Using Big Data to Connect the Dots from One Place to Another

Tidewater Big Data Enthusiasts
Chuck Cartledge
Developer

June 28, 2016 at 4:06pm

Contents

List of Figures **ii**

List of Tables **iii**

1 Introduction **1**
1.1 A little bit about graphs 1
1.2 Neo4j 1
1.3 Comparison of database technologies 9

2 Approach **9**
2.1 Neo4j database installation 9
2.2 Starting neo4j 10
2.3 Interacting with the database 10

3 Results **17**
3.1 EYW-YXY (Key West Intl to Whitehorse Intl) 17
3.2 ITO-KEF (Hilo Intl to Keflavik International Airport) 17
3.3 ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) 17
3.4 ORF-GME (Gomel to Gomel) 21
3.5 ORF-RGL (Norfolk Intl to Rio Gallegos) 21
3.6 ORF-THU (Norfolk Intl to Thule Air Base) 21
3.7 YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) 21
3.8 YXY-RGL (Whitehorse Intl to Rio Gallegos) 27

4 Conclusion **29**

A Neo4j Lessons Learned **30**

B Misc. files **32**

C Miscellaneous maps **33**

D References **40**

List of Figures

1 Map of Kaliningrad, Russia. 2
2 The seven bridges of Königsberg. 3
3 A simple graph with three nodes. 4
4 A graph with three nodes and three edges. 5
5 A directed graph. 6
6 An airline route on the East Coast. 7
7 Old Dominion University (ODU) Computer Science undergraudate prerequisistes. 8
8 Starting neo4j in a terminal window. 11
9 Neo4j browser homepage. 13
10 Neo4j browser database information page. 14
11 Neo4j shell starting prompt. 15
12 Neo4j shell command line arguments. 16
13 EYW-YXY (Key West Intl to Whitehorse Intl) flight path. 18
14 ITO-KEF (Hilo Intl to Keflavik International Airport) flight path. 19
15 ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) flight path. 20
16 ORF-GME (Gomel to Gomel) flight path. 22
17 ORF-RGL (Norfolk Intl to Rio Gallegos) flight path. 24
18 ORF-THU (Norfolk Intl to Thule Air Base) flight path. 25
19 YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) flight path. . . . 26
20 YXY-RGL (Whitehorse Intl to Rio Gallegos) flight path. 28
21 Computer system measurements before and after loading IMDb. 31
22 Serviced and non-serviced airports in and around Norfolk, Virginia. 34
23 Serviced and non-serviced airports in and around Paris, France. 35
24 Serviced and non-serviced airports in and around Seattle, Washington. . . . 36
25 Serviced and non-serviced airports in and around the United States. 37
26 Serviced and non-serviced airports in and around Virginia. 38
27 Serviced and non-serviced airports in and around the World. 39

List of Tables

1 A comparison of different database terms and ideas. 10
2 EYW-YXY (Key West Intl to Whitehorse Intl) airports along the flight path. 17
3 ITO-KEF (Hilo Intl to Keflavik International Airport) airports along the flight path. 21
4 ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) airports along the flight path. 21
5 ORF-GME (Gomel to Gomel) airports along the flight path. 21
6 ORF-RGL (Norfolk Intl to Rio Gallegos) airports along the flight path. . . . 23
7 ORF-THU (Norfolk Intl to Thule Air Base) airports along the flight path. . 23
8 YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) airports along the flight path. 23
9 YXY-RGL (Whitehorse Intl to Rio Gallegos) airports along the flight path. . 27
10 Time spent loading the IMDb via the neo4j-shell. 30

1 Introduction

1.1 A little bit about graphs

In the early 1730s, the mayor and people of Königsberg, Germany (Kaliningrad, Russia (see Figure 1)) had a problem. The people of the city would spend Sunday afternoons walking, trying to cross each of the city's seven bridges exactly one (see Figure 2)[8].

The mayor appealed to Leonhard Euler[7] to solve the problem. Euler initially declined to work on the problem, because he thought it was trivial. Later Euler changed his mind, and in 1736 publishes a general solution to the problem. Euler's paper is considered to the first paper on graph theory[1]. One of the many things that is interesting about Euler's paper, was that it never mentioned graphs, nodes, vertices, or any of the other terms now associated with graph theory[2].

A graph in world of graph theory is not like a graph in the world of Excel. In the graph theory world, a graph is composed of objects called nodes (or vertices), that are connected (or not) by things called arcs (or edges) (see Figure 3). The nodes or edges can be named, and edges can "circle back" the node the same node (see Figure 4). Edges can have direction to show that there is "movement" in only one direction along the edge (see Figure 4). The ideas of naming nodes, edges having directionality, naming arcs can be used to show how airports are connected to each other using different airlines. For instance, you can fly from Norfolk, VA (ORF) to Baltimore Washington International (BWI) via Frontier Air (FL), from there onto John F. Kennedy International (JFK) via American Airlines (AA), and return to ORF via AA (see Figure 6).

Another example of an interesting directed and disconnected graph is Old Dominion Univeristy Computer Science undergraduate prequisite course relationship (see Figure 7).

We will be using the airport and airline routes from OpenFlights.org as our data for a graph database.

1.2 Neo4j

> *"Neo4j is a graph database management system developed by Neo Technology, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing, Neo4j is the most popular graph database according to db-engines.com.*
> *Neo4j is available in a GPL3-licensed open-source "community edition", with online backup and high availability extensions licensed under the terms of the Affero General Public License. Neo also licenses Neo4j with these extensions under closed-source commercial terms.*
> *Neo4j is implemented in Java and accessible from software written in other languages using the Cypher Query Language through a transactional HTTP endpoint. Version 1.0 was released in February, 2010. Neo4j version 2.0 was released*

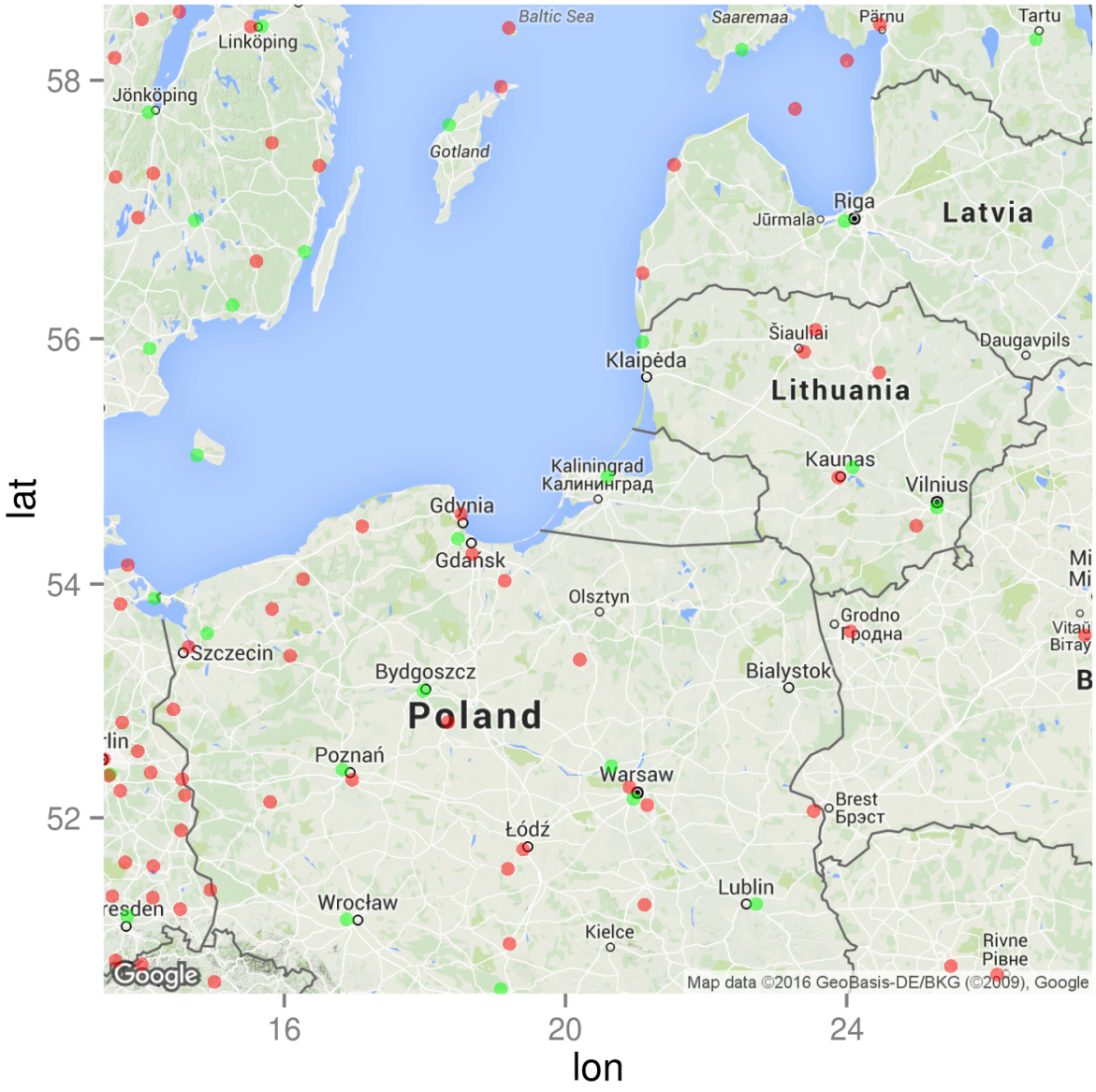


Figure 1: Map of Kaliningrad, Russia. Map created using ggmap and R[3, 4].

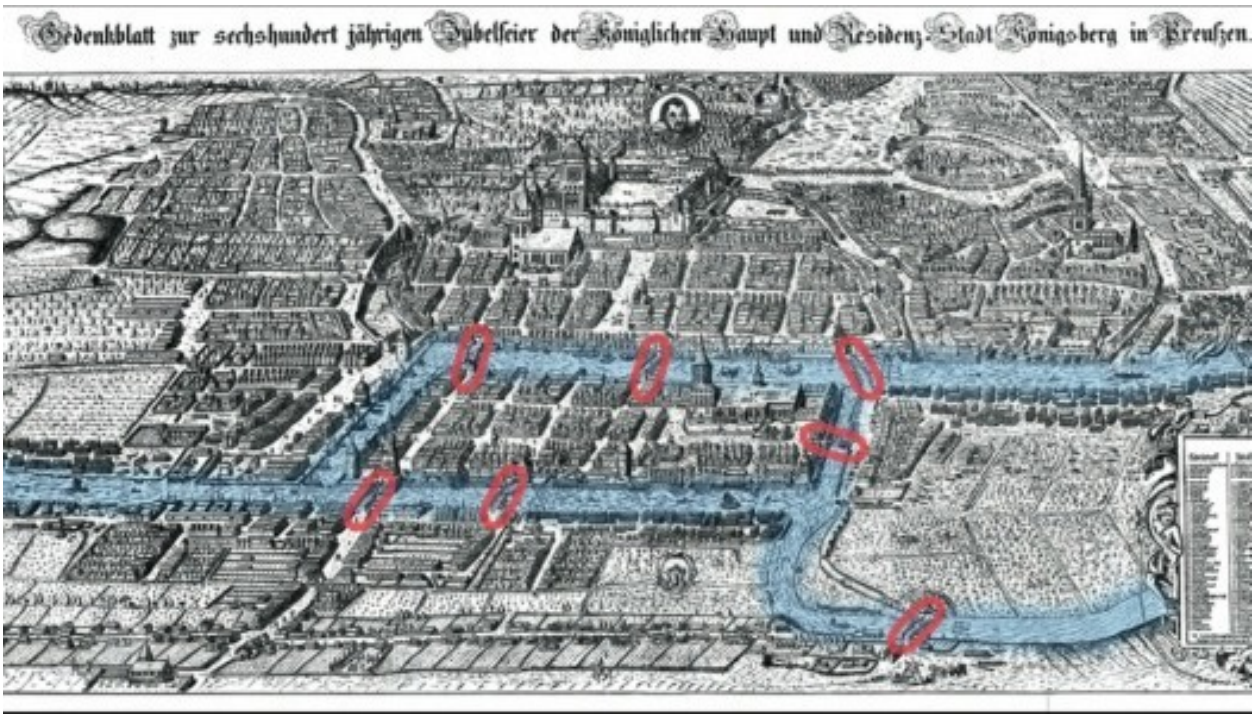


Figure 2: The seven bridges of Königsberg. The city as it was in the early 1730s.

in December, 2013.“

Wikipedia Staff [6]

Picking apart the above quote, we have:

1. Neo4j is ACID compliant. Meaning:

   A: *Atomicity* requires each database transaction be fully completed, or not completed at all. Atomicity does not permit partial completion of any transaction.

   C: *Consistency* requires that any transaction will take the database from whatever the previous state was to the next state, in its entirety. The database will not be left in an inconsistent state.

   I: *Isolation* requires that the effect of any transaction be the same if the transactions were executed in parallel, or serially.

   D: *Durability* requires that the effect of any transaction will remain if the database program were terminated immediately after a transaction (i.e., all database transactions are recorded to disk before they are reported as complete).

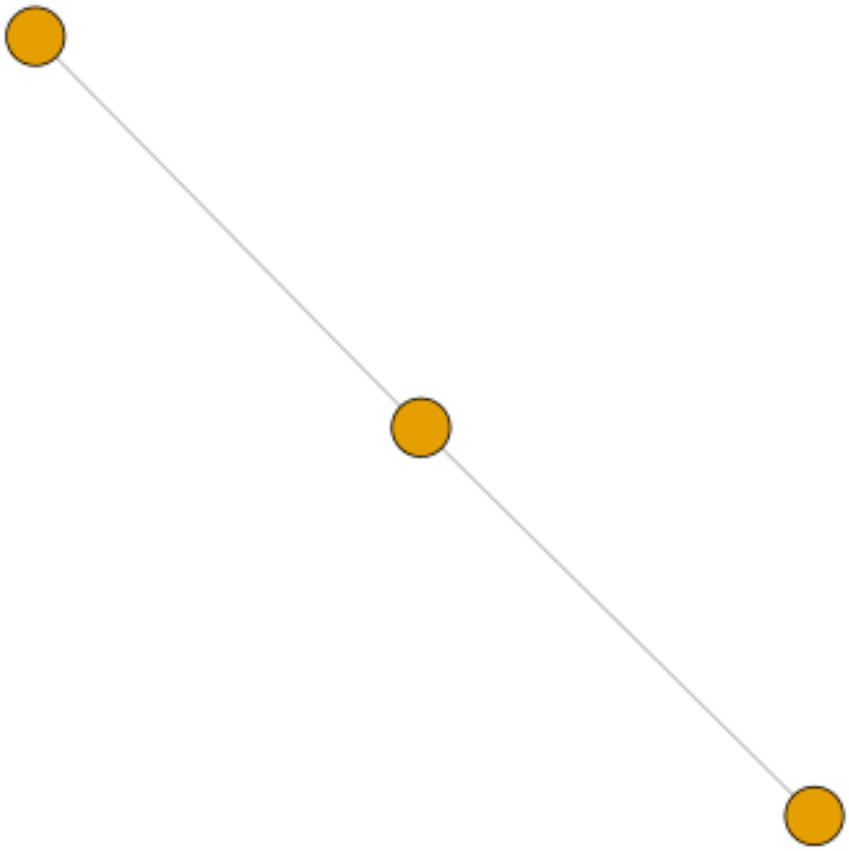


Figure 3: A simple graph with three nodes.

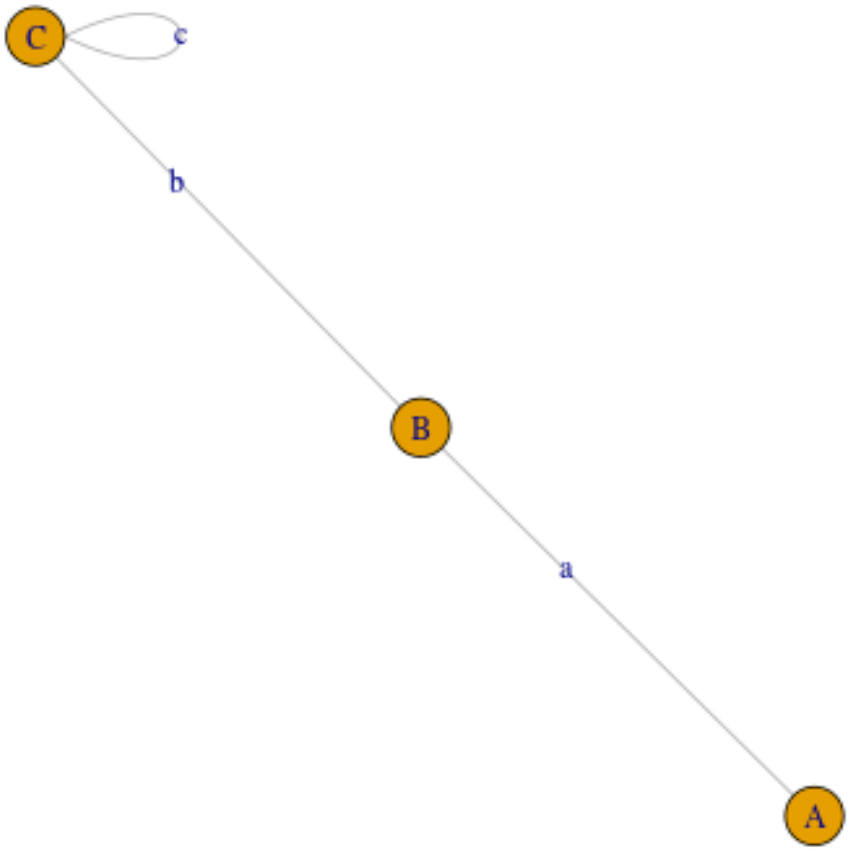


Figure 4: A graph with three nodes and three edges.

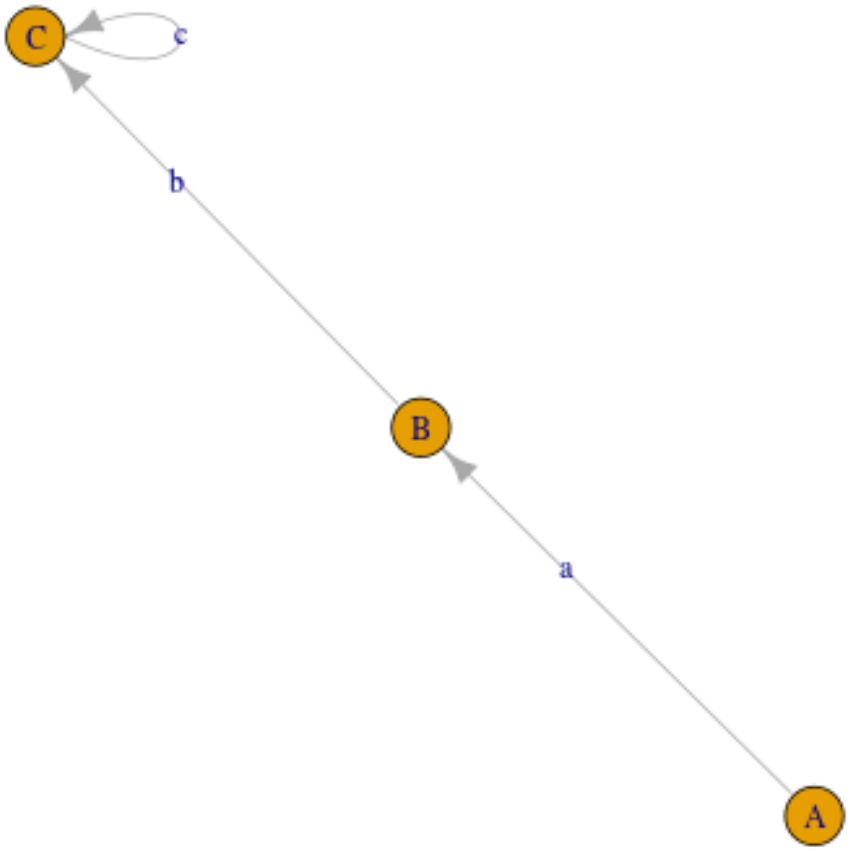


Figure 5: A directed graph.

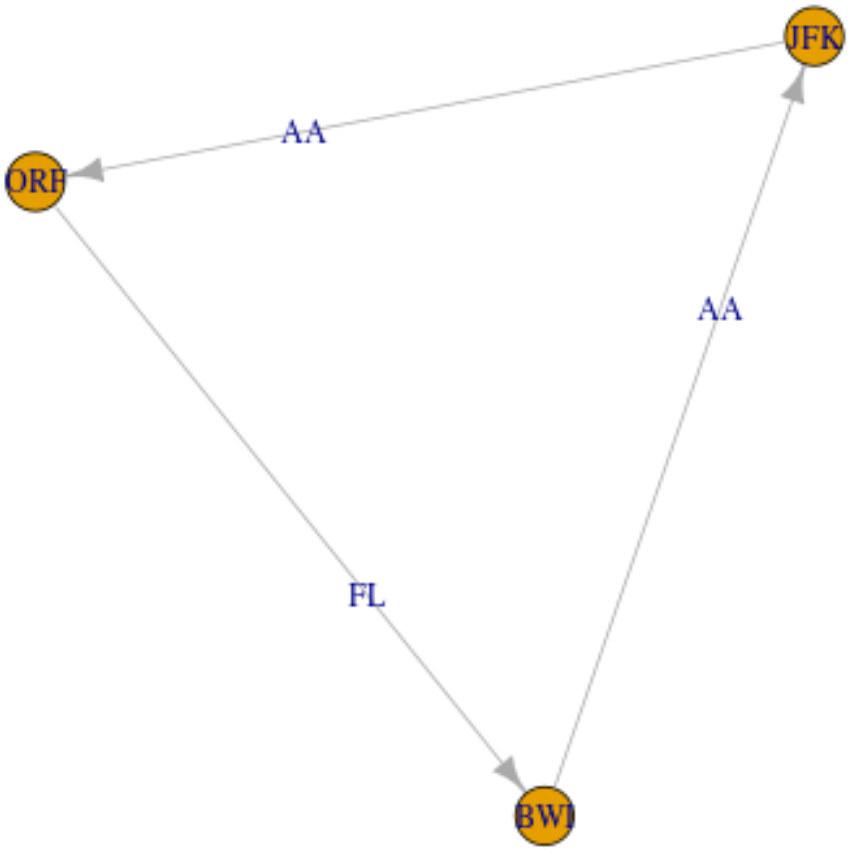


Figure 6: An airline route on the East Coast.

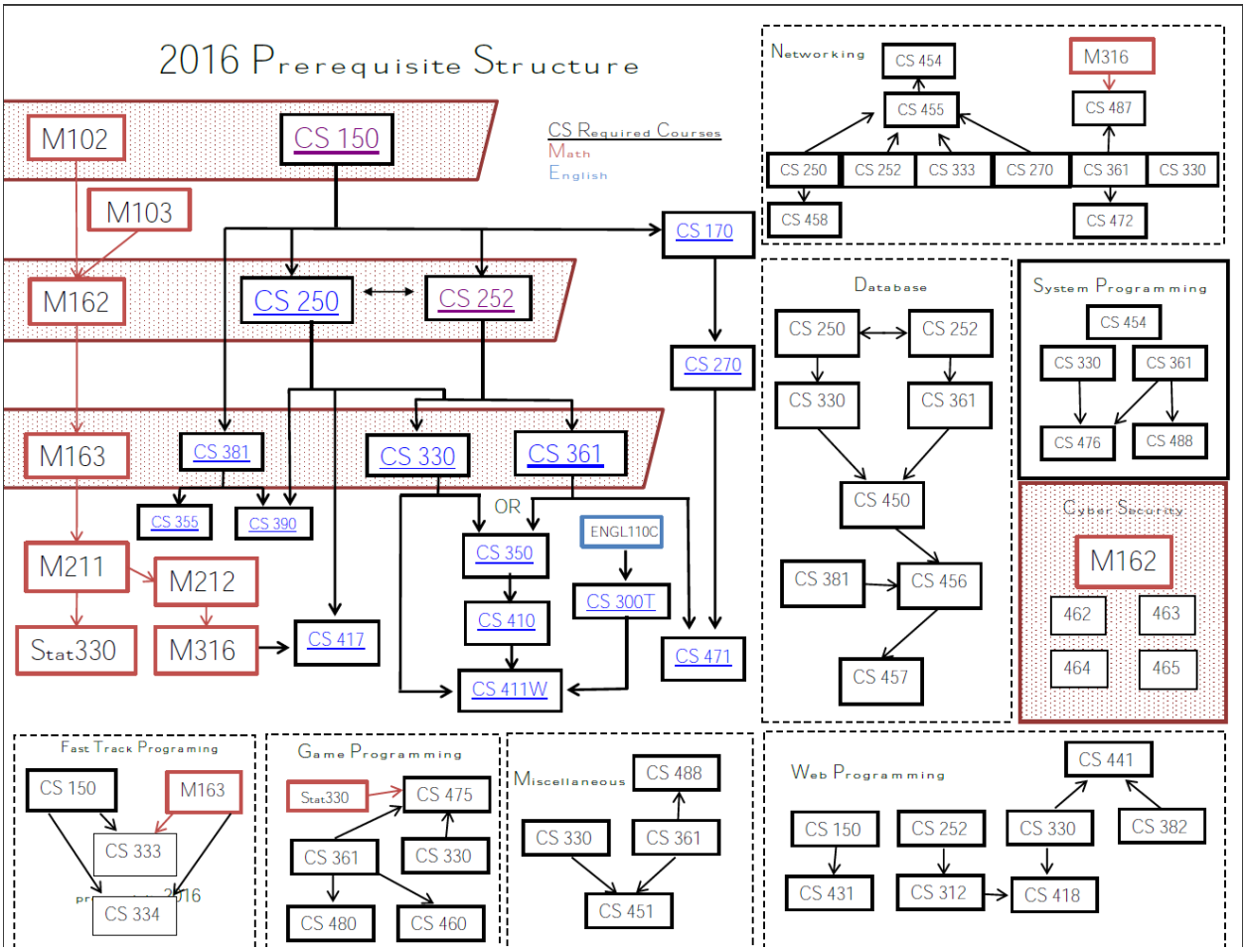


Figure 7: Old Dominion University (ODU) Computer Science undergraudate prerequisistes. The graph has examples of directed, unidirectional, and bi-directional edges. Courses in the graph can be connected or unconnected.

ACID compliance means that all database transactions will either complete fully or not at all, and that the database will be saved to external storage at the end of a transaction.

A database normally has these types of transactions:

C. *Create*: create an object in the database (a table, a row, a node, a relationship, etc.).

R. *Read*: read/report something from the database.

U. *Update*: update/change some-ting from the database (this includes adding something new to the database).

D. *Delete*: remove something from the database. If the thing being deleted has connections to other database objects, this deletion could have a significant effect on the database.

Neo4j is available under different licensing terms[1].

- Community Edition – available for free. Designed for personal use and limited practical database size.
- Enterprise Edition – available for a fee. Designed for commercial deployments where scale and availability are important.

Neo4j is a Java based application. The neo4j Java application starts an HTTP server, and executes "CRUDy" transactions on the database using by using the Cypher Query language[2]. Cypher is a relatively simple "ASCII art" language that still enables complex queries to be formed and executed efficiently. Because neo4j is a Java application, it inherits all the strengths and limitations of the Java Virtual Machine (JVM) that it uses.

1.3 Comparison of database technologies

Neo4j is a graph database, and has its own ideas and concepts about representing the relationship between data (see Table 1)[5].

2 Approach

2.1 Neo4j database installation

Neo4j is available in two different formats[3]:

- Enterprise – The Neo4j Enterprise Edition offers incredible power and flexibility, with enterprise-grade availability, management and scale-up & scale-out capabilities.

[1] `http://neo4j.com/licensing/`
[2] `http://neo4j.com/developer/cypher-query-language/`
[3] `http://neo4j.com/download/`

Table 1: A comparison of different database terms and ideas. "Well known" ideas from relational database management systems (RDBMS) are compared to those in key/value (K/V), columnar, document, graph, and Redis based database management systems.

RDBMS	K/V	Columnar	Doc.	Graph	Redis
DB. instance	cluster	cluster	instance	—	—
database	—	namespace	—	—	id
table	bucket	table	collection	node label	—
row	key-value	row	document	node id	key-value
rowid	key	—	_id	—	key
col.	—	col. fam.	—	attribute	key
schema	—	—	database	—	
join	—	—	DBRef	—	INT
—	—	—	—	rel. id	—

- Community – Ideal for learning, and smaller do-it-yourself projects that do not require high levels of scaling. Excludes professional services and support.

I chose the Community version because it is free.

2.2 Starting neo4j

The Community version is downloaded as a 62MByte tar ball (for *nix operating systems). The tar ball can be moved to any convenient directory (known as NEO4HOME) and untried with: `tar xref neo4j-community-3.0.1-unix.tar.gz`.

Neo4j can be started with: `$NEO4HOME/bin/neo4j console` (see Figure 8)

Neo4j will now be available.

2.3 Interacting with the database

At least three different ways exist for interacting with the database. They are:

1. Via a browser at `http://localhost:7474/browser/` (see Figures 9 and 10)
2. Via the neo4j-shell at `$NEO4JHOME/bin/neo4j-shell` (see Figure 11).
3. Via programming language bindings.

All interactions with the database are based on the Cypher language[4]. Cypher statements and commands can be entered manually at the neo4j-shell prompt, the browser $ prompt,

[4] `http://neo4j.com/developer/cypher/`

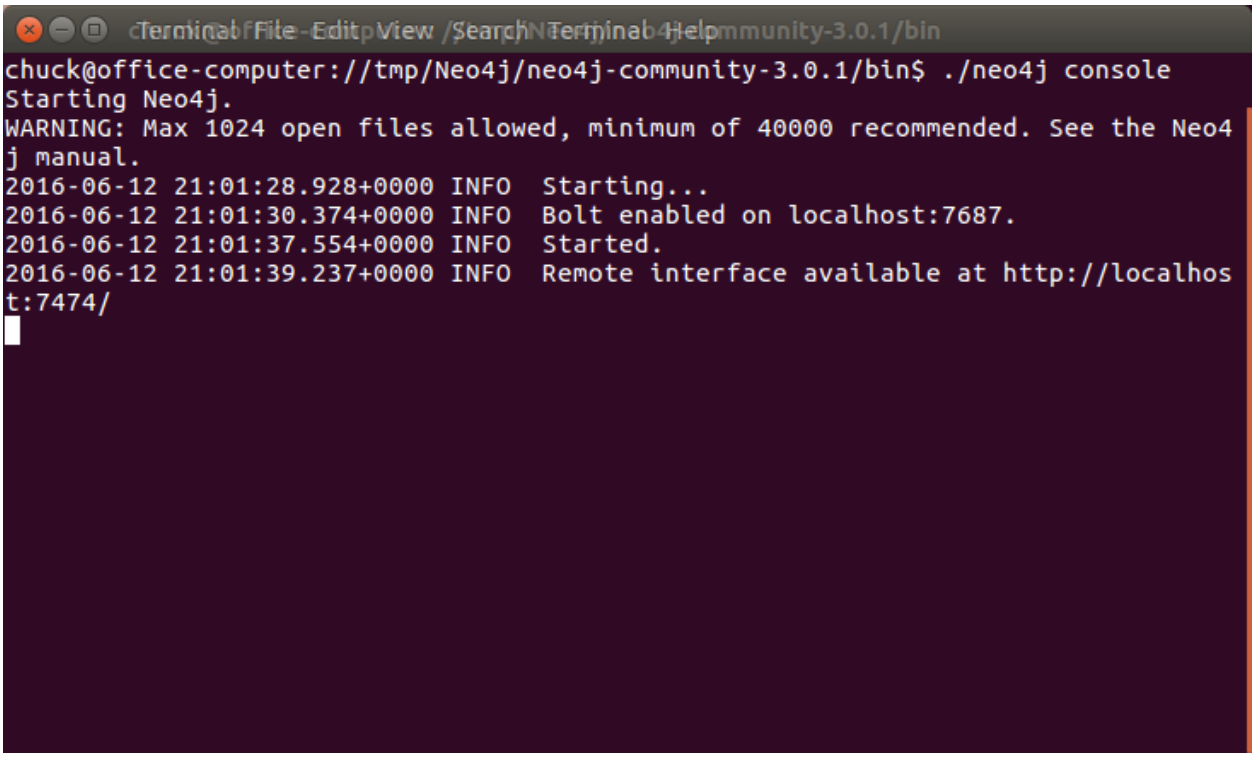


Figure 8: Starting neo4j in a terminal window.

via a file read processed by the neo4j-shell command line interface, or sent via language bindings.

The neo4j-shell accepts a collection of command line arguments (see Figure 12).

The browser interface is good for browsing a database, performing simple queries, and gaining confidence that the database is acting the way it should. The neo-shell interface is good for probing the database and constructing more complex queries because its error messages are more informative than the browser's. Packaging a collection of Cypher commands into one file that can be processed by the neo4j-shell from the command line makes it easy and fast to do a lot of operations. The programming interface enables you to perform operations on the data that are not supported by Cypher.

All three approaches were used while creating this report. Some approaches worked better than others (see Section A). The final approach was to write an R program[5] to load the database from local files. Total database load time was not recorded but was not excessive either. Time was probably on the order of 25 minutes to load the airport and route data.[6]

[5]The program is called `airlinePaths.R` and is attached to this report.

[6]By way of comparison, Nicole White (the Neo Technology data scientist) responsible for the R RNeo4j library felt that adding 313,227.000 nodes in 42,500.000 using an Enterprise version of Neo4j in milliseconds was "not too bad" (see `https://www.youtube.com/watch?v=Eh_79goBRUk` at time 27:12). I measured 198,998.000 nodes in 49,250.000 milliseconds. My measured times are not grossly different than hers.

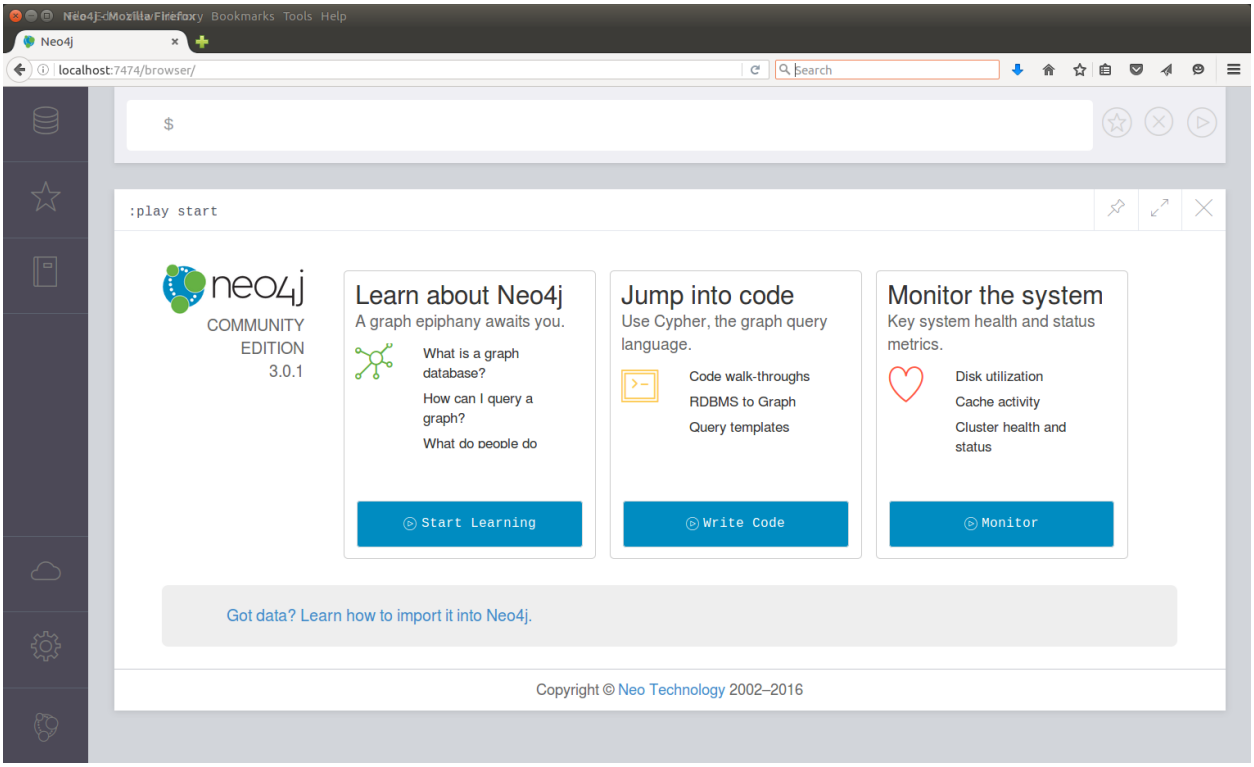


Figure 9: Neo4j browser homepage. Use `http://localhost:7474/browser/` to access the database.

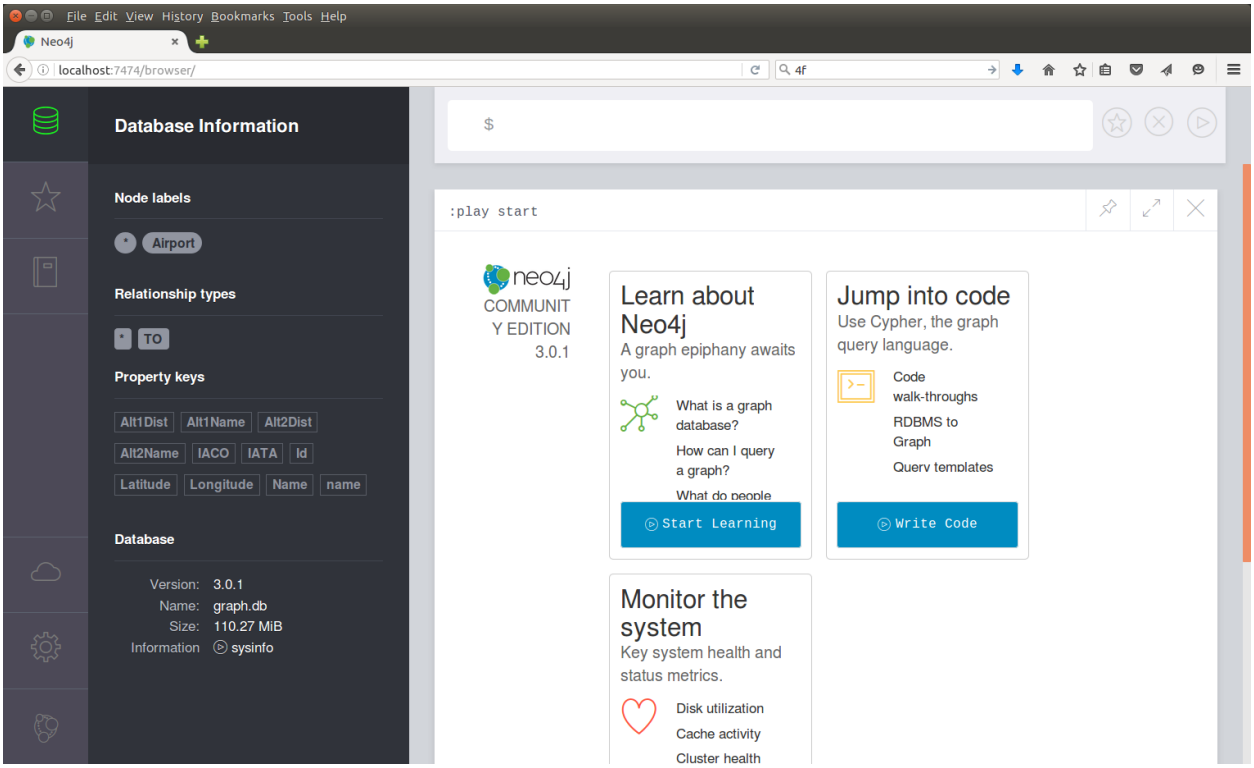


Figure 10: Neo4j browser database information page.

Figure 11: Neo4j shell starting prompt.

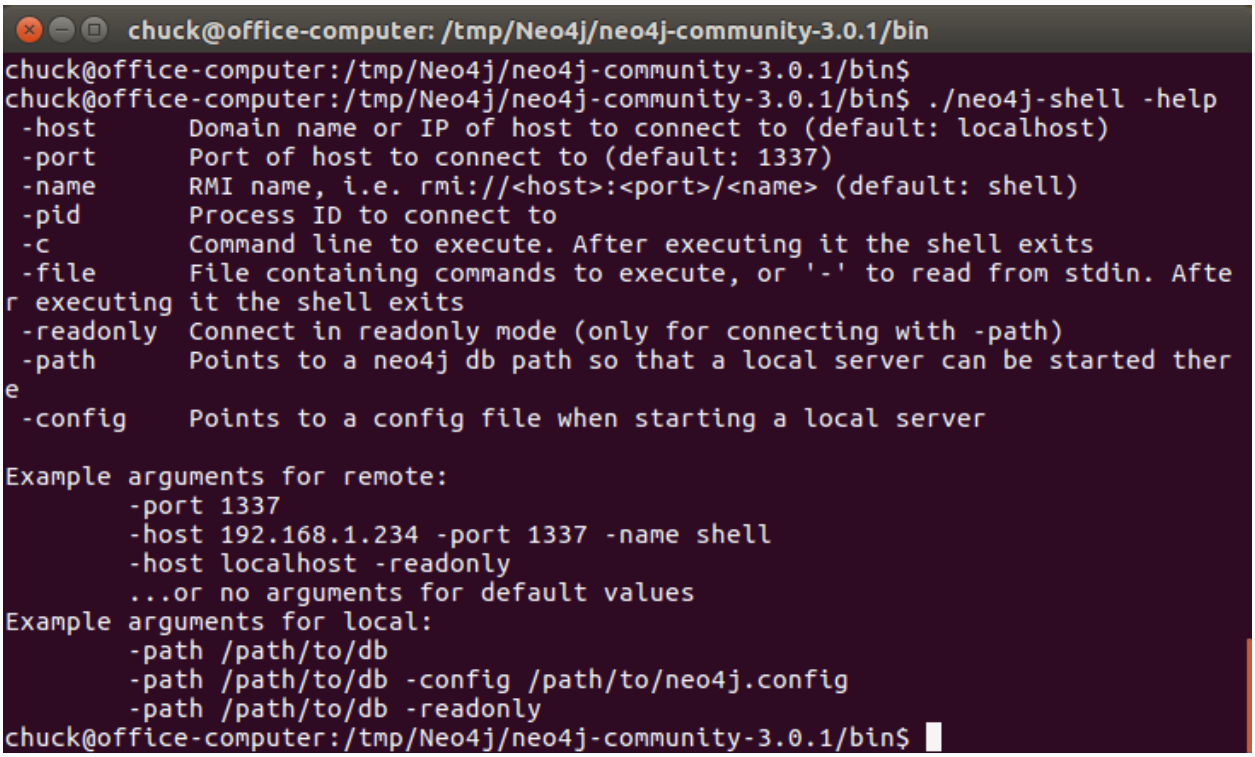


Figure 12: Neo4j shell command line arguments.

Table 2: EYW-YXY (Key West Intl to Whitehorse Intl) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Key West Intl	EYW	KEYW	24.556	−81.760
Orlando Intl	MCO	KMCO	28.429	−81.309
Mc Carran Intl	LAS	KLAS	36.080	−115.152
Vancouver Intl	YVR	CYVR	49.194	−123.184
Whitehorse Intl	YXY	CYXY	60.710	−135.067

3 Results

A series of IATA coded airports were selected at random to see how the system performed in attempting to find the shortest path that connected the airports. The shortest path is defined as the fewest number of hops between the starting airport and the ending airport. It is possible that a different path would be chosen if the definition of shortest was changed to "fastest" or "aircraft size" or "maximum travel time per leg" or "airline code" or some other criteria.

These "weights" could be applied to the legs between any of the airports and the shortest path would be one that was the sum of these weights.

The ggmap library was used to create the images in the following subsections. It appears that lines that leave the mapping area are not drawn from where they start to the edge of the map. Nor are lines drawn that originate outside the bounds of the map and terminate on the map. In this sense, the idea of a "viewport" into the visualized data is missing and can create discontinuities at the boundaries.

3.1 EYW-YXY (Key West Intl to Whitehorse Intl)

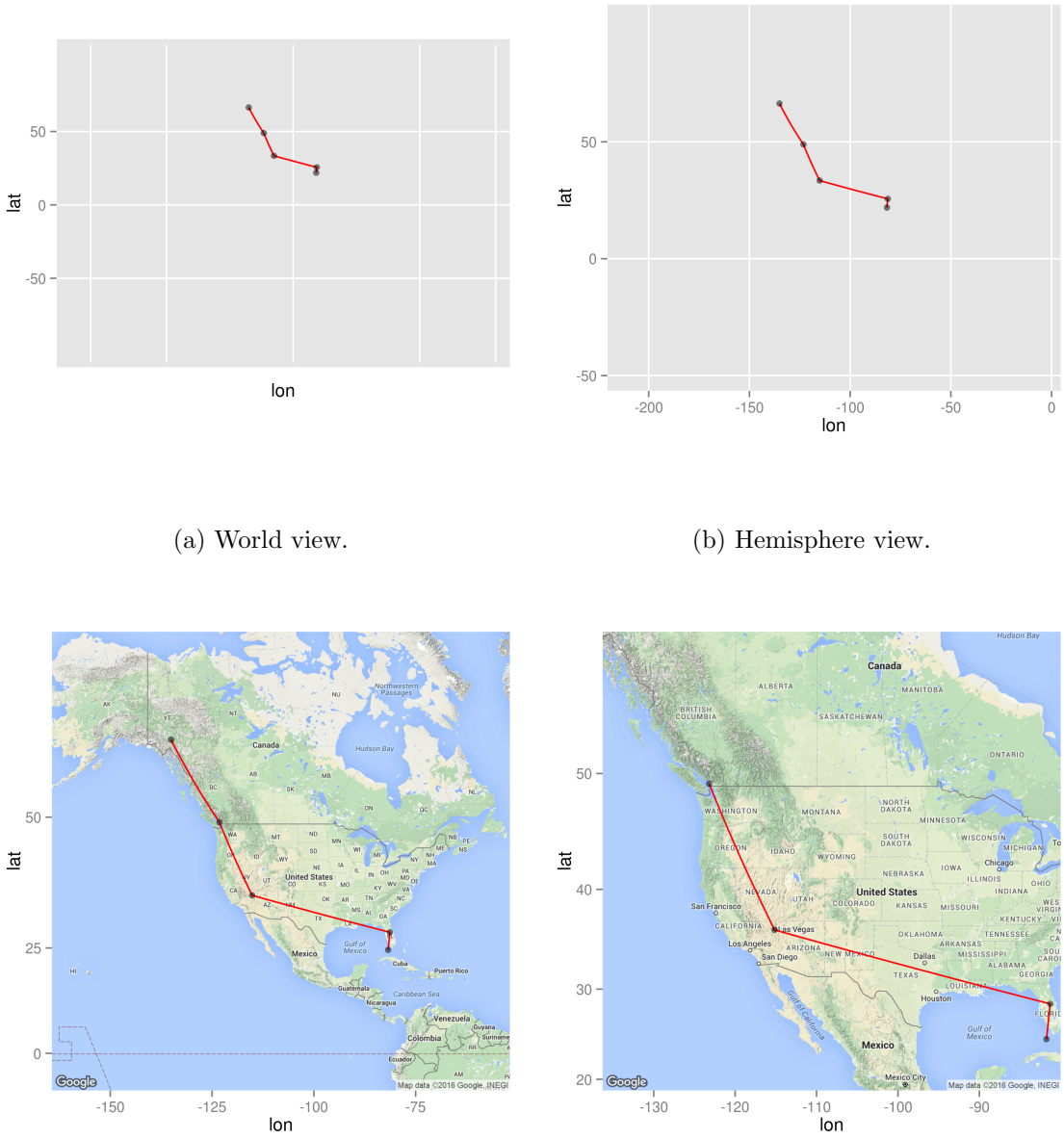
The various airports involved in the EYW-YXY (Key West Intl to Whitehorse Intl) flight path are listed (see Table 2) and shown on maps of different scales (see Figure 13).

3.2 ITO-KEF (Hilo Intl to Keflavik International Airport)

The various airports involved in the ITO-KEF (Hilo Intl to Keflavik International Airport) flight path are listed (see Table 3) and shown on maps of different scales (see Figure 14).

3.3 ORF-BPT (Norfolk Intl to Southeast Texas Rgnl)

The various airports involved in the ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) flight path are listed (see Table 4) and shown on maps of different scales (see Figure 15).



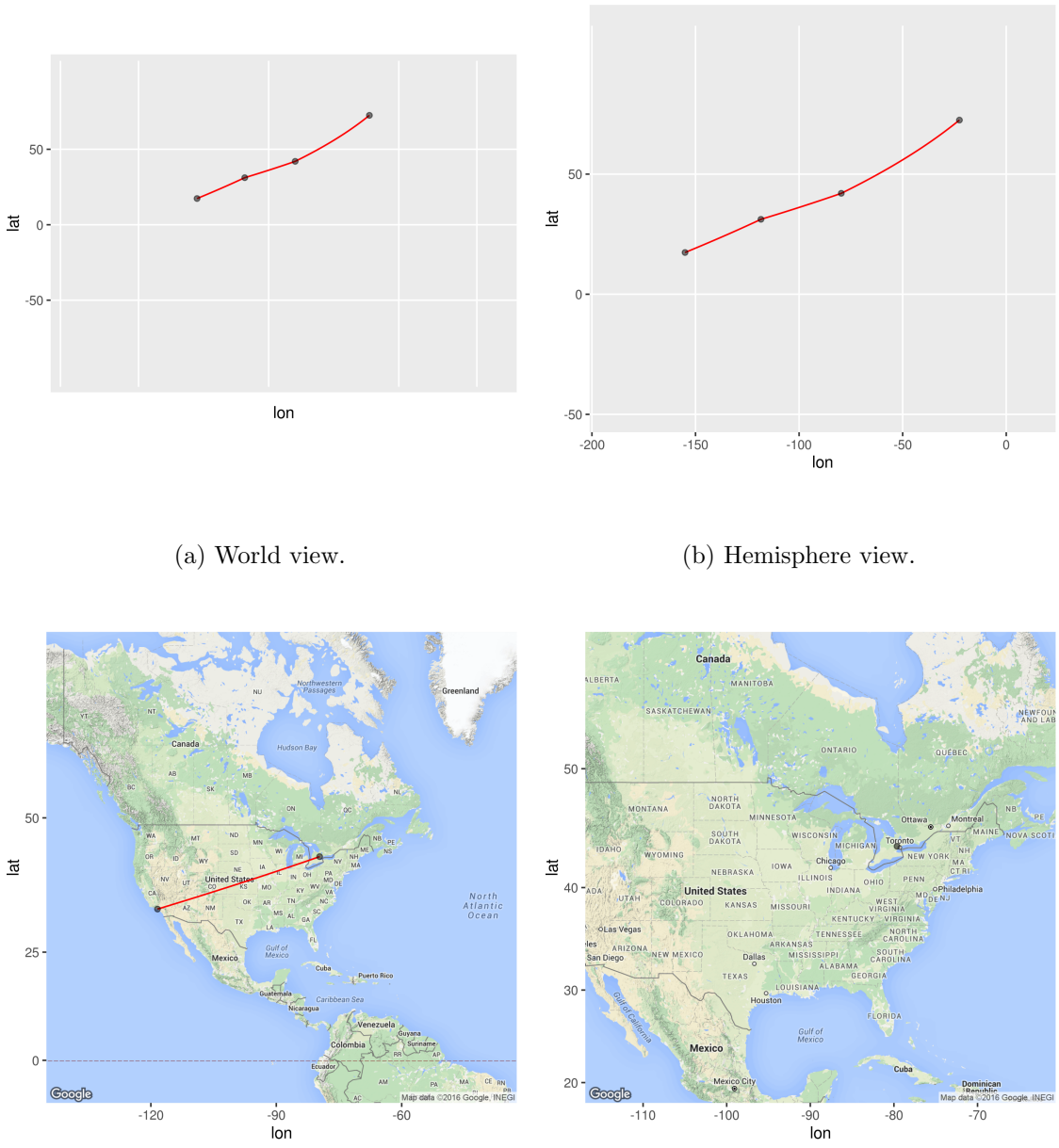
(a) World view.

(b) Hemisphere view.

(c) Continental view.

(d) Local view.

Figure 13: EYW-YXY (Key West Intl to Whitehorse Intl) flight path. The same are data shown at different scales.



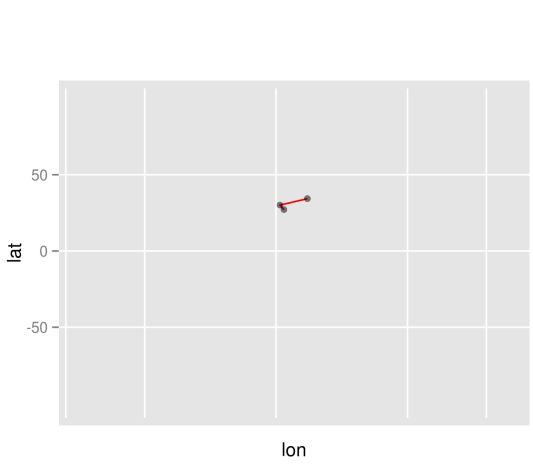
(a) World view.

(b) Hemisphere view.

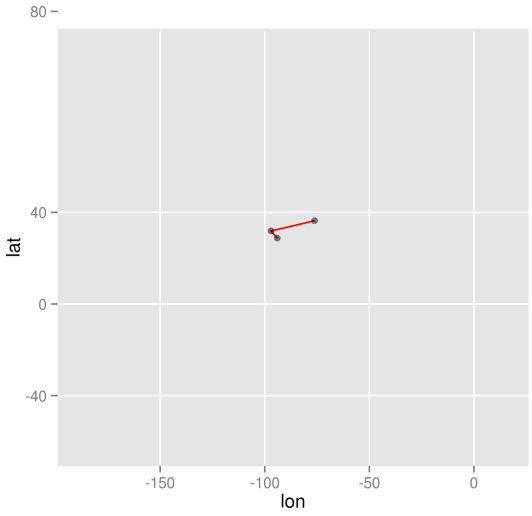
(c) Continental view.

(d) Local view.

Figure 14: ITO-KEF (Hilo Intl to Keflavik International Airport) flight path. The same are data shown at different scales.



(a) World view.



(b) Hemisphere view.

(c) Continental view.

(d) Local view.

Figure 15: ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) flight path. The same are data shown at different scales.

Table 3: ITO-KEF (Hilo Intl to Keflavik International Airport) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Hilo Intl	ITO	PHTO	19.721	−155.049
Los Angeles Intl	LAX	KLAX	33.943	−118.408
Lester B Pearson Intl	YYZ	CYYZ	43.677	−79.631
Keflavik International Airport	KEF	BIKF	63.985	−22.606

Table 4: ORF-BPT (Norfolk Intl to Southeast Texas Rgnl) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Norfolk Intl	ORF	KORF	36.895	−76.201
Dallas Fort Worth Intl	DFW	KDFW	32.897	−97.038
Southeast Texas Rgnl	BPT	KBPT	29.951	−94.021

3.4 ORF-GME (Gomel to Gomel)

The various airports involved in the ORF-GME (Gomel to Gomel) flight path are listed (see Table 5) and shown on maps of different scales (see Figure 16).

3.5 ORF-RGL (Norfolk Intl to Rio Gallegos)

The various airports involved in the ORF-RGL (Norfolk Intl to Rio Gallegos) flight path are listed (see Table 6) and shown on maps of different scales (see Figure 17).

3.6 ORF-THU (Norfolk Intl to Thule Air Base)

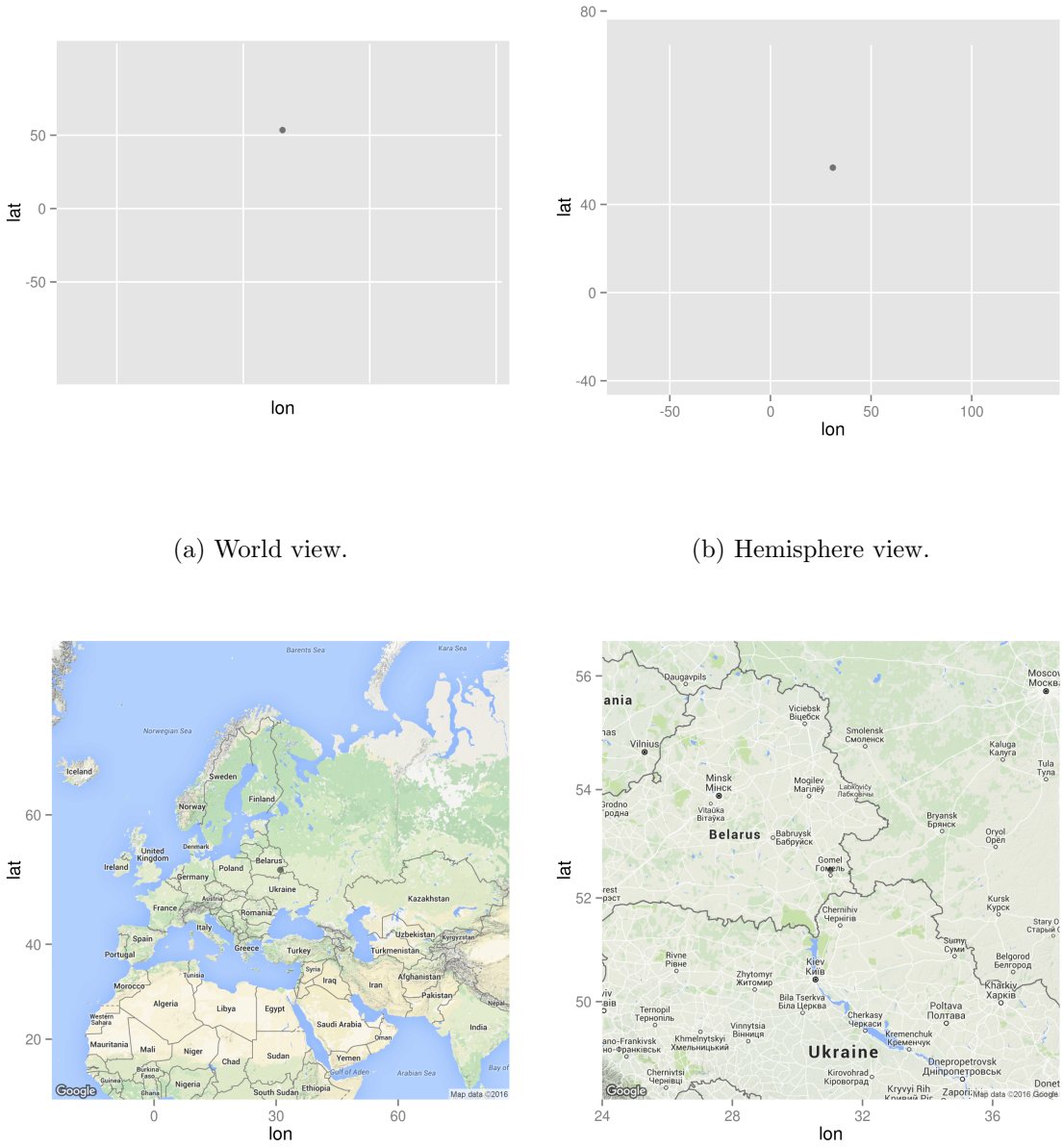
The various airports involved in the ORF-THU (Norfolk Intl to Thule Air Base) flight path are listed (see Table 7) and shown on maps of different scales (see Figure 18).

3.7 YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery)

The various airports involved in the YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) flight path are listed (see Table 8) and shown on maps of different scales (see Figure 19).

Table 5: ORF-GME (Gomel to Gomel) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Gomel	GME	UMGG	52.527	31.017



(a) World view.

(b) Hemisphere view.

(c) Continental view.

(d) Local view.

Figure 16: ORF-GME (Gomel to Gomel) flight path. The same are data shown at different scales.

Table 6: ORF-RGL (Norfolk Intl to Rio Gallegos) airports along the flight path.

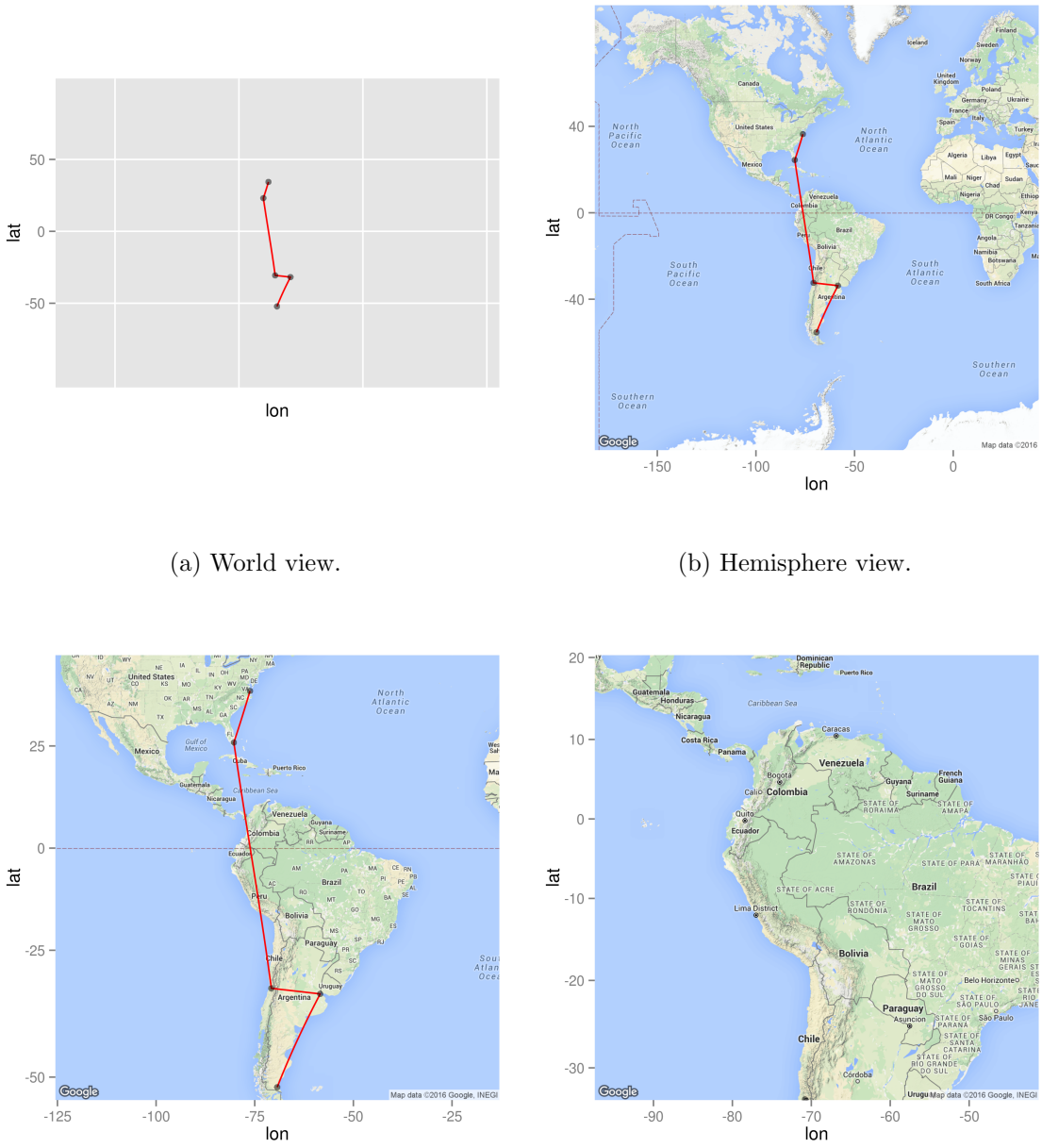
Name	IATA	IACO	Lat.	Lon.
Norfolk Intl	ORF	KORF	36.895	−76.201
Miami Intl	MIA	KMIA	25.793	−80.291
Arturo Merino Benitez Intl	SCL	SCEL	−33.393	−70.786
Aeroparque Jorge Newbery	AEP	SABE	−34.559	−58.416
Rio Gallegos	RGL	SAWG	−51.609	−69.313

Table 7: ORF-THU (Norfolk Intl to Thule Air Base) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Norfolk Intl	ORF	KORF	36.895	−76.201
Chicago Ohare Intl	ORD	KORD	41.979	−87.905
Kastrup	CPH	EKCH	55.618	12.656
Sondre Stromfjord	SFJ	BGSF	67.017	−50.689
Ilulissat	JAV	BGJN	69.234	−51.051
Upernavik Airport	JUV	BGUK	72.790	−56.131
Qaanaaq Airport	NAQ	BGQQ	77.489	−69.389
Thule Air Base	THU	BGTL	76.531	−68.703

Table 8: YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Whitehorse Intl	YXY	CYXY	60.710	−135.067
Vancouver Intl	YVR	CYVR	49.194	−123.184
Dallas Fort Worth Intl	DFW	KDFW	32.897	−97.038
Arturo Merino Benitez Intl	SCL	SCEL	−33.393	−70.786
Aeroparque Jorge Newbery	AEP	SABE	−34.559	−58.416



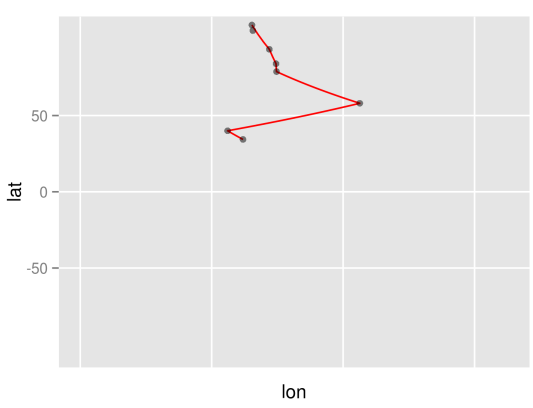
(a) World view.

(b) Hemisphere view.

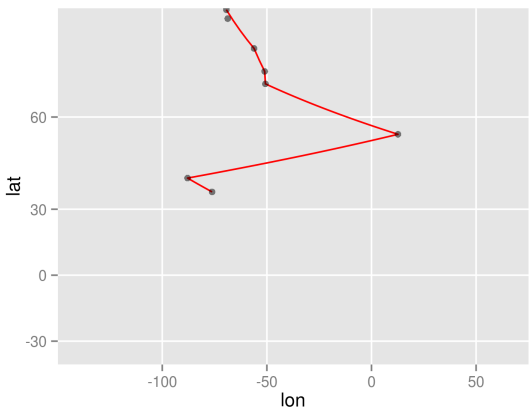
(c) Continental view.

(d) Local view.

Figure 17: ORF-RGL (Norfolk Intl to Rio Gallegos) flight path. The same are data shown at different scales.

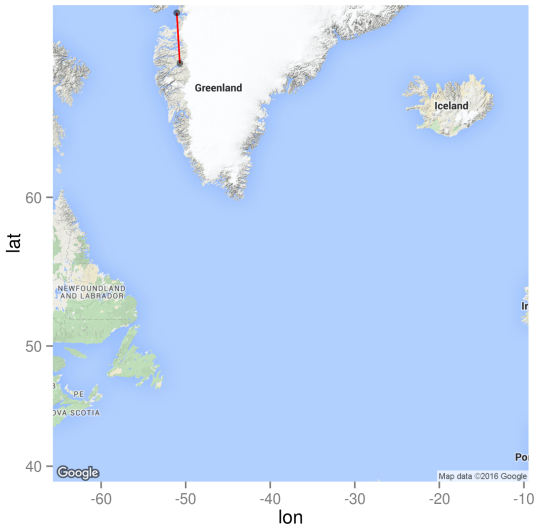


(a) World view.



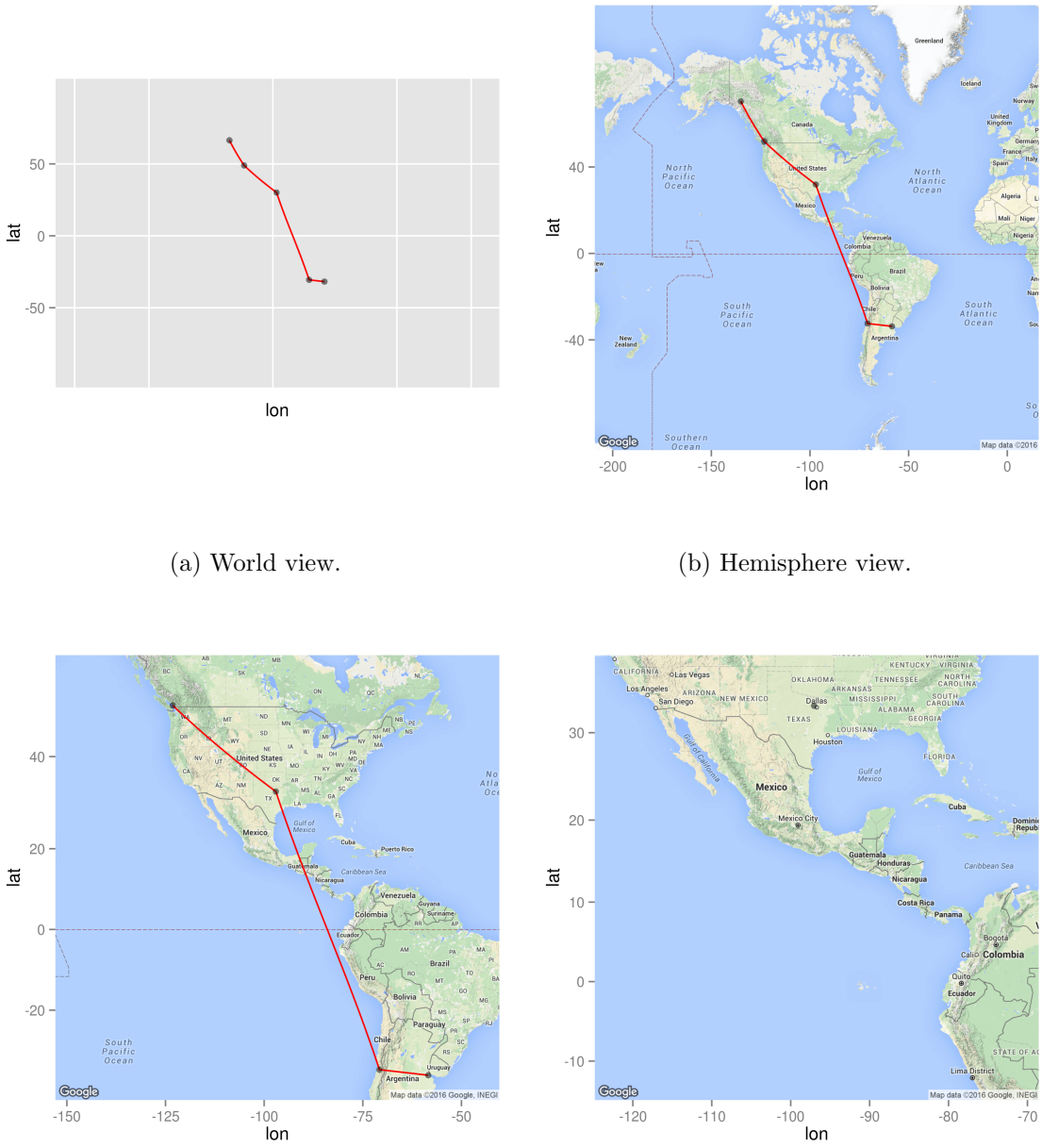
(b) Hemisphere view.

(c) Continental view.



(d) Local view.

Figure 18: ORF-THU (Norfolk Intl to Thule Air Base) flight path. The same are data shown at different scales.



(a) World view.

(b) Hemisphere view.

(c) Continental view.

(d) Local view.

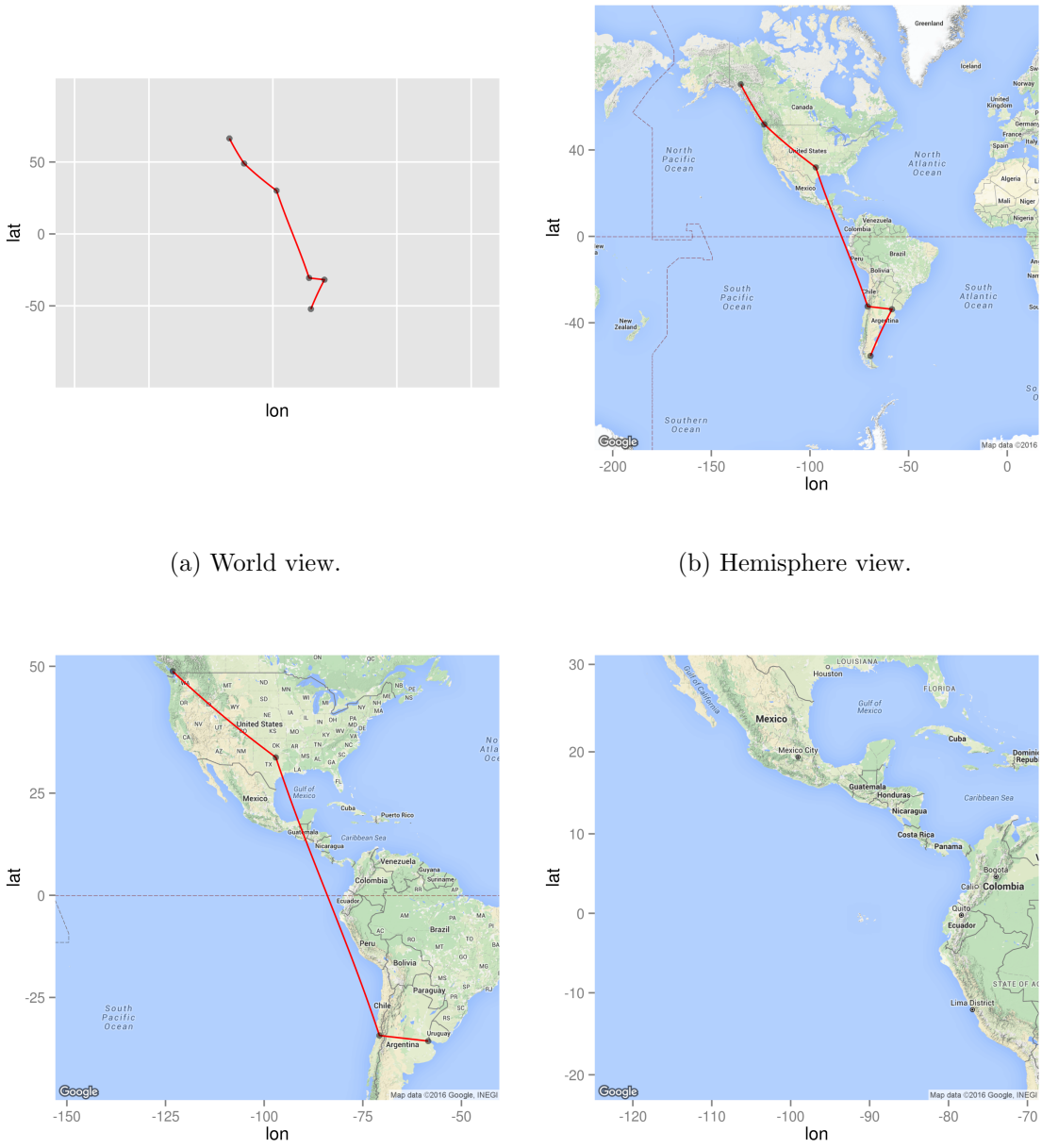
Figure 19: YXY-AEP (Whitehorse Intl to Aeroparque Jorge Newbery) flight path. The same are data shown at different scales.

Table 9: YXY-RGL (Whitehorse Intl to Rio Gallegos) airports along the flight path.

Name	IATA	IACO	Lat.	Lon.
Whitehorse Intl	YXY	CYXY	60.710	−135.067
Vancouver Intl	YVR	CYVR	49.194	−123.184
Dallas Fort Worth Intl	DFW	KDFW	32.897	−97.038
Arturo Merino Benitez Intl	SCL	SCEL	−33.393	−70.786
Aeroparque Jorge Newbery	AEP	SABE	−34.559	−58.416
Rio Gallegos	RGL	SAWG	−51.609	−69.313

3.8 YXY-RGL (Whitehorse Intl to Rio Gallegos)

The various airports involved in the YXY-RGL (Whitehorse Intl to Rio Gallegos) flight path are listed (see Table 9) and shown on maps of different scales (see Figure 20).



(a) World view.

(b) Hemisphere view.

(c) Continental view.

(d) Local view.

Figure 20: YXY-RGL (Whitehorse Intl to Rio Gallegos) flight path. The same are data shown at different scales.

4 Conclusion

Neo4j is a graph database management system. It is Java based and available in different configurations based on the licensing chosen. We installed and tested the community edition of neo4j using two different databases.

The first was the Internet Movie Database (IMDb), where we were interested in seeing the connections between different players. We were able to load all the IMDb tables we thought would be needed to answer the types of questions we were interested in asking. Unfortunately, we had to increase the Java Virtual Memory (JVM) heap size to 8GB, which as all the RAM in the test hardware. The test hardware started to use swap space and system performance was too slow to be useful.

The second database we used was the airport and route connecting data from the OpenFlight.org web site. This database could be loaded in about 45 minutes, after which queries would be completed generally within a half a second. The OpenFlight data was used to find the shortest path (if one existed) between source and destination airports. For our testing, the shortest path was defined as the path with the fewest number of legs between the source and destination. An alternative definition of shortest would be to make an assumption about the travel time between airports based on the type of aircraft normally flown between airports. Equipment information is available in the OpenFlight data. For each flight path, a series of geographic maps was created showing the path at different resolutions.

For fun, a small collection of geographic maps were created showing airports that did and did not have regularly scheduled airline service.

All programs used to load the database, query the database, generate textual and geographic maps are attached to this report.

Table 10: Time spent loading the IMDb via the neo4j-shell. The time to load the entire IMDb database was too long to be practical.

Size	% IMDb	R-seconds	Neo4j seconds	Nodes loaded
8,000	-	14.070	2.600	2,300
80,000	-	57.462	8.820	26,041
800,000	4	609.890	49.250	198,998
8,000,000	45	1,245.048	556.452	649,309
80,000,000	100	43,600,386.000	1,840,666.000	1,466,720

A Neo4j Lessons Learned

The first attempt to use neo4j as a graphical database was to investigate actors, actresses, and directors from the Internet Movie Database (IMDb)[7]. Portions of the IMDb database is available for download[8] for free. The actor's data file was downloaded and stored locally.

After initially experimenting with the different ways of loading data into the database, it appeared that using the neo4j-shell command line interface was the fastest, easiest, and least prone to human error. So loading the database was divided into two phases. The first was an R program to clean/normalize the data and format it for easy loading. While the second was loading the data. The wall clock time for loading different parts of the IMDb was collected (see Table 10).

Neo4j is a Java based application, and so runs in a Java Virtual Machine (JVM). It appears that large portions of the neo4j database is loaded into memory to be processed, and when the JVM runs out of memory, it throws an error. The amount of memory that JVM can use is controlled by entries in the `neo4j-wrapper.conf` file. Within this file, the `dbms.memory.heap.max_size` variable is commented out. It was incrementally increased until no errors were thrown. The final value was 8192 MBytes before all data could be processed. This was the total RAM available in the test machine, resulting in the machine swapping and totally unacceptable performance (see Figure 21).

The system's performance dictated that another database be used for this report.

Notes and observations.

The IMDb is an international database and has many Unicode characters. Neo4j appears to expect purely ASCII characters in its data files. Things like unescaped single and double quotes, cause spurious attributes to be created. The last approach that was tried was to base64 encode all fields. Cypher's load-csv function seemed to handle those fields OK.

[7] `http://www.imdb.com/`

[8] `http://www.imdb.com/interfaces`

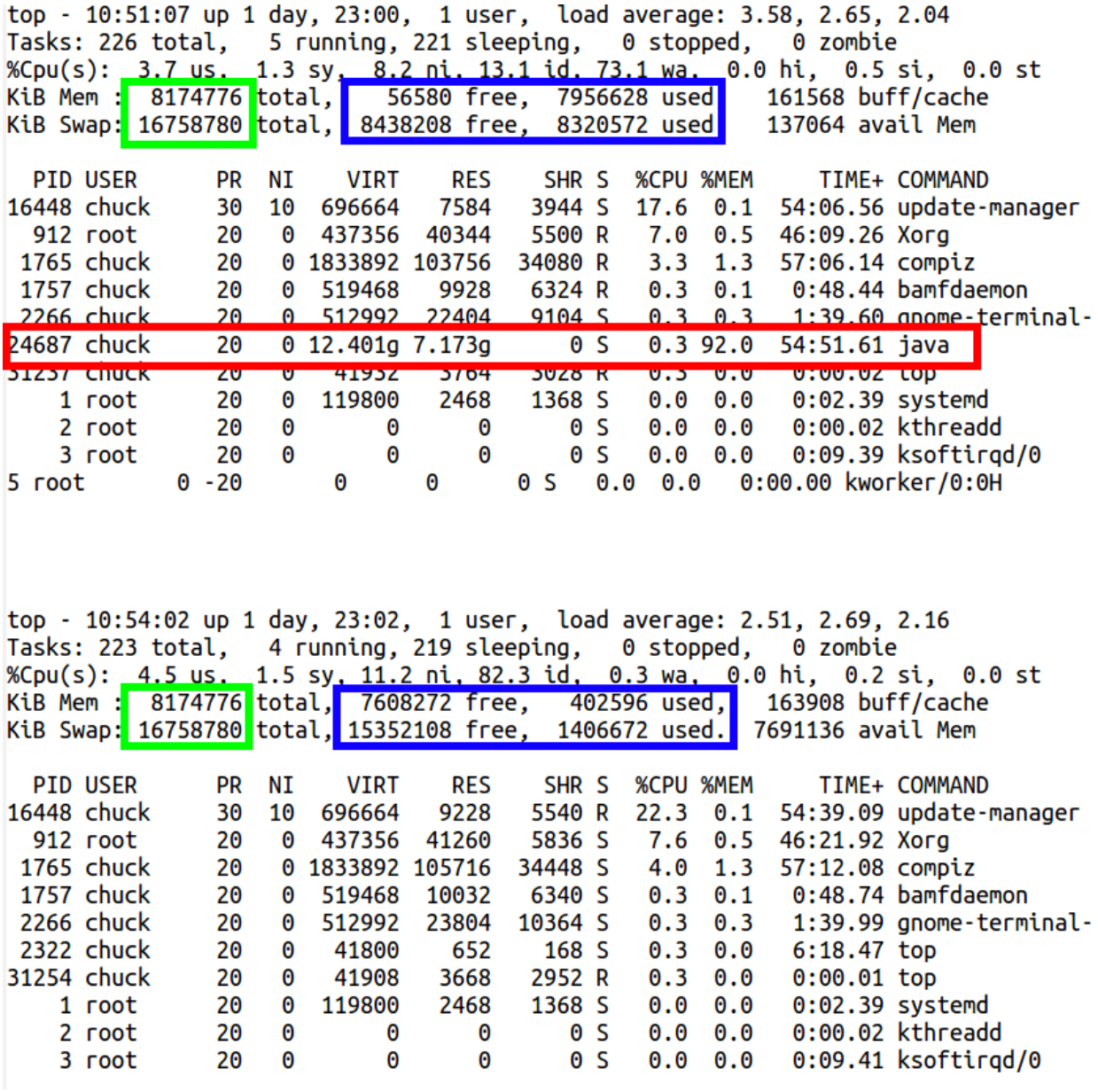


Figure 21: Computer system measurements before and after loading IMDb. Two "top" outputs were captured. One as neo4j was running and one after. System RAM and swap space are outlined in green. Free RAM and swap space are outlined in blue. The neo4j java application is outlined in red.

B Misc. files

The files used to create all these figures are attached to this report. They are:

1. airlinePaths.R - an R program used to:
   (a) Load the neo4j database with airport and connection information,
   (b) Find the shortest path between two IACA labeled airports (if one exists),
   (c) Create different geographic maps based on the path between the serviced airports, and
   (d) Create various support files for automatic report generation.
2. ipMap.pdf - a report generated by the ipMap.R program.
3. ipMap.R - an R program used to create a map of a small section of the Internet. The map starts from localhost and extends to a destination.
4. virginia.R - an R program used to create assorted maps showing serviced and non serviced airports.

Data comes from OpenFlights.org[9].

[9] `http://openflights.org/data.html`

C Miscellaneous maps

Once the database was up and running, there is a tendancy to "play" with it and see what it can do. The following images show the location of serviced and non-serviced airports.

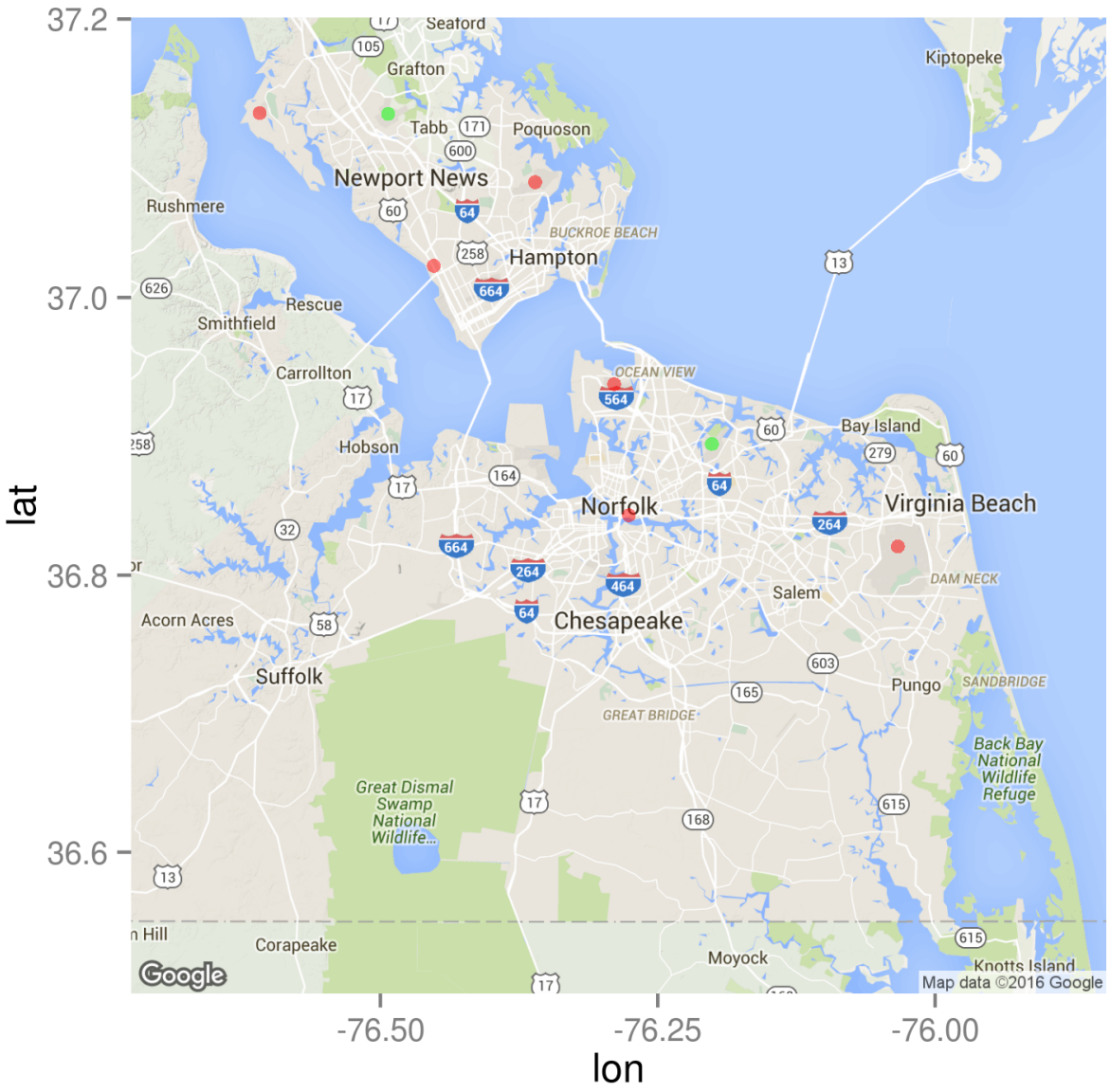


Figure 22: Serviced and non-serviced airports in and around Norfolk, Virginia. Serviced airports are shown as green. Non-serviced airports are in red.

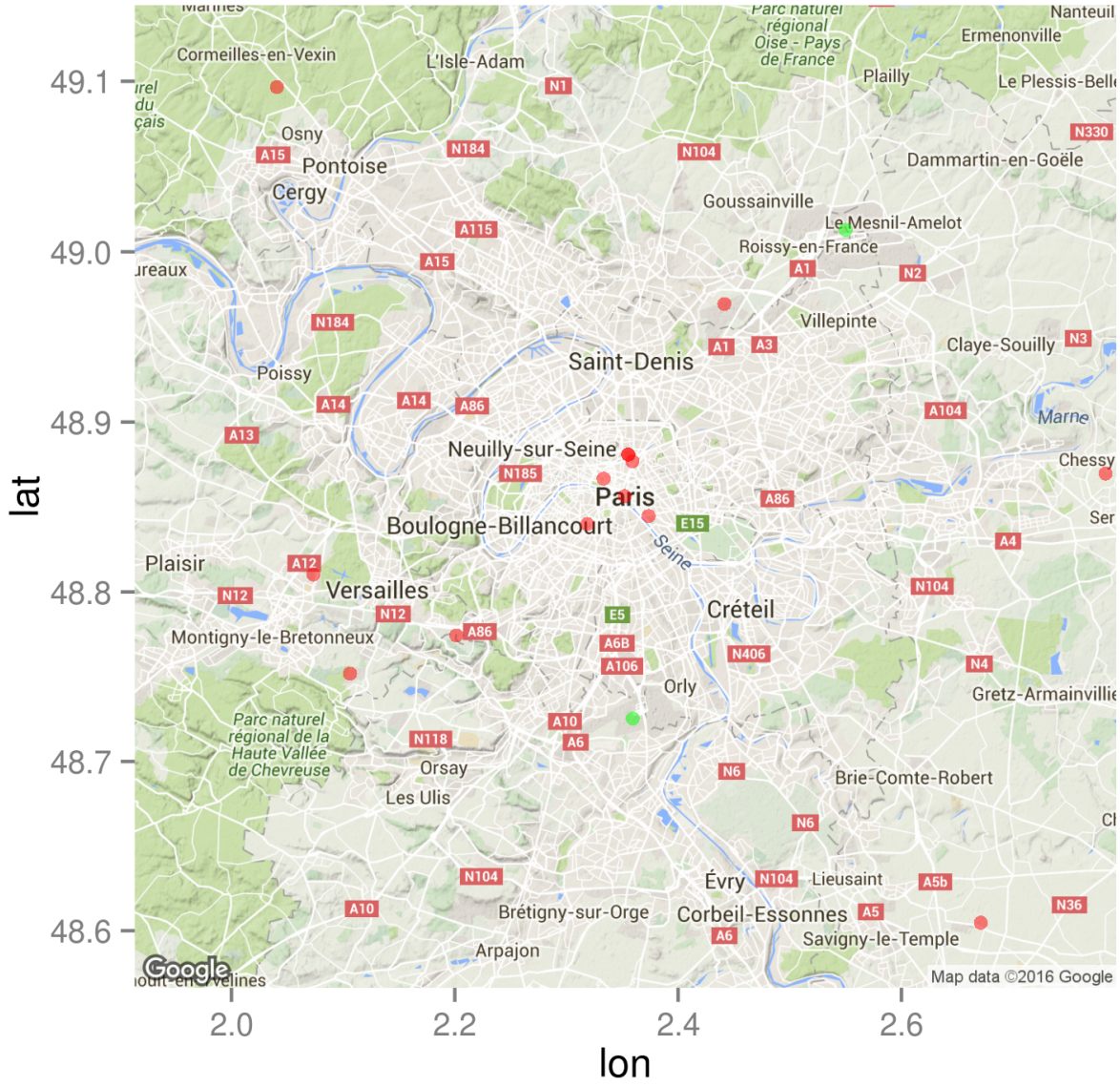


Figure 23: Serviced and non-serviced airports in and around Paris, France. Serviced airports are shown as green. Non-serviced airports are in red.

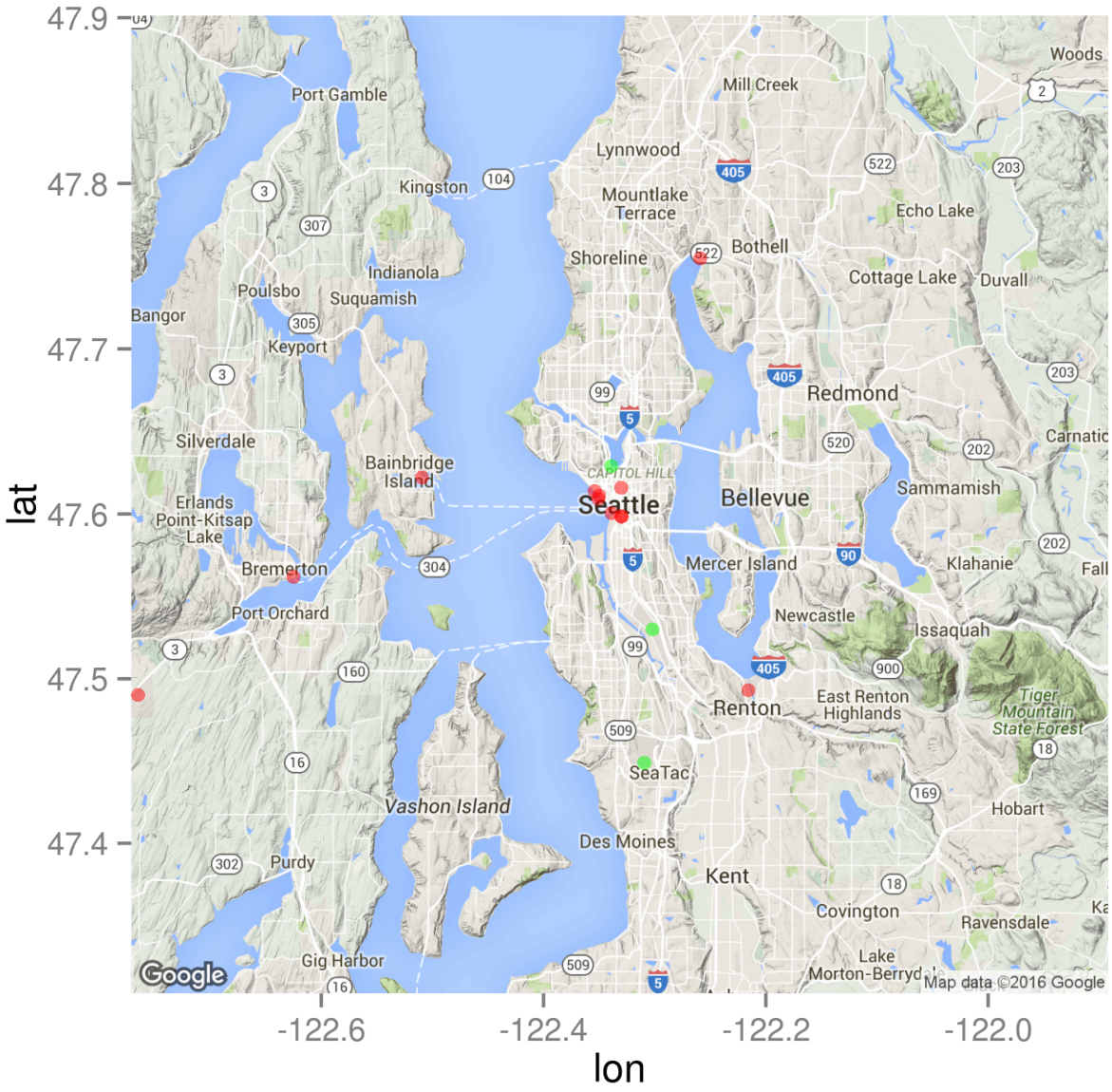


Figure 24: Serviced and non-serviced airports in and around Seattle, Washington. Serviced airports are shown as green. Non-serviced airports are in red.

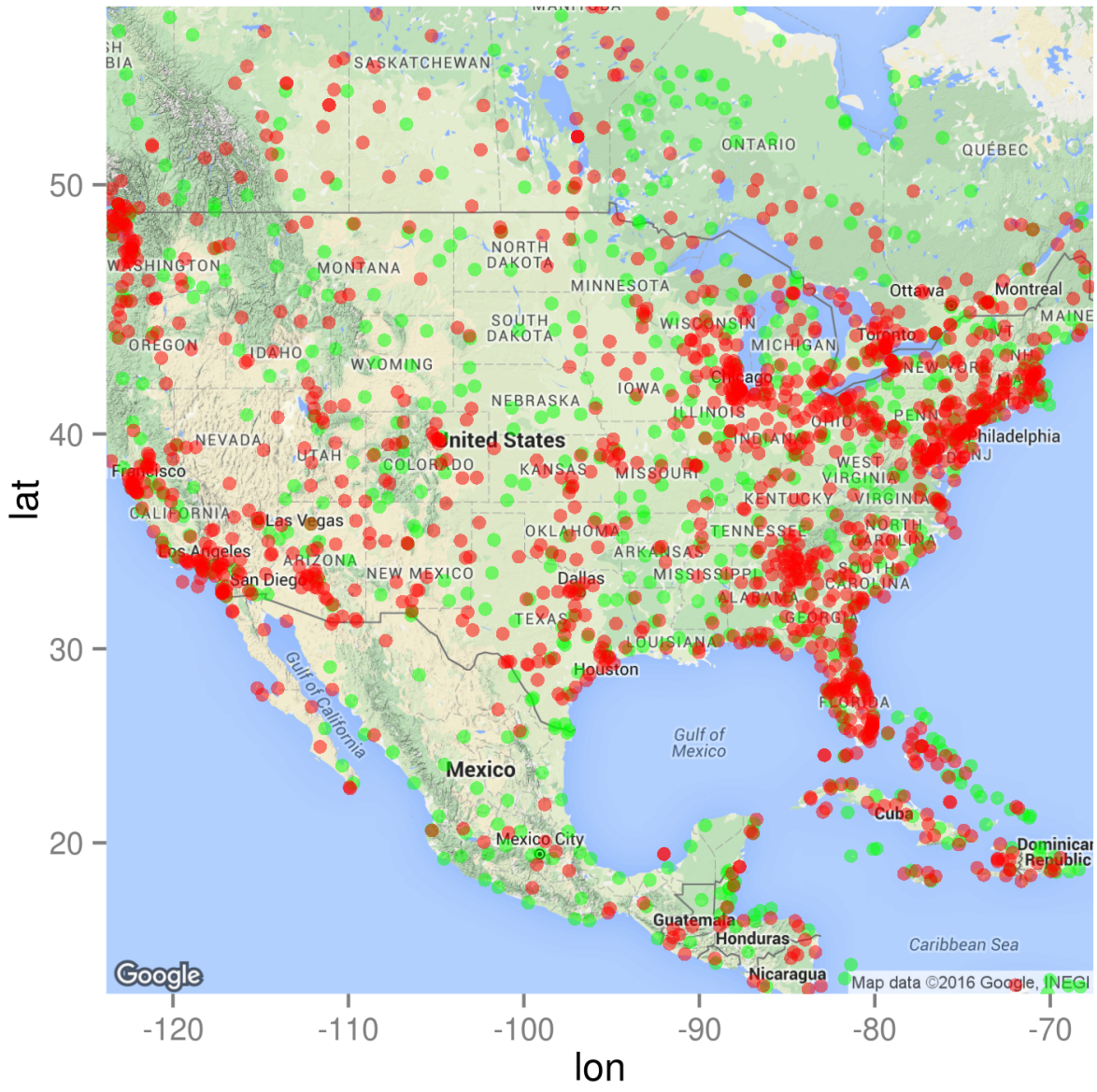


Figure 25: Serviced and non-serviced airports in and around the United States. Serviced airports are shown as green. Non-serviced airports are in red.

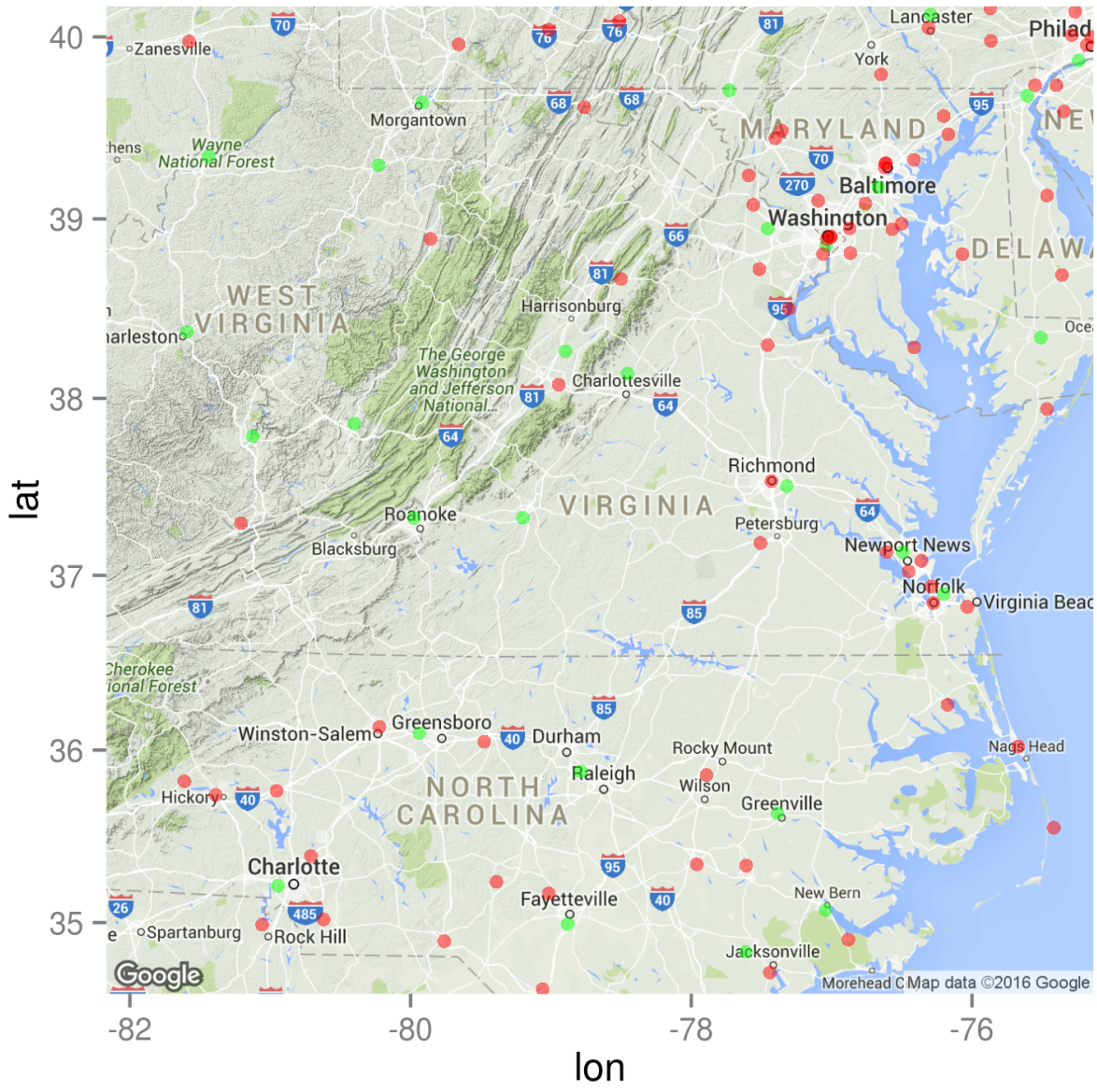


Figure 26: Serviced and non-serviced airports in and around Virginia. Serviced airports are shown as green. Non-serviced airports are in red.

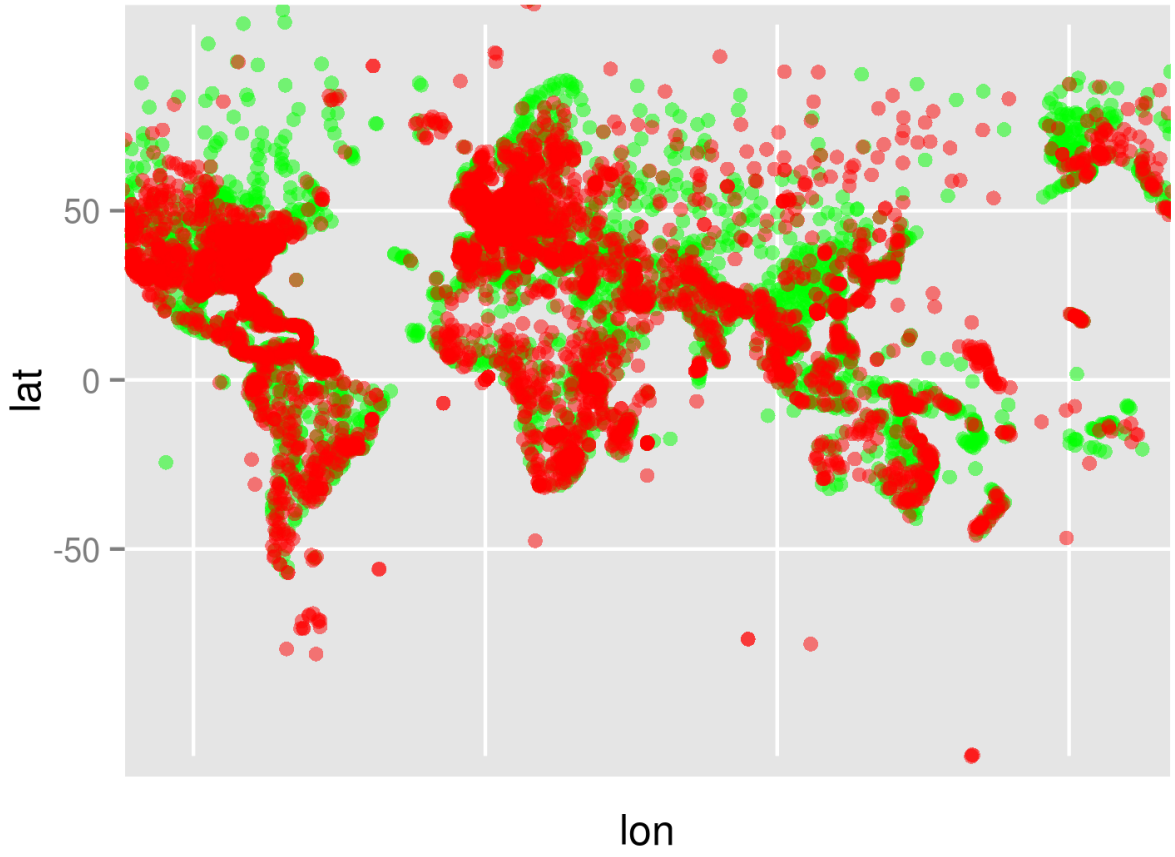


Figure 27: Serviced and non-serviced airports in and around the World. Serviced airports are shown as green. Non-serviced airports are in red.

D References

[1] Gerald L. Alexanderson, *About the cover:euler and königsbergs bridges: A historical view*, Bulletin of the American Mathematical Society **43** (2006).

[2] Brian Hopkins and Robin Wilson, *The truth about kõnigsberg*, The College Mathematics Journal **35** (2004), no. 3, 198.

[3] David Kahle and Hadley Wickham, *ggmap: Spatial visualization with ggplot2*, The R Journal **5** (2013), no. 1, 144–161.

[4] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0.

[5] Eric Redmond and Jim R Wilson, *Seven databases in seven weeks*, Pragmatic Bookshelf, 2012.

[6] Wikipedia Staff, *Neo4j*, `https://en.wikipedia.org/wiki/Neo4j`, 2016.

[7] Wolfram Staff, *Euler, leonhard (1707-1783)*, `http://scienceworld.wolfram.com/biography/Euler.html`.

[8] ———, *Königsberg bridge problem*, `http://mathworld.wolfram.com/KoenigsbergBridgeProblem.html`.