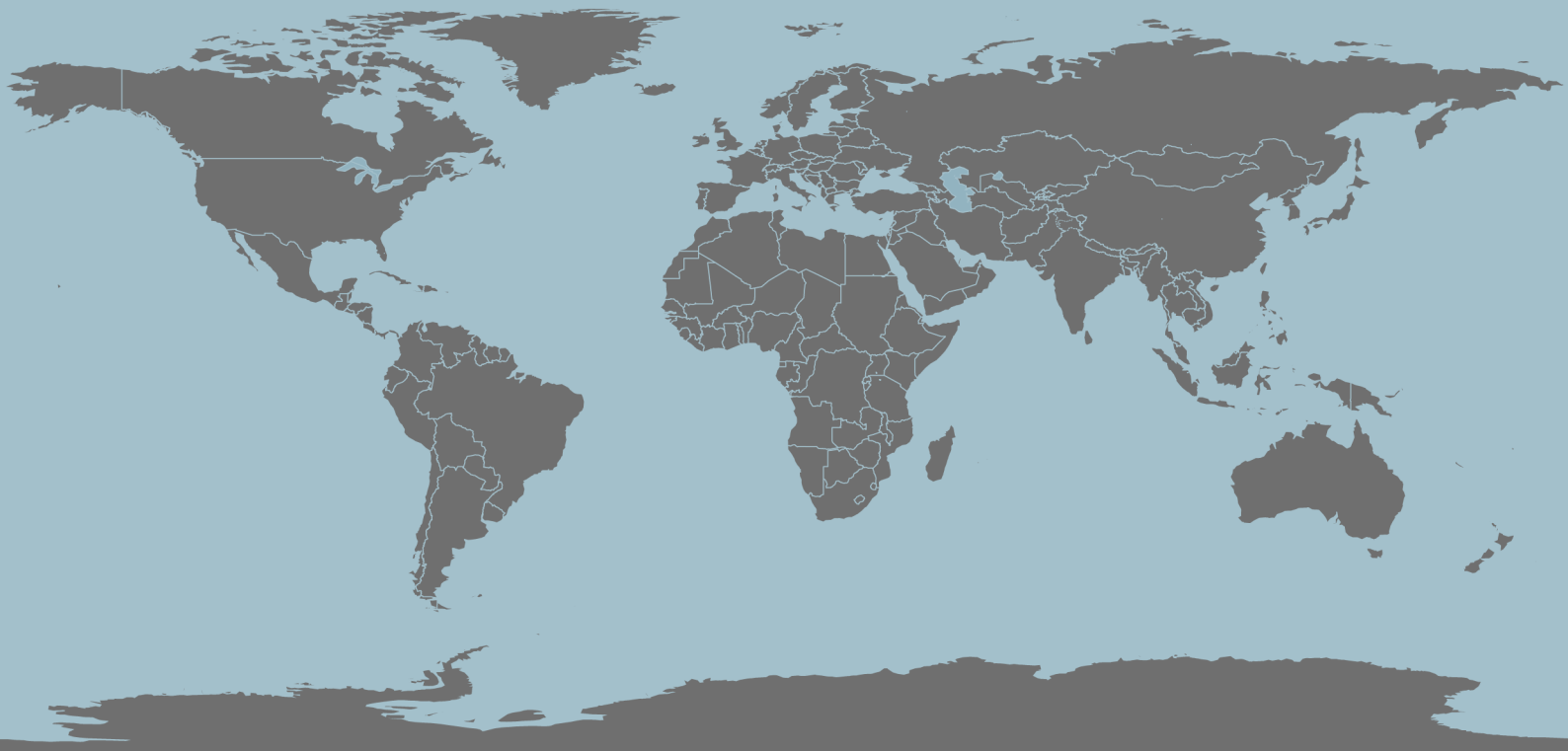
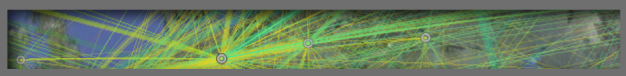
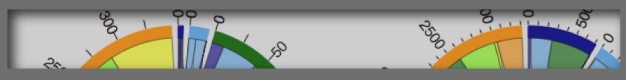
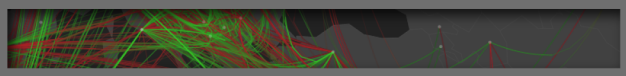


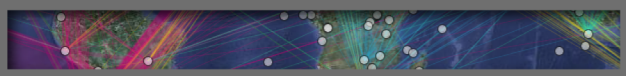
Resource Atlas

Global Resource Flow Data Visualization









User Manual

Universiteit Leiden and TU Delft: 15 February, 2012
Joint MSc Programme: Industrial Ecology
Interdisciplinary Project Group: 4413INTPGY

Universiteit Leiden

TU Delft
Delft University of Technology

Universiteit Leiden and TU Delft: 15 February, 2012
Joint MSc Programme: Industrial Ecology
Interdisciplinary Project Group: 4413INTPGY

Course Coordinator: Dr. ir. Gijsbert Korevaar
Primary Supervisor: Drs. ing. René Kleijn
Secondary Supervisor: PhD Researcher Christopher Davis

MSc Researchers: Boudewijn Boon, Eric Johnson, Melanie Studer, George Tsalidis, Jeroen van Houten, Carmen Vercauteren

Cover Map Image: Adapted from Gringer (2010).

Contents

1. Introduction

This user manual serves to provide the reader with a quick way to reproduce the interactive visualizations presented on our website:

http://resourceatlas.tudelft.nl/

These visualizations are based upon different pieces of software and applications. In this context of our work we will refer to the different visualizations as 'visualization tools' The manual will offer guidelines for the following visualization tools:

- Chord Diagram
- Google Earth
- Google Motion Charts
- JFlowMap

We are not by any means experts on any of these visualization tools as we do not have backgrounds in computer programming. This manual simply serves to provide people with the steps we took to create the results we have presented on our web page.

Before going through the guidelines, some preliminary steps are necessary. For most of the visualization tools some data had to be processed and manipulated to create the correct input formats. These steps will be explained in section 4. For these steps and other data manipulation we used "R", a language used for statistical computing and graphics. We made use of a development environment called "RStudio" which works together with "R". We will shortly introduce some basics of RStudio in section 3 that might be useful for the subsequent sections. Section 2 will first explain some necessary preparations to be able to follow the steps in this manual.

2. General Preparations

In this user manual we will make use of a basic set of files which will be often referred to. These files can be databases, R codes or other file types of which some will be modified throughout the course of following the guidelines.

You will need these files for the guidelines in the sections 5-8. The files can be downloaded at the following URL:

http://resourceatlas.tudelft.nl/downloads/UM_files.zip

Unzip the "UM_files.zip" file to a desired location on your computer from which you will be working thoughout the entire manual. The unzipped folder is called the "UM_files" folder (as in User Manual files) and we will refer to this folder multiple times while going through the guidelines. The "UM_files" folder contains subfolders named after the visualization tools, each with its own set of files that will be used in the associated guidelines.

3. Introducing RStudio

RStudio is a development environment compatible with the R language. The interface provides an easier way to work with R due to several of its functions and components.

You can download the software at the following URLs:

- http://www.r-project.org/ (R)
- http://rstudio.org/ (RStudio)

First download and install R. After this download, install and run RStudio. You will get to see a window similar to that in Figure 1.

The interface shows four frames of which the top-left frame, called the source editor, is the only one that will be used in the manual. It will contain the source codes in which you will be making several modifications.

The "Run" button, which you will use often, is indicated by the red arrow.

These are the only basic instructions required to continue to the guidelines of the visualization tools.

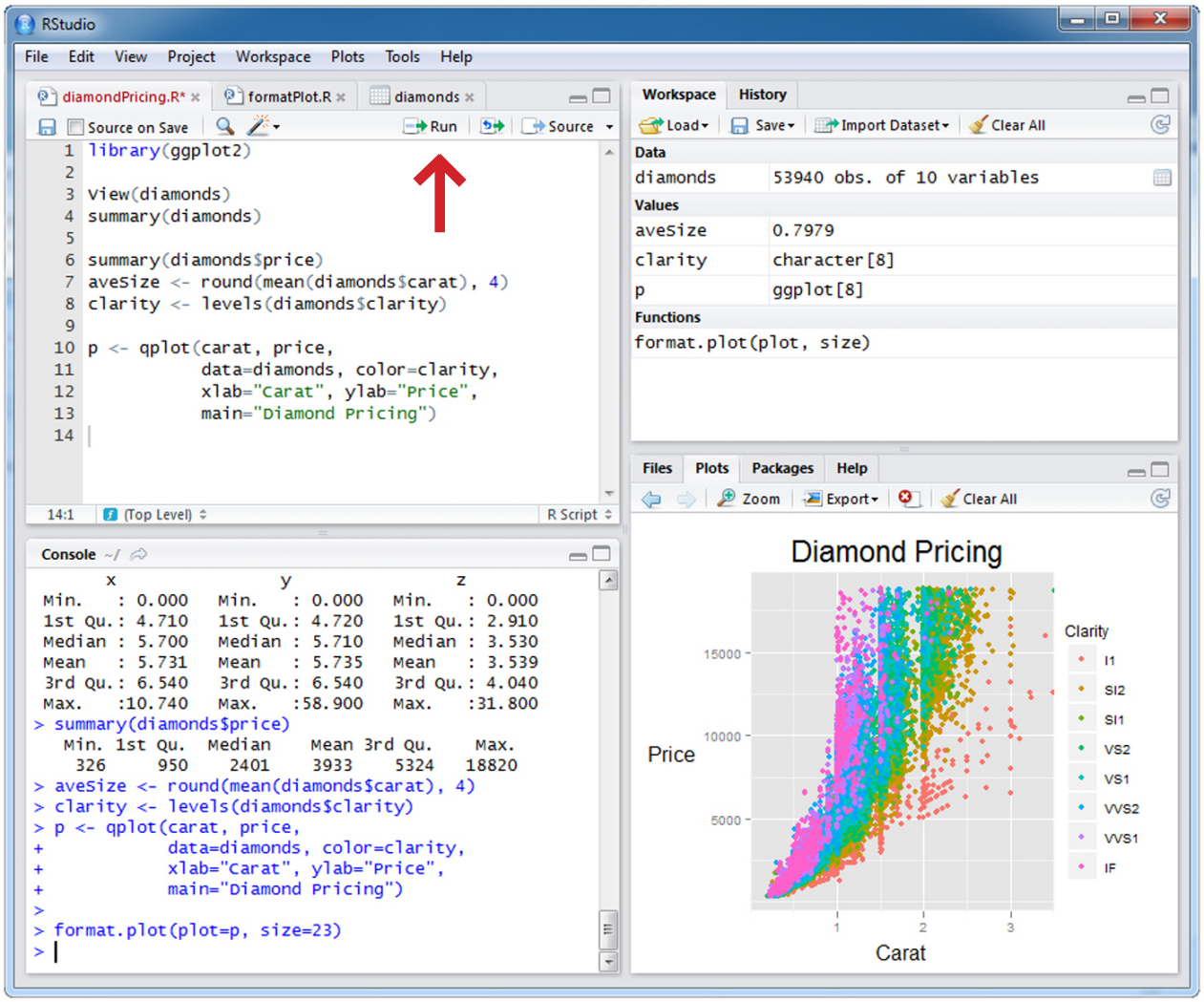


Figure 3.1: The RStudio interface

4. Data Collection & Processing

In this section a step-by-step description of the data collection and processing steps is given. The overall aim is to find appropriate and interesting data and to convert it into a format usable by our visualization tools.

4.1 Data Collection

Step 1: Define your aim and scope	
	The first step is to define the aim and scope of the project; This can be done by formulating research questions or hypotheses. By defining the aim and scope it will be easier to know what kind of data one should search for and what kind of data are important for the project. It provides the basis for the data search.

Step 2: Gain some background information	
Aim	Choosing a resource of your interest and identifying what data you need
How	1. Gain background knowledge on the resources of interest. For instance investigate on the resources life cycle in order to understand the types of resource flows (unrefined, refined, scrap, etc.) at stake and to understand whether the resources of interest is interconnected to other resources.
Tip	By gaining knowledge on the resources of interest, it will be easier to know what kind of data one should collect and what kind of data are important for the project.

Step 2: Gain some background information	
Aim	Choosing a resource of your interest and identifying what data you need
How	1. Gain background knowledge on the resources of interest. For instance investigate on the resources life cycle in order to understand the types of resource flows (unrefined, refined, scrap, etc.) at stake and to understand whether the resources of interest is interconnected to other resources.
Tip	By gaining knowledge on the resources of interest, it will be easier to know what kind of data one should collect and what kind of data are important for the project.

Step 3: Data collection	
Aim	Find a reliable data source and download datasets of interest
How	1. Search online for global statistical databases such as the United Nations Commodity Trade Database. One could also try to contact these via other ways than online. 2. If statistical databases do not provide the data sought, look at specific resources international groups and national geological surveys' publications and reports. To find such databases sources, you can make an Internet search but you also can look at the sources of figures and graphs found in papers about resources. 3. Download the dataset found in a format that is easily editable and convertable. For instance prefer .xls or .csv format to .pdf format.
Tips	1. The United Nations Commodity Trade Database (UNComtrade) is a good place to start searching for datasets as it includes freely available datasets over a long period of time and both in mass and in monetary value for an extensive list of commodities. The database is accessible at **http://comtrade.un.org/db/default.aspx** As we experienced difficulties to navigate the UNComtrade database and extract relevant datasets, we provide here below detailed guidelines on this process. a. Go to **http://comtrade.un.org/** b. Select *Database*. On this page you can already search in the database, but you are not able to download any datasets from here. c. Go to *Data Query > Basic Selection*. d. There is a lot on this page. What is most important is the field: *Step 2, Enter Selection Items > Step 2 Select Items*. Here you can either search on the basis of terms such as copper and nickel, but you can also use the commodity code numbers. A list on these commodities can be downloaded. Go to the top tab *Fast Tracks> Commodity List*. e. When you have searched for an item (on the page Data query>Basic Selection), try to unfold the founded commodities. If you choose items given a four digit code (i.e. the next category under the main groups) you will be able to get much more detailed datasets (i.e. also in mass units). f. There are multiple tabs in the field *Step 2. Enter Selection Items*. You will not always be able to fill in or make selection in all these. Still it is advised to filter the database as much as possible as you will only be able to download your dataset if it contains less then 50.000 records.

Step 3: ... continued

Tips

g. Because you need less than 50.000 records, you might have to repeat the filtering process or selecting in general a couple of times before you have all the datasets you want.

h. When you have made your selection, *Submit your query* and be very patient; it takes some time.

i. On the next page you will be able to *download the query* (if there are less than 50.000 records).

2. Specific resources international (study) groups and national geological surveys cooperate with national statistical office and often possess large databases. The US geoglogical survey (USGS) and the British geoglogical survey (BGS) have large databases for mineral resources accessible at **www.usgs.gov/** and **bgs.ac.uk/**. However these types of databases are in general not freely available. You can always try to approach them though to discuss about a possible collaboration.

3. National and/or continental statistical office, such as Eurostat, can also be rich in resources data flows. However collecting datasets from such offices makes the data processing more difficult since compiling data from different sources is not straightforward. Indeed, the reporting standards and norms (e.g. country names, units, resoruces categorization, etc.) might differ from one national/contienental statisical office to another.

Step 4: Data quality control

Aim

1. Check that the dataset collected is sufficient, relevant, valid and reliable[1,2]. The datasets should be:
   - Sufficient enough to support the final results.
   - Relevant, i.e. in a logical relation with the aim of the project.
   - Valid in terms of arithmetical units, classification etc.
   - Reliable in terms of its source and the way it was manipulated (unit transformation, calculations, etc.)

How

1. Check whether the database/data set contains all the information you need to answer your research questions. If not you need to make a judgment on whether it will be possible to get this from another source without this compromising in compatibility, or that you will need to look for a new database or data set.

1. Morgen, S.L. (2004). Guidance on Testing Data Reliability. Austin, Texas.
2. Crispieri, G. (2008). Data Quality Evaluation Methods. International SEMATECH Manufacturing Initiative.

Step 4: ... continued	
How	2. Check whether the dataset collected is consistent and coherent. For instance check : a. Whether the units are consistent b. Whether the number annotation is consistent (for instance the decimal separator should be the same everywhere) c. Whether the country names are are consistent d. Whether the classification/categorization is consistent (this applies only in the case of the use of different data sources) e. Whether the imports and exports match more or less. f. How non available data is reported (for instance, sometimes database use zero to report both non available data and data which are really equally to zero) 3. Check whether the data source is transparent about how the dataset was collected and calculated and check whether this has been done in a coherent and consistent way.
Tip	A more in dept (but more time consuming as well) quality check can be done by comparing several databases with each other and/or by comparing datasets in mass with data in economic value.
Note	Often in import-export datasets, the imports and exports do not match perfectly. This happens because of different ways of reporting or because of slight miscalculations.
Warning	Data collection via the several national statistical offices might cause misunderstandings with the material categories that the statistical offices use when they are developing their databases. Sometimes it might be possible to overcome this if it is just a different naming of the categories, or if these are aggregated of other categories. For this a better understanding of the resource (as in life-cycle knowledge and processing of the resource at stake) is needed.

Step 5: Data terms of use and copyright control!	
Aim	Ensure you are allowed to use and publish the data collected!
How	Search for the terms of use and copyrights of the data collected and ensure you comply with them.

4.2 Data Processing

The processing and composing of data files described in this section form a good basis to start using data in visualization tools. However for some visualization tools further processing is needed and will be described in later sections.

Required software:
- Spreadsheet software (e.g. Excel)
- Text editor or WordPad
- Google Refine
- RStudio

Step 6: Data processing

Aim	Process the data into a file format (csv) and structure that can be used by the visualization tools. More precisely, here we want to convert the data into 2 separate files (see Figure 4.1 and Figure 4.2) Figure 4.1: A first csv file containing the resource flow data (origin, destination, flow magnitude, flow type and year) in a table structure. Figure 4.2: A second csv file containing country information (Country code, country name, longitude, lattitude and possibly also altitude, capital and continent). The country codes and country names are based here on the ISO2 standards. This file should again be in a table structure.

Origin	Destination	FlowType	FlowMagnitude	Year
AL	BG	74	11911401	2010
AL	CA	74	4610	2010
AL	DE	74	263815	2010
AL	GR	74	4636080	2010
AL	IL	74	44362	2010
AL	IT	74	15800204	2010
AL	JO	74	311740	2010
AL	RS	74	10802	2010
AL	IN	74	103134	2010
AL	TR	74	30834	2010
DZ	BE	74	792975	2010
DZ	CA	74	603	2010
DZ	CV	74	182	2010
DZ	CN	74	194174	2010
DZ	FR	74	60914	2010
DZ	DE	74	322	2010
DZ	MT	74	3235	2010
DZ	CH	74	177729	2010
DZ	TN	74	122094	2010
DZ	TR	74	1320691	2010
AZ	FR	74	305	2010
AZ	GE	74	261025	2010
AZ	DE	74	247800	2010
AZ	KG	74	156	2010
AZ	NO	74	950	2010
AZ	RU	74	39146	2010
AZ	SG	74	44	2010
AZ	AE	74	290706	2010
AZ	TR	74	7921659	2010
AZ	GB	74	508	2010

Figure 4.1: Table with resource flow data

Country	CountryCode	Capital	Longitude	Latitude	Continent
Afghanistan	AF	Kabul	69,1952	34,5155	Asia
Albania	AL	Tirana	19,8172	41,3317	Europe
Algeria	DZ	Algiers	3,0597	36,7755	Africa
American Samoa	AS	Pago Pago	-170,7009	-14,2793	Oceania
Andorra	AD	Andorra la Vella	1,5218	42,5075	Europe
Angola	AO	Luanda	13,2306	-8,8159	Africa
Anguilla	AI	The Valley	-63,0669	18,2249	North America
Antarctica	AQ	Antarctica	0,071389	-57,459436	Antarctica
Antigua and Barbuda	AG	Saint John's	-61,8456	17,1175	North America
Argentina	AR	Buenos Aires	-58,4173	-34,6118	South America
Armenia	AM	Yerevan	44,509	40,1596	Asia
Aruba	AB	Oranjestad	-70,0265	12,5246	North America
Australia	AU	Canberra	149,1286	-35,282	Oceania
Austria	AT	Vienna	16,3728	48,2092	Europe
Azerbaijan	AZ	Baku	49,8932	40,3834	Asia
Bahamas	BS	Nassau	-77,339	25,0661	North America
Bahrain	BH	Manama	50,5354	26,1921	Asia
Bangladesh	BD	Dhaka	90,3978	23,7106	Asia
Barbados	BB	Bridgetown	-59,6105	13,0935	North America
Belarus	BY	Minsk	27,5766	53,9678	Europe
Belgium	BE	Brussels	4,3676	50,8371	Europe
Belgium-Luxembourg	BEL	Brussels	4,3676	50,8371	Europe
Belize	BZ	Belmopan	-88,7713	17,2534	North America
Benin	BJ	Porto-Novo	2,6323	6,4779	Africa
Bermuda	BM	Hamilton	-64,782	32,293	North America
Bhutan	BT	Thimphu	89,673	27,4405	Asia
Bolivia	BO	La Paz /Sucre	-65,2559	-19,0421	South America
Bosnia Herzegovina	BA	Sarajevo	18,4214	43,8608	Europe
Botswana	BW	Gaborone	25,9089	-24,657	Africa
Brazil	BR	Brasilia	-47,9292	-15,7801	South America

Figure 4.2: Table with country information

Step 6: ... continued

How	N.B.: The data processing depends on the file format (xls, cs, pdf, etc.) and the structure (list, symetric matrix, non-symetric matrix) of the data sets collected. Here we described the data processing steps starting form the data format and structure we collected, ie. a table in .csv format. Our datasets were gathered from the UNComtrade database. In these datasets, string variables are coded, eg. country names and commodity (resources flow types) names, are represented by code numbers (see Figure 4.3). The description of the codes used in the UNComtrade databases can be found on the website of the UNComtrade.

Year	Reporter Code	Trade Flow Code	Partner Code	Classification	Commodity Code	Quantity Unit Code	Supplementary Quantity	Netweight (kg)	Value	Estimation Code
2007	36	2	0	H3	7401	8	511520	511520	2340913	0
2007	36	2	608	H3	7401	8	511520	511520	2340913	0
2007	40	2	0	H3	7401	8	2323201	2323201	3862187	6
2007	40	2	251	H3	7401	8	300	300	4633	0
2007	40	2	381	H3	7401	8	567000	567000	1438596	0
2007	40	2	703	H3	7401	8	1755900	1755900	2418956	0
2007	40	2	792	H3	7401	8	1	1	1	6
2007	56	2	0	H3	7401	8	1103676	1103676	2917963	0
2007	56	2	251	H3	7401	8	68935	68935	291910	0
2007	56	2	276	H3	7401	8	984588	984588	2530017	0
2007	56	2	442	H3	7401	8	1218	1218	9229	0
2007	56	2	620	H3	7401	8	1	1	11	0
2007	56	2	688	H3	7401	8	43480	43480	58469	0
2007	56	2	724	H3	7401	8	2	2	47	0
2007	56	2	752	H3	7401	8	5450	5450	28243	0
2007	56	2	826	H3	7401	8	2	2	36	0
2007	100	2	0	H3	7401	8	6334977	6334977	10295766	0
2007	100	2	56	H3	7401	8	519904	519904	607139	0
2007	100	2	124	H3	7401	8	35255	35255	19313	0
2007	100	2	156	H3	7401	8	3982722	3982722	4716949	0
2007	100	2	251	H3	7401	8	556430	556430	1589687	0
2007	100	2	699	H3	7401	8	1240666	1240666	3362679	0
2007	124	2	0	H3	7401	8	24313460	24313460	166698447	0
2007	124	2	36	H3	7401	8	15390	15390	70896	0
2007	124	2	579	H3	7401	8	24139620	24139620	165995443	0
2007	124	2	842	H3	7401	8	158450	158450	632109	0
2007	152	2	0	H3	7401	8	15850523	15850523	84183120	0
2007	152	2	56	H3	7401	8	66003	66003	269859	0
2007	152	2	100	H3	7401	8	10024190	10024190	53547005	0
2007	152	2	251	H3	7401	8	21043	21043	120192	0
2007	152	2	276	H3	7401	8	5010170	5010170	26309594	0
2007	152	2	604	H3	7401	8	729117	729117	3936469	0

Figure 4.3: UNComtrade data table

Step 6: ... continued

How

1. Convert the country codes and the commodity codes into names (strings) in order to make the dataset more easily understandable. If one is experienced with R language, one could use a R code to do this. Here, as we were more familiar with Excel, the VLOOKUP formulas in Excel were used to replace the codes with the corresponding names (strings). You can use a VLOOKUP formula if you want to look up a value (in this case a code) in a given array and find the matching strings/value (in this case the names). For more details on how to use VLOOKUP formulas, check

   http://www.timeatlas.com/5_minute_tips/general/learning_vlookup_in_excel

2. For further processing the data you can use Google Refine. Often databases are too big for spreadsheet software. Therefor Google refine provides an easy accessible and free tool to further process it, for instance to scan mistakes and filter out the needed data.

3. Because the data processing steps can vary a lot depending on what software you use and what you want as an outcome yourself, we made a summary of the conversions and processing we applied to the UNComtrade data:

   a. Country codes were defined according to ISO2 standard

   b. Former countries received the same country code as the current countries they have become (for instance former East and West Germany and current Germany)

   c. Countries (most often small Islands) that are not independent and did not have there own country code were left out the dataset.

   d. 'not else where specified', special zones, bunkers and free zones were filtered out.

   e. Also 'world' as an entity is left out, this is the aggregation of all the flows from one country, but could not be visualized in our tools.

   f. Separate files were created for monetary trades flows and mass trade flows

Note

There are several option to convert or transform datasets. One can use software as different as R, Microsoft Excel, Openoffice Calc or Google Refine. All these softwares have their own advantages and disadvantages; R has lots of opportunities, but requires knowledge and experience on coding, Microsoft Excell known and widely used, but is not freely available, Openoffice then is freely available but is often more limited in functions, Google Refine is a good tool to explore and filter databases however it is not that straightforward to convert, calculate and transform datasets.

Warning

If you have data in a pdf format (e.g. tables of data in a pdf report), a PDF converter software can be used to convert it to xls or xlsx format. There are several open source pdf converters available. For example "Total PDF Converter" (available at **http://www.coolutils.com/TotalPDFConverter**) for windows users. For mac users there is an online PDF converter (available at **http://www.freepdfconvert.com/convert_pdf_to_source.asp**), however it is limited in amount and use.

5. Google Earth

Google Earth visualizes the earth in 3D, and provides satellite imagery, maps, terrain, 3D buildings and other features. Google Earth reads .kml files, which use XML syntax. These files can either be written directly using a text editor, or created in Google Earth. Although it is possible to create .kml files using Google Earth, it would be too time consuming to do so for large data sets. During the course of this process, RStudio was used to automate the production of .kml files.

These guidelines will lead you through this process. In short: Data in the form of .csv files will serve as input for RStudio which will then produce .kml files based on this data. These .kml files will be opened in Google Earth which will give a similar view as in Figure 5.

Almost no programming is required to get to these results with the exception of some minor modifications in the source code.

Required software:
- RStudio
- Google Earth

Step 1: First preparations	
Aim	Opening the R file and setting the working directory

How	1. Open the file "RCode_instr.R" located in the "Google Earth" folder which is a subfolder of the "UM_files" folder that was downloaded earlier.
	
	2. To be able to access .csv datafiles we need to set a working directory. If you look at the 5th line of the R code in the editor you will see:
	
	`setwd("")`
	
	3. In between the quotes the path to the working directory has to be placed, which is the path to "Files for RStudio & Google Earth" on your desktop. Paste the path in between the quotes similar to:
	
	`setwd("c:/Documents and Settings/Name/UM_files/Google Earth")`
	
	4. Now save the file under a different name, "`RCode_test.R`" in the same working directory. (*File > Save as...*).

Warning	When you set the working directory make sure to use slashes (`/`) and not the more often used backslashes (`\`)

Step 2: Creating the .kml file	
Aim	Creating the .kml file and opening it with Google Earth
How	1. In RStudio select the entire R code and run it by clicking the 'Run' button When RStudio has finished running the code the file "Dummy_Flows.kml" should now be in your working directory. 2. Open "Dummy_Flows.kml" in Google Earth and the result should show you something similar to Figure 2. below.

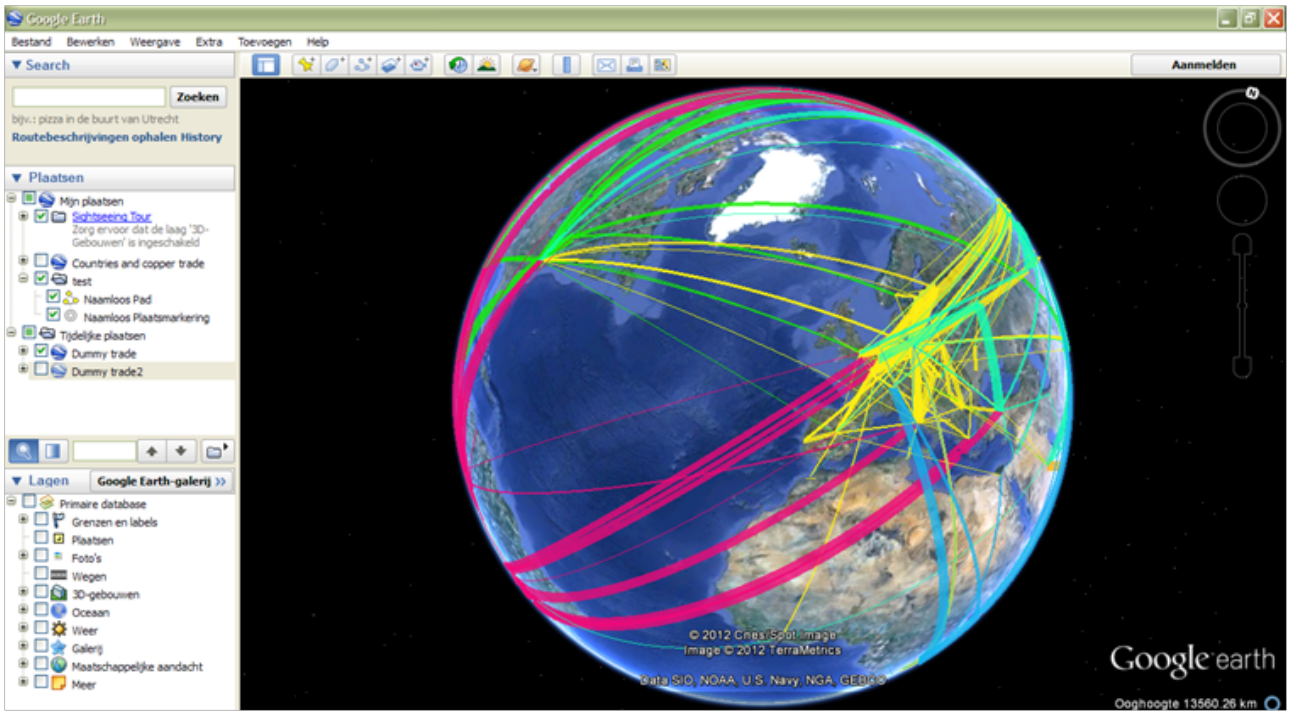


Figure 5.1: The RStudio interface

Step 3: Using other data

Aim	Knowing how to change the R code in order to use other .csv files as input
How	1. Go back to RStudio with the file "RCode_test.R" open. The R code refers only to the following two .csv files which can be seen in line 9 and 10 of the R code:

```
coordData = read.csv("Countries.csv", header=TRUE, strip.white=TRUE, sep=";")
data = read.table("Flows.csv", header=TRUE, strip.white=TRUE, sep=";")
```

2. As we have seen in Step 1 there is a "Flows2.csv" we we will now use. To select this file for use simply change "Flows.csv" (line 10) into "Flows2.csv" in the R code

3. Before we run the code we will change two things in the R code. First go to line 82 and 83 where you will see the following:

```
<Document>
      <name>Dummy trade</name>
```

This piece of code part of the data structure of the KML. Line 83 basically sais "Dummy trade" will be the name of the KML document.

4. Change "Dummy trade" into "Dummy trade2" to be able to distinguish the new KML from the earlier created KML in Google Earth.

5. The second thing to change is at line 270. You will see the following:

```
write(kmlText, file="Dummy_Flows.kml", append=FALSE)
```

6. Change "Dummy_Flows.kml" into "NewData_Flows.kml"

7. Run the code again by selecting the entire code and clicking the "Run" button. In your explorer you will now find "NewData_Flows.kml" in your working directory.

8. Open "NewData_Flows.kml" with Google Earth. You will probably see something similar as Figure 5.2.

As you can see there are not many flows shown in this file compared to the previous kml ("Dummy_Flows.kml") that was created, shown in Figure 5.1.

9. Go back to RStudio and go to line 25 of R code ("RCode_test.R"). You will see the following:

```
data <- data[with(data,which(data$FlowMagnitude >= 1000000)), ]
```

This command tells RStudio not to include flows below the magnitude of `1000000`.

10. Change the number `1000000` into `1000` and run the code again (Select entire code and push the "Run" button).

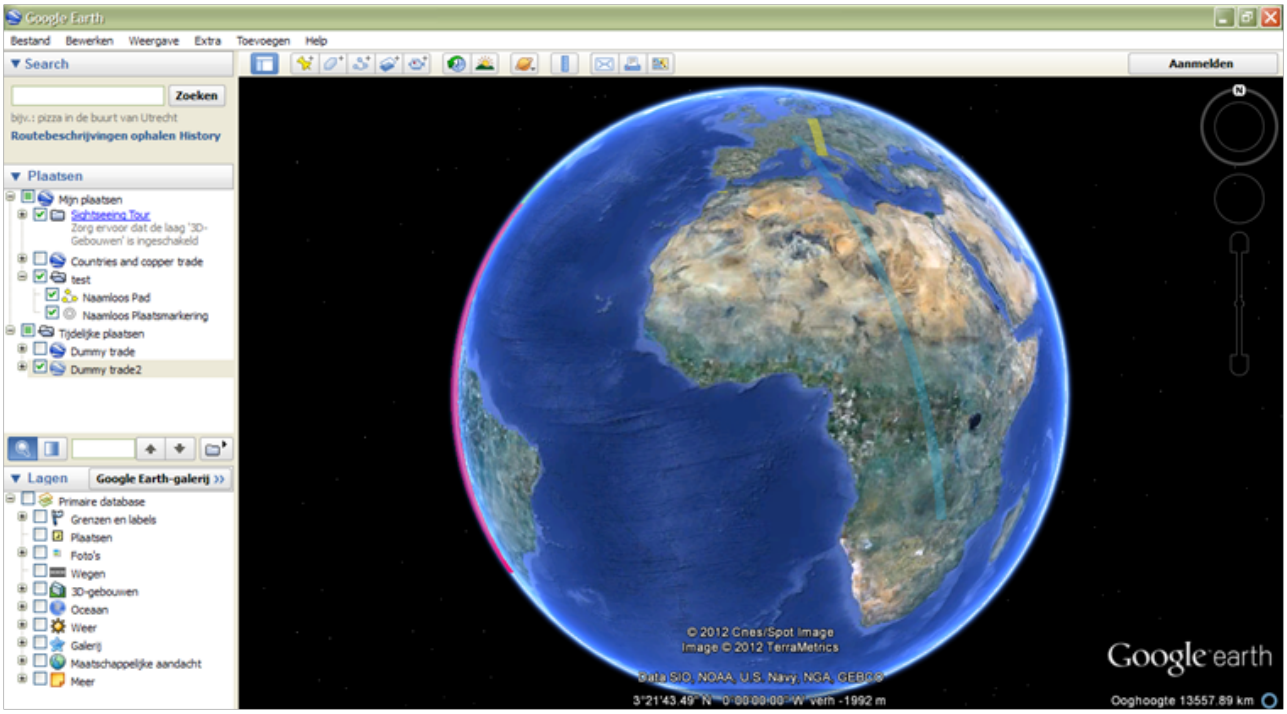


Figure 5.2: "NewData_Flows.kml"

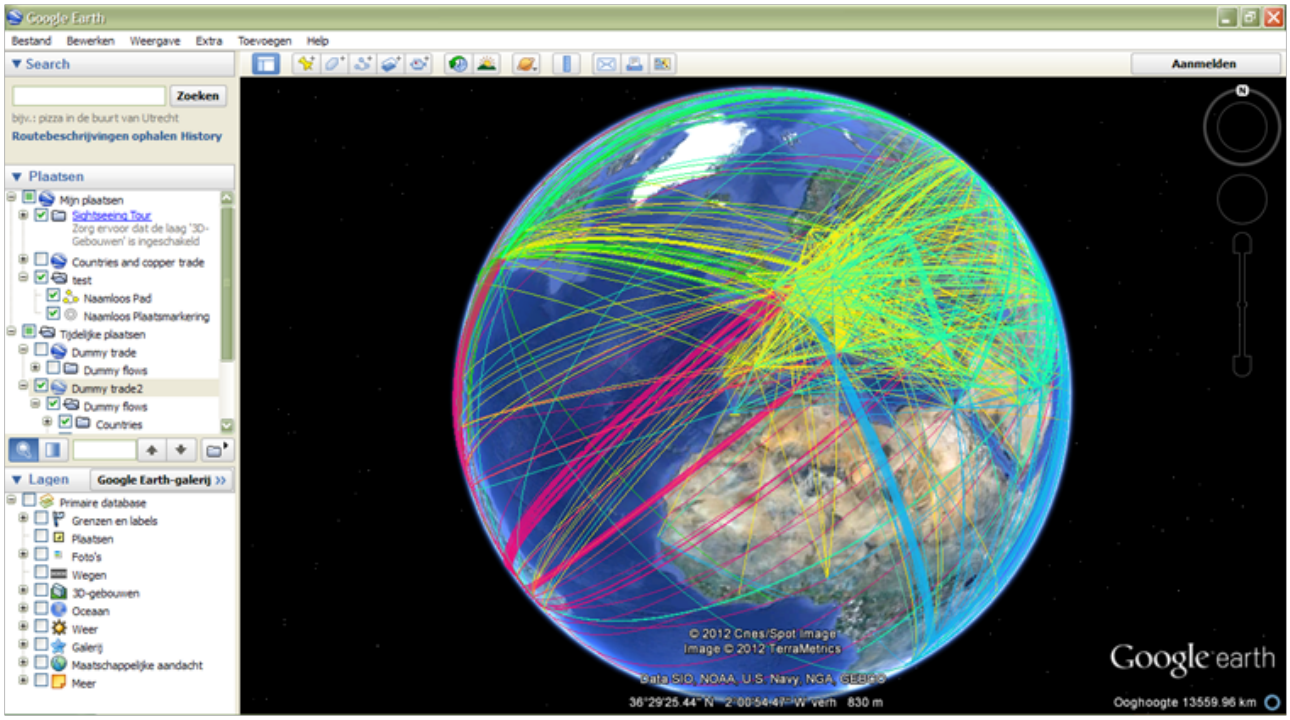


Figure 5.3: "NewData_Flows.kml" after lowering the magnitude threshold to 1000

Step 3: ...continued	
How	11. The file “NewData_Flows.kml” will be rewritten. Open it and click “Yes” when a window asks you if you want to reload the file. The result will be similar to what can be seen in Figure 5.3.
Note	There are more things you could alter within the R code. As you might have noticed there are comments throughout the file indicated by the ‘#’ symbol. This symbol basically tells RStudio not to see this piece of text behind it as a command. The comments in the file can help you indicate where things could be changed. A version of the R code without the comments is also provided: “RCode_clean.R”
Warning	If you want to use your own data make sure it is formated in the same way as “Flows.csv” and “Countries.csv”. This includes the following aspects • The column names should not defer from the original .csv files. This is because the R code specifically refers to these columns. • The country- and continent names of both files should be corresponding • Any errors such as “N.A.” in the .csv files could stop the final .kml from working

You have completed the Google Earth guidelines.

6. Google Motion Chart

Google motion chart is one of the chart types that Google Chart Tools offers. It can be freely inserted in Google Spreadsheet (an online spreadsheet similar to excel) as a gadget. Motion Charts allow the conversion of data-series in a dynamic flash-based chart. The motion chart can explore several indicators over time and be embedded in a website.
Below the steps to produce a Google motion chart are described. First the way the data should be formatted to fit the Google motion chart requirement is explained. Then we describe how to produce a motion chart.

Required software:

- Spreadsheet software (e.g. Excel)
- Google Spreadsheet
- Google Earth (http://www.google.com/earth/index.html)

N.B. A Google account is required for the use of Google spreadsheet!

Step 1: Getting your data in Google Spreadsheet

Aim

Producing a table suitable for Google Motion Chart

How

1. Open your spreadsheet software (e.g. Excel)
2. When using your own data structure it in a way similar to:

Country	Year	Continent	Mine production	Refined production	Refined consumption
Country 1	1999	Cont 1	Numerical value	Numerical value	Numerical value
Country 2	2000	Cont 1	Numerical value	Numerical value	Numerical value
.....	....	....	....	....	....

3. You can save your file in one of the following format .xls, .xlsx, .ods, .csv, .txt, .tsv or .tab.
4. We will be using “DummyData_CM.xls”. The file is located in the “Google Motion Chart” folder which is a subfolder of the “UM_files” folder that was downloaded earlier.
5. Select all data and copy it
6. Login to your google account and go to Google Docs
7. Click the *Create > Spreadsheet*
8. Paste the copied data in the spreadsheet

Step 2: Creating a Motion Chart	
Aim	Inserting the Motion Chart gadget in your Google spreadsheet file
How	1. Go to *insert > Gadget* 2. Click on *Charts* and scroll down untill you see *Motion Chart* and click on its associated picture. 3. A window will appear. Click on *Apply & close* 4. Your Motion Chart gadget will appear, looking similar to Figure 6.1. 5. To publish the gadget first click on the gadget. A dropdown menu called *Gadget* will appear. Use the menu to click *Publish gadget...* 6. A script will be given which can be embedded in a website.
Warning	If you use different variable for the x axis, the y axis and the size of the bubbles, be aware that if the value of any one of the variable equals “0” or “NA”, the country will not appear on the motion chart. For example, let’s say you have “mine production” as the size of the bubbles, “copper production” on the x axis and “refined copper consumption” on the y axis, if a country has zero “mine production”, it will not appear on the motion chart. A way to get around this is to convert the zeros to a minimum value (e.g. one) so that the countries still shows up in the graph. If you use this solution though you should notify it clearly in your visualization.

You have completed the Google Motion Chart guidelines.

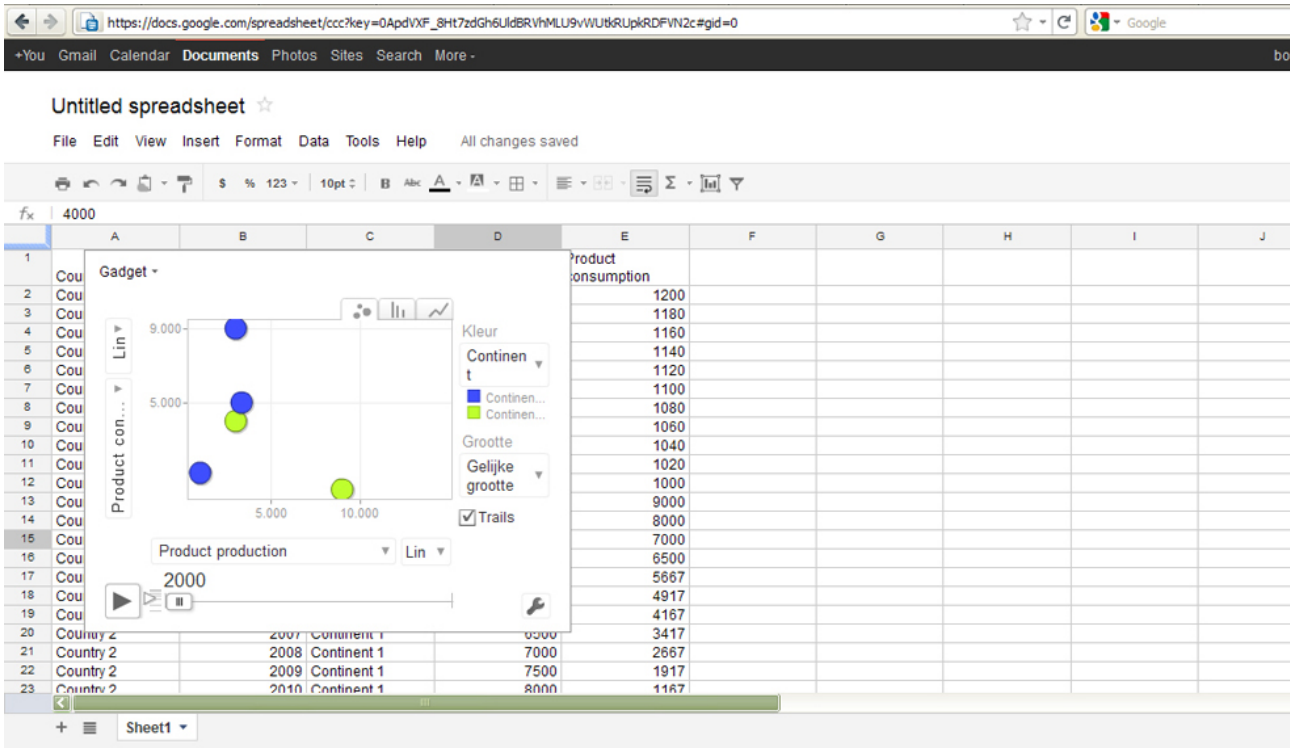


Figure 6.1: Google Spreadsheet with the inserted gadget “Motion Chart”

7. JFlowMap

JFlowMap is an open source graphical tool that was developed at the University of Fribourg in Switzerland. It offers various visualization techniques for producing and analyzing flow maps. The tool, a Java application, offers two separate views: a space-centric view which is a flow map representing flows for one year (Flowmap) and multiple years at once (Flownap Small Mulptiple), and a time-centric view which is an aggregated overview of the whole dataset represented in the form of a time line (Flowstrates).

The strength of JFlowMap is that it offers many interaction options, such as selection, zooming, filtering and animations, and it allows the representation of the temporal dimension. Moreover it requires little coding and little data processing and thus visualizations can be fairly quickly created.

To make a JFlowMap, one needs to make a simple .jfmv file that can be opened by the Java application Jflowmap.jar. Several files will be connected to each other and therefor we suggest to keep the folder structure as provided with the UM_files.zip.

The first two steps will entail data preparation. There are 2 files needed: one with information about the countries and the location of the anchor points (country capitals or country's central point) and one with the (resource) trade flows. They will get a specific place in the folder structure as well. Then we will adapt the flowmap files (.jfmv) to our data, which can than be opened by the java applet (jflowmap.jar).
Note: You can also use dummydata.csv and nodes.csv located in the "JFlowMap" folder which is a subfolder of the "UM_files" folder that was downloaded earlier. You can also use these files as a reference for structuring your own data

Required software:
- RStudio
- Reshape package (RStudio)
- Spreadsheet software (e.g. Excel)
- Text editor or wordpad

Step 1: Structure the resource flow data file

Aim

Producing a square wide formatted table with column names: origin, dest, year01, year02, year03, ..., yearXX, similar to:

```
Origin,De
st,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007
SEN,ESP,,,,1,1,,2,2,3,4,4,3,3,3,3,2,2,2
CHN,PER,,,,,,,,1,,1,1,1,1,1,2,2,2,2,
CHL,MOZ,,,,300,250,190,140,130,130,130,130,160,160,140,,,,,
ECU,DEU,,,,,,,,,,,,86,100,98,91,73,160
SRB,ISR,,,,,,,,,1,,,,,,,,
IRQ,HUN,,,,11,17,,,,,,201,258,700,938,1294,1252,1235,1272
URY,ARG,570,1050,,,,,,,,,,,,1,1,1,1
...
```

How

1. If your data is already in this structure you can skip this step. Otherwise we assume you have data in a table with column names: Origin, Dest and year. If there are more columns, for instance Units, Type, etc, you can remove these or filter them out by using the r-code in the next step. We also assume it is a csv files with ',' as separator. We refer to this file in the next steps as 'filename.csv'

2. In the folder "...UM_files\JFlowMap" you can find R file 'reshapeData.R'. Move this file to the same folder as your dataset 'filename.csv'

Step 1: ...continued

How

1. Open the r-code 'reshapeData.R' with RStudio. It should look like this:

```
#never ever ever convert strings to factors - we're not using factors in the data below
options(stringsAsFactors = FALSE)

library(reshape)

data = read.csv('filename.csv', sep=',')

data$Unit=NULL
data$FlowType=NULL

#we have the long format, want the wide format
betterData = reshape(data, idvar=c("Origin", "Dest"), timevar="Year", direction="wide")

#http://stackoverflow.com/questions/1296646/how-to-sort-a-dataframe-by-columns-in-r
sortedData = data[with(data, order(Year)), ]
betterData = reshape(sortedData, idvar=c("Origin", "Dest"), timevar="Year",
direction="wide")

write.table(betterData, file="data.csv", na="", sep=";", row.names=FALSE)
```

2. 'filename.csv' (in red) should refer to your dataset.
3. 'data.csv' (in red) refers to the new file you want to produce.
4. Make sure the column names (in green) match the column names of your data file.
5. The commands in orange refer to the extra columns 'Unit' and 'FlowType', these commands filter out these columns.
6. Run the code: Select the whole code and press *Run*

   You do not need RStudio after this

Tip	The R Script needs to be in the same folder as your table / data file. You could also provide the complete document address of your table in 'filename.csv' instead of just the name.
Warning	Be careful with the comma separators, make sure they match your data files. If you change something in the code, please check if you all the " and ' signs are present.

Step 2: Create nodes.csv

Aim	Create a new dataset with country or city information such as naming, longitude and latitude. It should look like this:
	`Code,Name,Lat,Lon`
	`LCA,Saint Lucia,13.903085,-60.9659`
	`BRN,Brunei Darussalam,4.581283,114.819152`
	`MDG,Madagascar,-18.054455,47.108621`
	`KNA,Saint Kitts and Nevis,17.313103,-62.736679`
	`UZB,Uzbekistan,41.447353,64.79929`
	`LSO,Lesotho,-29.595733,28.244114`
	`SLB,Solomon Islands,-8.910545,159.537743`
	`MDV,Maldives,3.353159,73.260862`
	`...`

How	1. If you are using Countries as your nodes (destination and origin) and Capitals as anchor points, then you can use the nodes.csv file in the "JFlowMap" folder located in the "UM_files" folder. Otherwise you need to structure your country information as in the example given above.

Step 3: Getting JFlowMap ready

Aim	Either using JFlowMap provided in the "UM_files.zip" or downloading the latest JFlowMap application from the web.
How	1. Either access JFlowMap by going to the folder "…\UM_files\JFlowMap\Dummy Project"; 2. Or download the latest version of JFlowMap at: **http://code.google.com/p/jflowmap/downloads/list** 3. If you created the .csv files in Step 1 and Step 2 (data.csv and nodes.csv), place them in the folder "Data".
Warning	If you downloaded the latest version of JFlowMap, make sure to check if the folder structure, as it might have changed.

Step 4: Create your own flow map	
Aim	Making a .jfmv file linked to your data which can be run by jflowmap.jar
How	1. In the folder "...UM_files\JFlowMap\Dummy Project\viewconf" you can find the file "flowmap.jfmv" 2. Open "flowmap.jfmv" with a text editor or WordPad 3. What is important are the 5th and 6th paragraph which should look similar to: 4. Make sure the separator matches your data files 5. Line 5 (in red) refers to your data file, if you created your own (step 1) then you need to change this to 'data.csv'. 6. The next lines (in green) should match the column names in your data files. 7. The last line (in orange) is where you can define the Unit

```
…
data=csv
data.csv.separator=;
data.csv.charset=utf-8
data.csv.nodes.src=../Data/nodes.csv
data.csv.flows.src=../Data/dummydata.csv

data.attrs.node.id=Code
data.attrs.node.label=Name
data.attrs.node.lat=Lat
data.attrs.node.lon=Lon
data.attrs.flow.origin=Origin
data.attrs.flow.dest=Dest
data.attrs.flow.weight.re=[0-9]{4}
data.attrs.flow.weight.legendCaption=Kg
…
```

Step 5: Launching the .jmfv files	
Aim	Opening the jmfv files, view your visualization and explore the possibilities
How	1. Open the jflowmap.jar file in the folder "…UM_files\JFlowMap\Dummy Project" This will run the java application 2. In the application open the "flowmap.jfmv" file located at "... UM_files\JFlowMap\ Dummy Project\viewconf. (*File > Open View > flowmap.jfmv*)
Warning	It could be that it gives an error. This could be due to a mistake in your csv files or due to a mistake in referring to files. Make sure that you used the correct comma separator as defined in the code, that every row has the same length and that the nodes.csv doesn't contain N/A's. A nice thing about JFlowMap is that it tells you where the mistake is: for instance it will tell you what he can't find and on which row there is a mistake. You will need to trace these back in your data files.

Step 6: Changing the settings	
Aim	Changing default settings and add functions
How	1. Re-open "flowmap.jfmv" with a text editor or WordPad 2. To change the function: Delete the line "`window.settings.showTabs=Edge bundling,Animation,Filter`" so that there is no selection made of which functions to show. 3. In the application re-open the "flowmap.jfmv" file in JFlowMap (see Step 4). You will see that there are more functions. 4. To make a selection you need to add the line "`window.settings.showTabs=Edge bundling,Animation,Filter`" again. After the equation mark you can add the functions you like to show, leave out those that you do not want to show. 5. To change the default setting: You can play with the default settings by changing some of the numbers and *TRUE/FALSE* parts in the first 2 paragraphs. 6. More explanation on what each line represents can be found in the "Script with explanations.rtf" file located in the "...UM_files\JFlowMap" folder.
Warning	It could be that it gives an error. This could be due to a mistake in your csv files or due to a mistake in referring to files. Make sure that you used the correct comma separator as defined in the code, that every row has the same length and that the nodes.csv doesn't contain N/A's. A nice thing about JFlowMap is that it tells you where the mistake is: for instance it will tell you what he can't find and on which row there is a mistake. You will need to trace these back in your data files.

(Additional) step 7: Small multiples and Flowstrates	
Aim	Create Small Multiples and Flowstrates
How	1. Repeat Step 4, but open flowmap-sm.jfmv or flowstrates.jfmv instead 2. You will find the same paragraphs in these files for which you need to ensure they match you data files 3. Also repeat Step 5 but now open the “flowmap-sm.jfmv” or “flowstrates.jfmv” file.

You have completed the JFlowMap guidelines.

8. Chord Diagrams

A chord diagram is a type of visualization offered by D3, a free JavaScript library for the manipulation of documents based on data. Chord diagrams show directed relationships within a group of entities.

Chord diagrams are based on 3 different types of files: (1) a HyperText Markup Language (html) file for webpage content, (2) a Cascading Style Sheets (css) file for appearance and layout, (3) and a javascript (js) file for interaction. They require fairly complex coding but as the as the adage goes "never program what you can steal". Fortunately, the code for chord diagrams has already been written and is shared online. The trick is to "steal" the code and then adapt it.

Below the steps to produce a chord diagram are described. In brief, first the input data has to be formatted to fit with the chord diagram. Then the code to create chord diagrams has to be adapted to the input data. Finally, the chord diagram can be opened on a web browser.

Required software:
- Spreadsheet software (e.g. Excel)
- Text editor or wordpad
- Web browser
- Text editor or wordpad

Step 1: Strucuring the resource flow data

Aim

From the data structure created at the data collection and processing step (or from a dummy file), produce a square matrix of numbers written in a scripting style as below and representing resources imports and exports per continents.

```
[20,2,22,0,0,0],
[0,65,95,0,0,0],
[111,333,1547,0,0,0],
[0,481,29,333,1,2],
[0,0,0,0,0,0],
[0,676,1399,804,582,0]
```

How

1. Open either the resource flow data file you created in section "4.2 Data processing" or open the dummy file "Dummy_data_file_1.xls". Make sure to only include three columns such as shown below:

OriginContinent	DestinationContinent	FlowMagnitude
South America	Europe	1289
South America	Oceania	582
South America	North America	439
North America	Asia	404
South America	North America	350
South America	Asia	343
Europe	Europe	338
Europe	Europe	274
Europe	Europe	232
Europe	Europe	225
South America	Asia	219
North America	North America	219
Europe	Asia	210

2. Produce a symmetric matrix where rows represent OriginContinent and columns represent DestinationContinent and the number of the matrix represent the Flow Magnitude like in the figure below. *Spreadsheet software > Insert > PivotTable*.

   These steps might differ per spreadsheet software. The result should be similar to Figure 8.1.

Somme sur FlowMagnitude	DestinationContinent					
OriginContinent	Africa	Asia	Europe	North America	Oceania	South America
Africa	20	2	22	0	0	0
Asia	0	65	95	0	0	0
Europe	111	333	1547	0	0	0
North America	0	481	29	333	1	2
Oceania	0	0	0	0	0	0
South America	0	676	1399	804	582	0

Figure 8.1: Pivot table format in spreadsheet software

Step 1: ...continued

How

3. Export the symmetric matrix in a csv format. Spreadsheet software > save as > .csv. Open the .csv in a text editor (e.g. WordPad) and it should look similar to:

```
Somme sur FlowMagnitude,DestinationContinent,,,,,
OriginContinent,Africa,Asia,Europe,North America,Oceania,South America
Africa,20,2,22,0,0,0
Asia,0,65,95,0,0,0
Europe,111,333,1547,0,0,0
North America,0,481,29,333,1,2
Oceania,0,0,0,0,0,0
South America,0,676,1399,804,582,0
```

4. Manually remove texts (e.g. the first two lines and all continent names). Then add squared brackets ([ ]) so that it will be exactly the same as below. You will later need to copy this piece of syntax.

```
[20,2,22,0,0,0],
[0,65,95,0,0,0],
[111,333,1547,0,0,0],
[0,481,29,333,1,2],
[0,0,0,0,0,0],
[0,676,1399,804,582,0]
```

Tip

The chord diagram crashes when the numbers in the input matrix are too complex. To overcome this problem, round up your numbers. If you have data in kilograms, divide them by 1000 to obtain data in Ktons and remove decimals.

Warning

Be careful that your input matrix is symmetric and separated by commas (rather than semi columns for instance).

Note

This whole step aims to produce manually a matrix of numbers that can be copy-paste in the chord javascript code (.js file) that will create the Chord diagram. However in principle, it should be possible to make the javascript code read an external csv or a json file as opposed to inserting directly the matrix of data in the javascript code. Thus in principle this whole step (ie. the production of a matrix of numbers in the above format) could be avoided if you have the appropriate javascript code (i.e. a code that reads an external csv or a json file of data). Unfortunately, due to the restricted amount of time, we use the supoptimal yet easier solution of copy-pasting the matrix of data in the javascript code.

Step 2: Prepare the .js file	
Aim	Preparing the “File_Name_1.js” file
How	1. Access the “Chord Diagram” folder located in the “UM_files” folder. 2. Open the “File_Name_1.js” file with a text editor. It should look similar to Box 8.1 (on the next page). If not, try a different text editor. 3. The “File_Name_1.js” file will contain data similar to that in Box 8.1. Replace the pieces of syntax in “File_Name_1.js” with the corresponding pieces of syntax indicated in red in Box 8.1 (Note that one of these pieces of syntax is the earlier created matrix, you can copy paste this now) 4. Save and close the file
Note	The .css file does not require any change (except if you want to change the appearance and layout of the chord diagrams).

Step 3: Prepare the .html file	
Aim	Preparing the “chord.html” file so that it refers to the .js files
How	1. Open the “chord.html” file in the “Chord Diagram” folder with a text editor and adapt it to your data in way resembling Box 8.2 (page 30). 2. Save and close the file
Tip	Keep the .js, .css and html files in a single folder as these files refer to one another.

Step 4: Admire the result	
Aim	Open the chord diagram in explorer
How	1. To view the chord diagram open “chord.html” in a web browser

You have now completed the Chord Diagram guidelines and, thereby, you have completed the complete User Manual.

Box 8.1: "File_Name_1.js" (colors added)

```
// From http://mkweb.bcgsc.ca/circos/guide/tables/

// Set the chart dimensions.
// "w" = width
// "h" = height
// "r0" =  inner radius
// "r1" = outer radius
var w = 400,
    h = 400,
    r0 = Math.min(w, h) * .30,
    r1 = r0 * 1.1;

// Create the chord.
// "padding" = the space between the variables (for instance the continents) on the
chord arc.
// "matrix" refers to your input data, which must be in a symmetric matrix of numbers.
var chord1 = d3.layout.chord()
  .padding(.05)
  .matrix([
    [1959,248,2225,0,0,0],
    [16,6524,9475,4,1,1],
    [11111,33281,154710,29,11,46],
    [3,48059,2885,33270,138,174],
    [0,0,0,0,0,0],
    [0,67636,139887,80370,58235,0]
   ]);

// Labels the variables. The order of the variables should follow the order of vari-
ables in the matrix.
var labels = new Array(
    "Africa","Asia","Europe","North America","Oceania","South America"
  );

// Set the colors in HEX format (i.e. web color codes). The order of the color corre-
sponds to the order of the labels set above.

// You need to choose a palette of colors to represent your variables (for instance
the different continents). To find an appropriate color palette, check http://www.col-
orbrewer2.org/

// Here Dark blue (Africa) = #00008b, Light blue (Asia) = #5cacee, Dark green (Europe)
= #006400, Light green (North America) = #7cfc00, Yellow (Oceania) = #ffff00, Orange
(South America) = #ff8f00
var fill = d3.scale.ordinal()
  .range(
    ["#00008b","#5cacee","#006400","#7cfc00","#ffff00","#ff8f00"]
  );

// Define the attributes of the chord diagram
var svg = d3.select("#chart1")
  .append("svg:svg")
    .attr("width", w)
    .attr("height", h)
  .append("svg:g")
    .attr("transform", "translate(" + w / 2 + "," + h / 2 + ")");

// Define the style and appearance of the chord arc
// "fade" = function that makes the rest of the chord fade when one variable is se-
lects on the chord arc.
svg.append("svg:g")
  .selectAll("path")
    .data(chord1.groups)
```

```
  .enter().append("svg:path")
    .style("fill", function(d) { return fill(d.index); })
    .style("stroke", function(d) { return fill(d.index); })
    .attr("d", d3.svg.arc().innerRadius(r0).outerRadius(r1))
    .on("mouseover", fade(.1))
    .on("mouseout", fade(1))
      .append("svg:title")
      .text(function(d) {return labels[(d.index)]});


// Define the style and appearance of the inside of chord
svg.append("svg:g")
    .attr("class", "chord")
  .selectAll("path")
    .data(chord1.chords)
  .enter().append("svg:path")
    .style("fill", function(d) { return fill(d.target.index); })
    .attr("d", d3.svg.chord().radius(r0))
    .style("opacity", 1);

// Define how the ticks on the chord arc are computed.
var ticks = svg.append("svg:g")
  .selectAll("g")
    .data(chord.groups)
  .enter().append("svg:g")
  .selectAll("g")
    .data(groupTicks)
  .enter().append("svg:g")
    .attr("transform", function(d) {
      return "rotate(" + (d.angle * 180 / Math.PI - 90) + ")"
          + "translate(" + r1 + ",0)";
    });

// Define how the ticks should look like.
// "x1" = distance of the ticks from the chord arc
// "x2" = length of the ticks
ticks.append("svg:line")
    .attr("x1", 1)
    .attr("x2", 6)
    .style("stroke", "#000000");

// Define where the label of the ticks.
ticks.append("svg:text")
    .attr("x", 8)
    .attr("dy", ".35em")
    .attr("text-anchor", function(d) {
      return d.angle > Math.PI ? "end" : null;
    })

    .attr("transform", function(d) {
      return d.angle > Math.PI ? "rotate(180)translate(-16)" : null;
    })
    .text(function(d) { return d.label; });;

// Return an array of tick angles and labels, given a group.
function groupTicks(d) {
  var k = (d.endAngle - d.startAngle) / d.value;
  return d3.range(0, d.value, 25000).map(function(v, i) {
    return {
      angle: v * k + d.startAngle,
      label: i % 2 ? null : v/1000
```

```
    };
  });
}

// Returns an event handler for fading a given chord group.
function fade(opacity) {
  return function(g, i) {
    svg.selectAll("g.chord path")
      .filter(function(d) {
          return d.source.index != i && d.target.index != i;
      })
      .transition()
          .style("opacity", opacity);
          svg.selectAll("g.chord path")
      .filter(function(d) {
          return d.source.index != i && d.target.index != i;
      })
      .transition()
          .style("opacity", opacity);
          svg2.selectAll("g.chord path")
      .filter(function(d) {
          return d.source.index != i && d.target.index != i;
      })
      .transition()
          .style("opacity", opacity);
          svg3.selectAll("g.chord path")
      .filter(function(d) {
          return d.source.index != i && d.target.index != i;
      })
      .transition()
          .style("opacity", opacity);
  };
}
```

Box 8.2: “chord.html” opened in a text editor (colors added)

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <title>Webpage Title </title>
    <script type="text/javascript" src="http://mbostock.github.com/d3/d3.js?2.4.6"></
script>
    <script type="text/javascript" src="http://mbostock.github.com/d3/d3.layout.
js?2.4.6"></script>
    <link type="text/css" rel="stylesheet" href="chord.css"/>
  </head>
  <body>
     <div id="container"><center><h1>Container Title</h1></center>

    <div id="chart1"><center><h3>Chord Title 1 [Unit] </h3></center></div>
    <script type="text/javascript" src="File_Name_1.js"></script>

    <div id="chart2"><center><h3>Chord Title 2 [Unit]</h3></center></div>
    <script type="text/javascript" src="File_Name_2.js"></script>

    <div id="chart3"><center><h3>Chord Title 3 [Unit]</h3></center></div>
    <script type="text/javascript" src="File_Name_3.js"></script></script>
  </body>
</html>
```