_data.csv')
```

Code Cell # 2. Creation of the Travel Time and Distance matrixes.

```
import pandas as pd
import matplotlib.pyplot as plt

traveltime = pd.read_csv('/Air_Transport/Project/traveltime_data.csv')
traveltime.rename(columns = {'Unnamed: 0':'Nodes'}, inplace = True)
traveltime.set_index('Nodes', drop = True, inplace = True)
traveltime = traveltime /60

traveltime_copy = pd.read_csv('/Air_Transport/Project/traveltime_data.csv')
traveltime_copy.rename(columns = {'Unnamed: 0':'Nodes'}, inplace = True)
traveltime_copy.set_index('Nodes', drop = True, inplace = True)
traveltime_copy = traveltime /60

locations = pd.read_csv('/Air_Transport/Project/locations.csv')
locations.rename(columns = {'Unnamed: 0':'Nodes'}, inplace = True)
locations.set_index('Nodes', drop = True, inplace = True)

for i in (list(traveltime_copy.columns)):
    for k in traveltime_copy.index:
        if traveltime_copy[i][k]<=30.0:
            traveltime_copy[i][k] = 0
        elif traveltime_copy[i][k] <=60.0:
            traveltime_copy[i][k] = 1
        elif traveltime_copy[i][k] <= 90.0:
            traveltime_copy[i][k] = 2
        elif traveltime_copy[i][k] <= 120.0:
            traveltime_copy[i][k] = 3
        else:
            traveltime_copy[i][k] = 4

traveltime_copy.to_csv('/Air_Transport/Project/Time_Thresholds.csv')


origins = locations.iloc[:44]
destinations = locations.iloc[44:]


origins_long = origins.loc[:, 'Longitude']
origins_lat = origins.loc[:, 'Latitude']
destinations_long = destinations.loc[:, 'Longitude']
destinations_lat = destinations.loc[:, 'Latitude']


pics = ['Parma', 'Milan', 'Verona', 'Venice', 'Bologna', 'Pisa']
boundaries = {'Parma': [9.3988, 11.2665, 44.2524, 45.1640],
              'Milan': [8.337, 12.072, 44.108, 45.921],
              'Verona': [8.549, 12.285, 43.658, 45.701],
              'Venice': [9.300, 13.035, 43.917, 45.951],
              'Bologna': [9.4977, 11.3654, 44.1486, 45.1703],
              'Pisa': [8.465, 12.200, 43.254, 45.311]}

linewidth = 5.0
```

Code Cell # 3. Time Thresholds identification and Isochrone maps plotting, part A.

```python
destinations_long = destinations.loc[:, 'Longitude']
destinations_lat = destinations.loc[:, 'Latitude']


pics = ['Parma', 'Milan', 'Verona', 'Venice', 'Bologna', 'Pisa']
boundaries = {'Parma': [9.3988, 11.2665, 44.2524, 45.1640],
              'Milan': [8.337, 12.072, 44.108, 45.921],
              'Verona': [8.549, 12.285, 43.658, 45.701],
              'Venice': [9.300, 13.035, 43.917, 45.951],
              'Bologna': [9.4977, 11.3654, 44.1486, 45.1703],
              'Pisa': [8.465, 12.200, 43.254, 45.311]}

linewidth = 5.0

for destination in range(len(destinations)):

    Italia = plt.imread(f'/Air_Transport/Project/raw_maps/{pics[destination]}.
 ↪png')

    BBox = boundaries[pics[destination]]
    fig, ax = plt.subplots(figsize = (36,24))



    ax.scatter(origins_long, origins_lat, s = 200, marker='o', color = 'black')
    ax.scatter(destinations_long[destination], destinations_lat[destination], s␣
 ↪= 800, marker='v', color = 'red')

    for origin in range(len(origins)):

        x = [origins_long[origin], destinations_long[destination]]
        y = [origins_lat[origin], destinations_lat[destination]]


        if (traveltime.iloc[origin, destination]) <= 30.0:
            ax.plot(x, y, '-.', color = 'green', linewidth = linewidth)
        elif (traveltime.iloc[origin, destination]) <= 60.0:
            ax.plot(x, y, '-.', color = 'blue', linewidth = linewidth)
        elif (traveltime.iloc[origin, destination]) <= 90.0:
            ax.plot(x, y, '-.', color = 'yellow', linewidth = linewidth)
        elif (traveltime.iloc[origin, destination]) <= 120.0:
            ax.plot(x, y, '-.', color = 'black', linewidth = linewidth)
        else:
            ax.plot(x, y, '-.', color = 'red', linewidth = linewidth)



    ax.set_title(f'Accessibility of {pics[destination]} Airport from Parma␣
 ↪region')

    ax.set_xlim(BBox[0],BBox[1])
    ax.set_ylim(BBox[2],BBox[3])

    ax.imshow(Italia, zorder=0, extent = BBox, aspect= 'auto')

    plt.savefig(f'/Air_Transport/Project/accessibility_maps/
 ↪Accessibility_of_{pics[destination]}_Airport.jpg', bbox_inches='tight')
```

Code Cell # 4. Time Thresholds identification and Isochrone maps plotting, part B.

Part a) Sources: http://demo.istat.it/popres/index.php?anno=2020&lingua=ita

Notes:

- Sissa and Tresacali united in one comune Sissa tresacali
- Polesine Parmense and Zibello in one comune Polesine Zibello
- Sorbolo and Mezzani in one comune Sorbolo Mezzani

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
data_26_65_2011 = {}
for i in range(0, 47):
    key = pd.read_csv(f'/Air_Transport/Project/demografics/2011/26_65/
 ↪tavola_pop_res ({i}).csv', header = 1, nrows=1).columns[0][8:]
    value = pd.read_csv(f'/Air_Transport/Project/demografics/2011/26_65/
 ↪tavola_pop_res ({i}).csv', header = 2).iloc[-1, :]['Maschi+Femmine']
    data_26_65_2011.update({key:value})

data_66_over_2011 = {}
for i in range(0, 47):
    key = pd.read_csv(f'/Air_Transport/Project/demografics/2011/65_over/
 ↪tavola_pop_res ({i}).csv', header = 1, nrows=1).columns[0][8:]
    value = pd.read_csv(f'/Air_Transport/Project/demografics/2011/65_over/
 ↪tavola_pop_res ({i}).csv', header = 2).iloc[-1, :]['Maschi+Femmine']
    eta_65 = pd.read_csv(f'/Air_Transport/Project/demografics/2011/65_over/
 ↪tavola_pop_res ({i}).csv', header = 2).iloc[0, :]['Maschi+Femmine']
    value = value - eta_65
    data_66_over_2011.update({key:value})


totale = pd.read_csv('/Air_Transport/Project/demografics/2011/total_2011.csv',␣
 ↪header=2)
comuni = totale.loc[:, 'Comuni']
values = totale.loc[:, 'Maschi+Femmine']

data_total_2011 = {}

for i in range(len(comuni)):
    data_total_2011.update({comuni[i]:values[i]})

data_total_2011.pop('TOTALE')


#################################################################################


data_26_65_2020 = {}

for i in range(1, 45):
    key = pd.read_csv(f'/Air_Transport/Project/demografics/2020/26_65/{i}.csv',␣
 ↪nrows = 1).iloc[0,0][8:]
    value = pd.read_csv(f'/Air_Transport/Project/demografics/2020/26_65/{i}.
 ↪csv', sep=';', header = 2).iloc[-2, 3]
    data_26_65_2020.update({key:value})
```

Code Cell # 5. Population data acquisition and analysis, part A.

```
data_66_over_2020 = {}

for i in range(1, 45):
    key = pd.read_csv(f'/Air_Transport/Project/demografics/2020/66_over/{i}.
 ↪csv', nrows = 1).iloc[0,0][8:]
    value = pd.read_csv(f'/Air_Transport/Project/demografics/2020/66_over/{i}.
 ↪csv', sep=';', header = 2).iloc[-2, 3]
    data_66_over_2020.update({key:value})

totale = pd.read_csv('/Air_Transport/Project/demografics/2020/total_2020.csv',␣
 ↪sep = ';', header=2)
totale.drop([44, 45], inplace = True)

comuni = totale.loc[:, 'Comune']
values = totale.loc[:, 'Maschi + Femmine']


data_total_2020 = {}

for i in range(len(comuni)):
    data_total_2020.update({comuni[i]:values[i]})


munic_2020 = list(data_total_2020.keys())
munic_2011 = list(data_total_2011.keys())


print('These municipalities were separated in 2011:')

munic_2011_separate = []
for i in range(len(munic_2011)):
    if munic_2011[i] not in munic_2020:
        munic_2011_separate.append(munic_2011[i])
        print(munic_2011[i])

print('')
print('')
print('Then they are united in these municipalities:')

for i in range(len(munic_2020)):
    if munic_2020[i] not in munic_2011:
        print(munic_2020[i])



data_26_65_2011.update(    {'Sissa Trecasali' : (data_26_65_2011['Sissa'] +␣
 ↪data_26_65_2011['Trecasali'])    }    )
data_66_over_2011.update( {'Sissa Trecasali' : (data_66_over_2011['Sissa'] +␣
 ↪data_66_over_2011['Trecasali'])    }    )
data_total_2011.update(    {'Sissa Trecasali' : (data_total_2011['Sissa'] +␣
 ↪data_total_2011['Trecasali'])    }    )
```

Code Cell # 6. Population data acquisition and analysis, part B.

```
data_26_65_2011.update(   {'Polesine Zibello' : (data_26_65_2011['Polesine␣
↪Parmense'] + data_26_65_2011['Zibello'])   }   )
data_66_over_2011.update( {'Polesine Zibello' : (data_66_over_2011['Polesine␣
↪Parmense'] + data_66_over_2011['Zibello'])   }   )
data_total_2011.update(   {'Polesine Zibello' : (data_total_2011['Polesine␣
↪Parmense'] + data_total_2011['Zibello'])   }   )

data_26_65_2011.update(   {'Sorbolo Mezzani' : (data_26_65_2011['Sorbolo'] +␣
↪data_26_65_2011['Mezzani'])   }   )
data_66_over_2011.update( {'Sorbolo Mezzani' : (data_66_over_2011['Sorbolo'] +␣
↪data_66_over_2011['Mezzani'])   }   )
data_total_2011.update(   {'Sorbolo Mezzani' : (data_total_2011['Sorbolo'] +␣
↪data_total_2011['Mezzani'])   }   )

for i in range(len(munic_2011_separate)):
    data_26_65_2011.pop(munic_2011_separate[i])
    data_66_over_2011.pop(munic_2011_separate[i])
    data_total_2011.pop(munic_2011_separate[i])

final_data_matrix = {'popul_26_65_2011':data_26_65_2011, 'popul_over_65_2011':
↪data_66_over_2011,
                     'total_2011':data_total_2011, 'popul_26_65_2020':
↪data_26_65_2020,
                     'popul_over_65_2020':data_66_over_2020, 'total_2020':
↪data_total_2020}

final_data = pd.DataFrame(final_data_matrix)

final_data['difference_26_65_2011_2020'] = final_data.popul_26_65_2020 -␣
↪final_data.popul_26_65_2011
final_data['variation_26_65_2011_2020'] = (final_data.
↪difference_26_65_2011_2020/final_data.popul_26_65_2011)*100
final_data['proportion_26_65_2011'] = (final_data.popul_26_65_2011/final_data.
↪total_2011)*100
final_data['proportion_26_65_2020'] = (final_data.popul_26_65_2020/final_data.
↪total_2020)*100

final_data['difference_over_65_2011_2020'] = final_data.popul_over_65_2020 -␣
↪final_data.popul_over_65_2011
final_data['variation_over_65_2011_2020'] = (final_data.
↪difference_over_65_2011_2020/final_data.popul_over_65_2011)*100
final_data['proportion_over_65_2011'] = (final_data.popul_over_65_2011/
↪final_data.total_2011)*100
final_data['proportion_over_65_2020'] = (final_data.popul_over_65_2020/
↪final_data.total_2020)*100

final_data['propor_variation_26_65_2011_2020'] = final_data.
↪proportion_26_65_2020 - final_data.proportion_26_65_2011
final_data['propor_variation_over_65_2011_2020'] = final_data.
↪proportion_over_65_2020 - final_data.proportion_over_65_2011

final_data['total_difference_2011_2020'] = final_data.total_2020 - final_data.
↪total_2011
final_data['total_variation_2011_2020'] = (final_data.
↪total_difference_2011_2020 / final_data.total_2011)*100
```

Code Cell # 7. Population data acquisition and analysis, part C.

```
var_26_65_lower_limit = final_data.describe(percentiles=[0.25, 0.5, 0.75]).
 →loc['25%', :]['variation_26_65_2011_2020']
var_26_65_upper_limit = final_data.describe(percentiles=[0.25, 0.5, 0.75]).
 →loc['75%', :]['variation_26_65_2011_2020']

prop_var_26_65_lower_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
 →loc['20%', :]['propor_variation_26_65_2011_2020']
prop_var_26_65_upper_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
 →loc['80%', :]['propor_variation_26_65_2011_2020']

prop_var_over_65_upper_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
 →loc['80%', :]['propor_variation_over_65_2011_2020']

print('Population in 26-65 age range variation lower limit is ',␣
 →var_26_65_lower_limit)
print(' ')
print('Population in 26-65 age range variation upper limit is ',␣
 →var_26_65_upper_limit)
print(' ')
print('Population in 26-65 age range proportion variation lower limit is ',␣
 →prop_var_26_65_lower_limit)
print(' ')
print('Population in 26-65 age range proportion variation upper limit is ',␣
 →prop_var_26_65_upper_limit)
print(' ')
print('Population in over 65 age range proportion variation upper limit is ',␣
 →prop_var_over_65_upper_limit)
print(' ')
```

```
Population in 26-65 age range variation lower limit is  -9.861742463327374

Population in 26-65 age range variation upper limit is  -1.4316538647023054

Population in 26-65 age range proportion variation lower limit is
-2.74947298695096

Population in 26-65 age range proportion variation upper limit is
-0.8206508474059405

Population in over 65 age range proportion variation upper limit is
2.585204008835539
```

Code Cell # 8. Threshold limits determination for population indicators.

```python
path = '/Air_Transport/Project/demografics/separated_by_municipalities'


for year in range(2011, 2020):

    os.mkdir(path + f'/{year}')

    for munic in range(1, 100):

        url = f"http://demo.istat.it/pop{year}/popol.php?
→m1=&m2=&m3=&m4=&m5=y&f1=&f2=&f3=&f4=&f5=y&daanno=0&adanno=100&lingua=ita&Rip=S2

        r = requests.get(url, allow_redirects=True)

        open(path + f'/{year}' + f'/munic_{munic}.csv', 'wb').write(r.content)
```

Code Cell # 9. Data scrapping with python.

```python
population_prediction = {}

for year in range(2011, 2018):

    current_year_popul_26_65 = {}
    current_year_popul_over_65 = {}

    for munic in range(1, 100):
        try:
            current_file = f'/Air_Transport/Project/demografics/
→separated_by_municipalities/{year}/munic_{munic}.csv'
            comune = pd.read_csv(current_file, skiprows=1, nrows = 0).
→columns[0][8:]

            df = pd.read_csv(current_file, skiprows = 2)
            df.drop('Unnamed: 4', axis = 1, inplace = True)

            current_year_popul_26_65.update({comune:sum(df.iloc[26:66, 3])})
            current_year_popul_over_65.update({comune:sum(df.iloc[66:-1, 3])})
        except:
            continue


    key_1 = 'age_26_65' + f'_{year}'
    key_2 = 'age_over_65' + f'_{year}'

    population_prediction.update({key_1:current_year_popul_26_65, key_2:
→current_year_popul_over_65})
```

Code Cell # 10. Population prediction with Linear Regression Model, Part A).

```
for year in range(2018, 2020):

    current_year_popul_26_65 = {}
    current_year_popul_over_65 = {}

    for munic in range(1, 100):
        try:
            current_file = f'/Air_Transport/Project/demografics/
 ↪separated_by_municipalities/{year}/munic_{munic}.csv'
            comune = pd.read_csv(current_file, skiprows=1, nrows = 0).
 ↪columns[0][8:]

            df = pd.read_csv(current_file, skiprows = 2)
            df.drop('Unnamed: 10', axis = 1, inplace = True)

            current_year_popul_26_65.update({comune:sum(df.iloc[26:66, 9])})
            current_year_popul_over_65.update({comune:sum(df.iloc[66:-1, 9])})
        except:
            continue


    key_1 = 'age_26_65' + f'_{year}'
    key_2 = 'age_over_65' + f'_{year}'

    population_prediction.update({key_1:current_year_popul_26_65, key_2:
 ↪current_year_popul_over_65})


for key in list(population_prediction.keys()):
    if 'Sissa' and 'Trecasali' in population_prediction[key]:
        population_prediction[key]['Sissa Trecasali'] =␣
 ↪population_prediction[key]['Sissa'] + population_prediction[key]['Trecasali']
        population_prediction[key].pop('Sissa')
        population_prediction[key].pop('Trecasali')
    if 'Polesine Parmense' and 'Zibello' in population_prediction[key]:
        population_prediction[key]['Polesine Zibello'] =␣
 ↪population_prediction[key]['Polesine Parmense'] +␣
 ↪population_prediction[key]['Zibello']
        population_prediction[key].pop('Polesine Parmense')
        population_prediction[key].pop('Zibello')
    if 'Sorbolo' and 'Mezzani' in population_prediction[key]:
        population_prediction[key]['Sorbolo Mezzani'] =␣
 ↪population_prediction[key]['Sorbolo'] + population_prediction[key]['Mezzani']
        population_prediction[key].pop('Sorbolo')
        population_prediction[key].pop('Mezzani')

population_prediction = pd.DataFrame(population_prediction)
```

Code Cell # 11. Population prediction with Linear Regression Model, Part B).

```
age_26_65_2021 = []
age_over_65_2021 = []
age_26_65_2031 = []
age_over_65_2031 = []


for munic in list(population_prediction.index):

    age_26_65 = []

    for year in range(2011, 2020):
        age_26_65.append(population_prediction.loc[munic, f'age_26_65_{year}'])

    lm = LinearRegression()
    X = np.array(range(2011, 2020))
    X = X.reshape(len(X), 1)
    Y = age_26_65
    lm.fit(X, Y)


    X = np.array(range(2011, 2032))
    X = X.reshape(len(X), 1)
    yhat_2021 = lm.predict(X)[10]
    yhat_2031 = lm.predict(X)[-1]

    age_26_65_2021.append(round(yhat_2021))
    age_26_65_2031.append(round(yhat_2031))

    #########################################


    age_over_65 = []

    for year in range(2011, 2020):
        age_over_65.append(population_prediction.loc[munic,␣
 ↪f'age_over_65_{year}'])

    X = np.array(range(2011, 2020))
    X = X.reshape(len(X), 1)
    Y = age_over_65
    lm.fit(X, Y)


    X = np.array(range(2011, 2032))
    X = X.reshape(len(X), 1)

    yhat_2021 = lm.predict(X)[10]
    yhat_2031 = lm.predict(X)[-1]

    age_over_65_2021.append(round(yhat_2021))
    age_over_65_2031.append(round(yhat_2031))


population_prediction['age_26_65_2021'] = age_26_65_2021
population_prediction['age_over_65_2021'] = age_over_65_2021
population_prediction['age_26_65_2031'] = age_26_65_2031
population_prediction['age_over_65_2031'] = age_over_65_2031

population_prediction.head()
population_prediction.to_csv('/Air_Transport/Project/population_prediction.csv')
```

Code Cell # 12. Population prediction with Linear Regression Model, Part C).

```python
import pandas as pd


population_prediction = pd.read_csv('/Air_Transport/Project/
↪population_prediction.csv')
final_data = pd.read_csv('/Air_Transport/Project/final_data.csv')

var_26_65_lower_limit = final_data.describe(percentiles=[0.25, 0.5, 0.75]).
↪loc['25%', :]['variation_26_65_2011_2020']
var_26_65_upper_limit = 0

prop_var_26_65_lower_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
↪loc['20%', :]['propor_variation_26_65_2011_2020']
prop_var_26_65_upper_limit = 0

prop_var_over_65_upper_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
↪loc['80%', :]['propor_variation_over_65_2011_2020']
prop_var_over_65_lower_limit = final_data.describe(percentiles=[0.2, 0.5, 0.8]).
↪loc['20%', :]['propor_variation_over_65_2011_2020']


print('Population in 26-65 age range variation lower limit is ',␣
↪var_26_65_lower_limit)
print(' ')
print('Population in 26-65 age range variation upper limit is ',␣
↪var_26_65_upper_limit)
print(' ')
print('Population in 26-65 age range proportion variation lower limit is ',␣
↪prop_var_26_65_lower_limit)
print(' ')
print('Population in 26-65 age range proportion variation upper limit is ',␣
↪prop_var_26_65_upper_limit)
print(' ')
print('Population in over 65 age range proportion variation upper limit is ',␣
↪prop_var_over_65_upper_limit)
print(' ')
print('Population in over 65 age range proportion variation lower limit is ',␣
↪prop_var_over_65_lower_limit)
```

Code Cell # 13. Demand generation in three scenarios. First step – thresholds identification.

```
adj_coef = []

d_1_scenario = []
d_2_work_scenario = []
d_2_non_work_scenario = []
d_3_work_scenario = []
d_3_non_work_scenario = []

rule_of_thumb_applied_1 = []
rule_of_thumb_applied_2 = []
rule_of_thumb_applied_3 = []



for i in range(len(population_prediction.age_26_65_2021)):

    if (final_data.variation_26_65_2011_2020[i] < var_26_65_lower_limit) and␣
 ↪((final_data.propor_variation_26_65_2011_2020[i] >␣
 ↪prop_var_26_65_lower_limit) and (final_data.
 ↪propor_variation_26_65_2011_2020[i] < prop_var_26_65_upper_limit)):
        Tp = 0.8
        adj_coef.append(Tp)

    elif ((final_data.variation_26_65_2011_2020[i] < var_26_65_lower_limit) and␣
 ↪(final_data.propor_variation_26_65_2011_2020[i] <␣
 ↪prop_var_26_65_lower_limit))  or ((final_data.variation_26_65_2011_2020[i] <␣
 ↪var_26_65_lower_limit) and (final_data.propor_variation_over_65_2011_2020[i]␣
 ↪> prop_var_over_65_upper_limit)):
        Tp = 0.9
        adj_coef.append(Tp)

    elif (final_data.variation_26_65_2011_2020[i] > var_26_65_upper_limit) and␣
 ↪((final_data.propor_variation_26_65_2011_2020[i] >␣
 ↪prop_var_26_65_lower_limit) and (final_data.
 ↪propor_variation_26_65_2011_2020[i] < prop_var_26_65_upper_limit)):
        Tp = 1.1
        adj_coef.append(Tp)

    elif ((final_data.variation_26_65_2011_2020[i] > var_26_65_upper_limit) and␣
 ↪(final_data.propor_variation_26_65_2011_2020[i] >␣
 ↪prop_var_26_65_upper_limit))  or ((final_data.variation_26_65_2011_2020[i] >␣
 ↪var_26_65_upper_limit) and (final_data.propor_variation_over_65_2011_2020[i]␣
 ↪< prop_var_over_65_lower_limit)):
        Tp = 1.2
        adj_coef.append(Tp)
    else:
        Tp = 1
        adj_coef.append(Tp)

        # We assume that Rule of Thumb is uniformly distributed on all␣
 ↪municipalities of the region
```

Code Cell # 14. Demand generation in three scenarios. Second step – demand generation process, part A).

```
    rule_of_thumb_2021 = 0.04 * (population_prediction.age_26_65_2021[i] +␣
 ↪population_prediction.age_over_65_2021[i])

    d_1 = (population_prediction.age_26_65_2021[i] + population_prediction.
 ↪age_over_65_2021[i])*1.57*0.24*0.68*0.15*0.4*Tp

    if d_1 < rule_of_thumb_2021:

        d_1 = rule_of_thumb_2021
        rule_of_thumb_applied_1.append('Yes')
    else:
        rule_of_thumb_applied_1.append('No')


    d_2_work = (population_prediction.age_26_65_2021[i] + population_prediction.
 ↪age_over_65_2021[i])*1.57*0.24*0.68*0.15*0.7*Tp
    d_2_non_work = (population_prediction.age_26_65_2021[i] +␣
 ↪population_prediction.age_over_65_2021[i])*1.57*0.24*0.68*0.85*0.7*Tp

    if d_2_work + d_2_non_work < rule_of_thumb_2021:

        d_2_work = rule_of_thumb_2021*0.15
        d_2_non_work = rule_of_thumb_2021*0.85
        rule_of_thumb_applied_2.append('Yes')
    else:
        rule_of_thumb_applied_2.append('No')


    rule_of_thumb_2031 = 0.04 * (population_prediction.age_26_65_2031[i] +␣
 ↪population_prediction.age_over_65_2031[i])

    d_3_work = (population_prediction.age_26_65_2031[i] + population_prediction.
 ↪age_over_65_2031[i])*1.57*0.24*0.68*0.15*Tp
    d_3_non_work = (population_prediction.age_26_65_2031[i] +␣
 ↪population_prediction.age_over_65_2031[i])*1.57*0.24*0.68*0.85*Tp


    if d_3_work + d_3_non_work < rule_of_thumb_2031:

        d_3_work = rule_of_thumb_2031*0.15
        d_3_non_work = rule_of_thumb_2031*0.85
        rule_of_thumb_applied_3.append('Yes')
    else:
        rule_of_thumb_applied_3.append('No')

    d_1_scenario.append(d_1)

    d_2_non_work_scenario.append(d_2_non_work)
    d_2_work_scenario.append(d_2_work)

    d_3_work_scenario.append(d_3_work)
    d_3_non_work_scenario.append(d_3_non_work)



demand_generation_data = {'adj_coef':adj_coef, 'd_1_scenario':d_1_scenario,␣
 ↪'rule_of_thumb_applied_1':rule_of_thumb_applied_1,
                          'd_2_work_scenario':d_2_work_scenario,␣
 ↪'d_2_non_work_scenario' : d_2_non_work_scenario,
                          'rule_of_thumb_applied_2':rule_of_thumb_applied_2,␣
 ↪'d_3_work_scenario':d_3_work_scenario,
                          'd_3_non_work_scenario':d_3_non_work_scenario,␣
 ↪'rule_of_thumb_applied_3':rule_of_thumb_applied_3,}

demand_generation_data = pd.DataFrame(demand_generation_data, index =␣
 ↪population_prediction['Unnamed: 0'])
demand_generation_data.to_csv('/Air_Transport/Project/demand_generation_data.
 ↪csv')
```

Code Cell # 15. Demand generation in three scenarios. Second step – demand generation process, part B).

```python
import pandas as pd
import numpy as np

attr_data = pd.read_excel('/Air_Transport/Project/data_aeroporti_2019_12.xls',␣
↪sheet_name='Movimenti', header=1)



attr_milan = attr_data.iloc[20, 4]
attr_parma = attr_data.iloc[24, 4]
attr_bologna = attr_data.iloc[4, 4]
attr_venice = attr_data.iloc[37, 4]
attr_pisa = attr_data.iloc[27, 4]
attr_verona = attr_data.iloc[38, 4]



attr_d = {'parma':attr_parma, 'milan':attr_milan, 'verona':attr_verona,
          'venice':attr_venice, 'bologna':attr_bologna, 'pisa':attr_pisa}




traveltime_data = pd.read_csv('/Air_Transport/Project/traveltime_data.csv',␣
↪index_col=0)
traveltime_data.columns = ['parma', 'milan', 'verona', 'venice', 'bologna',␣
↪'pisa']
traveltime_data = traveltime_data/60/60



prob_matrix = []

betas = [0.01, 0.14, 0.12, 0.14, 0.1, 0.12]

prob_total = {'prob_2021_working':0, 'prob_2021_non_working':0,␣
↪'prob_2031_working':0, 'prob_2031_non_working':0}
time_values = [18.76, 4.46, 29.078, 5.5304]


for time_value in range(len(prob_total)):

    prob_matrix = []

    for munic in range(len(traveltime_data.index)):

        imped_func = []

        for i in range(len(traveltime_data.columns)):
            imped_func.append(np.exp(-traveltime_data.iloc[munic,␣
↪i]*time_values[time_value]*betas[i]))

        x = []

        for i in range(len(imped_func)):
            x.append(list(attr_d.values())[i] * imped_func[i])

        y = sum(x)

        prob = []

        for i in range(len(x)):
            prob.append(round(x[i]/y, 5))

        prob_matrix.append(prob)

    prob_total.update({list(prob_total.keys())[time_value]:prob_matrix})
```

Code Cell # 16. Demand split on destinations by Gravity model.