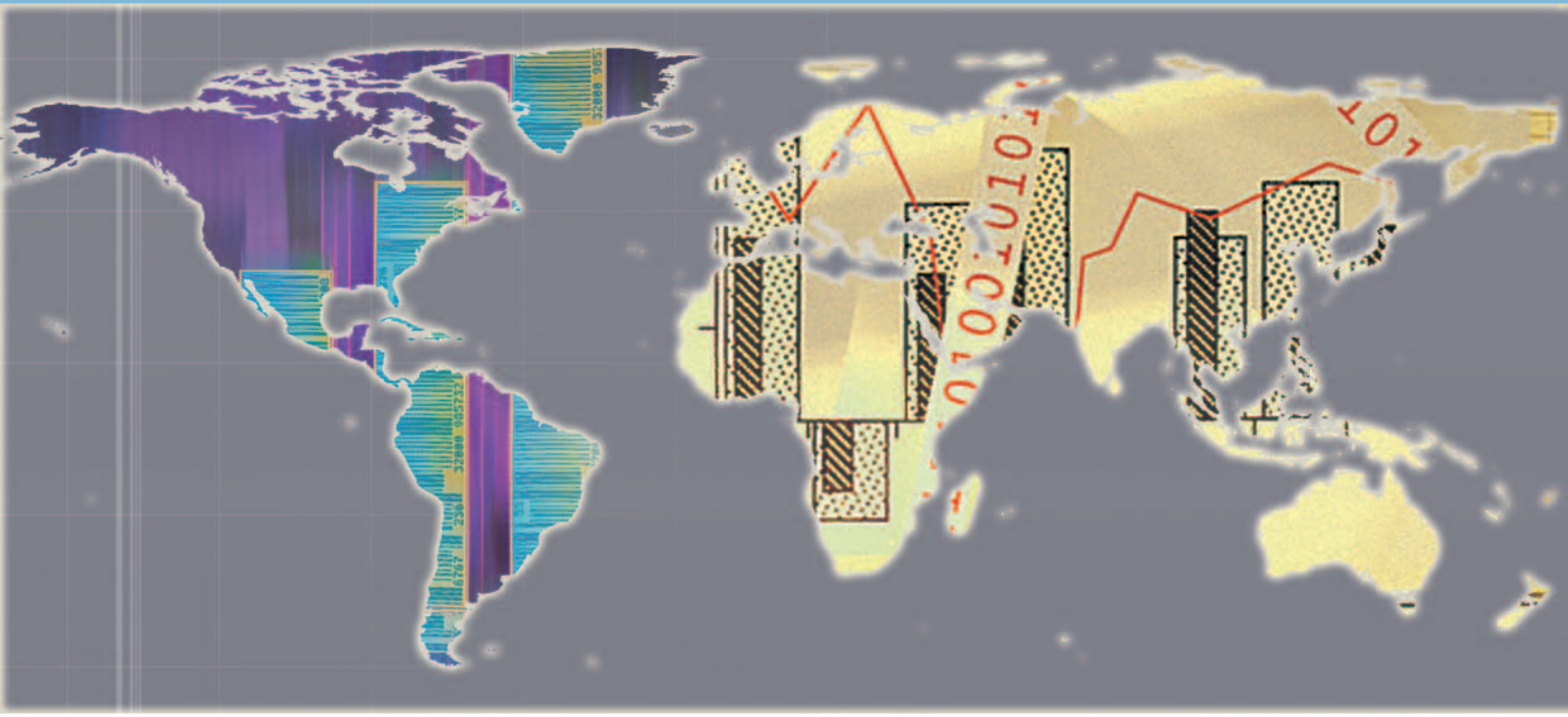
WORKING PAPER 03/2010



R in the Statistical Office: The UNIDO Experience

UNITED NATIONS
INDUSTRIAL DEVELOPMENT ORGANIZATION

DEVELOPMENT POLICY AND STRATEGIC RESEARCH BRANCH
WORKING PAPER 03/2010

R in the Statistical Office: The UNIDO Experience

Valentin Todorov

Development Policy and Strategic Research Branch
Regional Strategies and Field Operations Division
UNIDO

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna, 2011

Acknowledgements
We are thankful to Matthias Templ from the Technical University of Vienna and Statistics Austria, Ugyen Tenzin from the Ministry of Labour and Human Resources of Bhutan and the members of the Sharing Advisory Board (SAB) of the Conference of European Statisticians for provided information, comments and suggestions. We would like to express sincere appreciation to Rita Liang from RSF/DPR/STA for carefully reading the manuscript and suggesting many helpful corrections.

Contents

List of Tables

List of Figures

1 Introduction

The R language ((Ihaka and Gentleman, 2009) is a freely available environment for statistical computing and graphics. It can be used for handling and storage of data and performing of statistical calculations. R provides a number of tools for data analysis, and features excellent graphical capabilities included in a straightforward programming language. R is widely used for statistical analysis in a variety of fields and is backed up by a large number of add-on packages that extend the system. The R Core Team (`http://www.r-project.org/`) defines R as an *environment* rather than a statistical system or a programming language, as an integrated suite of software facilities for data manipulation, calculation and graphical display which includes:

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License (GPL) (`http://en.wikipedia.org/wiki/GNU_General_Public_License`) in source code form. R can be obtained as both source and binary (executable) forms from the Comprehensive R Archive Network (CRAN). The source files are available for a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux) as well as for Windows and MacOS for which are available also pre-compiled binary distributions of the base system and contributed packages. The most recent release of R is version 2.12.1 (released on December 16, 2010) and pre-release versions of 2.12.2 are in progress.

The statistical techniques available in R—linear and non-linear modeling, classical statistical tests, time-series analysis, classification, clustering, robust methods, etc.—as well as its data manipulation and presentation tools make this package an ideal integrated environment for both research and production in the field of official statistics. However its application in the National Statistical Offices and International Organizations is still quite limited. This is mainly due to the wide spread opinion that R is hard to learn compared to the other statistical packages, like SAS and SPSS and it is said to have a very steep learning curve. The lack of a true point-and-click interface adds to this (meanwhile the development goes on and already several packages are available providing graphical user interface for R). The technical documentation accompanying R and most of the add-on packages rarely include syntax examples related to the analytical methods applied in the official statistics.

The present overview of R focusses on the strengths of this statistical environment relevant for the typical tasks performed in the national and international statistical offices and its purpose is to demonstrate on examples from the UNIDO practise some of the advantages of R. The paper has an informative character rather than a tutorial form. Many aspects of the application of R in the statistical offices are considered but the details are left for another publication. A tutorial covering the discussed areas as well as other relevant for the official statistics topics with example data sets and code snippets is under preparation.

A wide variety of add-on functionality (actually the normal way of extending R) is available from the same web page in the form of contributed R packages, which can be downloaded in source form or installed directly from the R console by using the `install.packages()` function (provided the computer is connected to the Internet). A few examples relevant for national and international statistical organizations are: survey analysis (**survey**, **sampling**, **laeken**), handling of missing data (**VIM**, **mice**, **mi**, **mvnmle**, **mitools**, **EMV**, **mix**, **pan**), time series analysis (**stats**, **tseries**, **zoo**, **its**, **pastecs** and **lmtest**), robust statistics (**robustbase**, **rrcov**, **rrcovNA**, **robust**). The number of contributed packages at CRAN is increasing rapidly and sometimes it is difficult to find the appropriate package. To facilitate this, a number of "CRAN task views" are maintained that provide an overview of packages for certain tasks. Current task views include econometrics, finance, social sciences, robust and multivariate statistics. See http://CRAN.R-project.org/web/views/ for further details. A task view on Official Statistics is available at http://CRAN.R-project.org/view=OfficialStatistics. This CRAN task view contains a list of packages that include methods typically used in official statistics and survey methodology.

```
library("ctv")
x <- read.ctv(file.path(.find.package("ctv"), "ctv",
        "OfficialStatistics.ctv"))
x

CRAN Task View
--------------
Name:       OfficialStatistics
Topic:      Official Statistics & Survey Methodology
Maintainer: Matthias Templ
Contact:    templ@tuwien.ac.at
Version:    2010-10-12

Packages:   Amelia, cat, EVER, Hmisc, impute, ineq, laeken,
            lme4, memisc, mi, mice, micEcon, MImix,
            missMDA, mitools, mix, nlme, norm,
            odfWeave.survey, pan, pps, RecordLinkage,
            reweight, robCompositions, rrcovNA, sampfling,
            sampling, samplingbook, SDaA, sdcMicro,
            sdcTable, SeqKnn, simFrame, simPopulation,
            spsurvey, StatMatch, stratification, survey*,
            surveyNG, TeachingSampling, urn, VIM, x12,
            yaImpute
            (* = core package)
```

All packages related to this task view can be installed by

```
install.views("OfficialStatistics")
```

The content of this task view is reported in Appendix A.

More information on R can be found at the CRAN web site `http:\cran.r-project.org`. One could also read a brief overview of R from the standpoint of official statistics in the paper Todorov (2008). A comprehensive list of books on R can be found at `http://www.r-project.org/doc/bib/R-publications.html` like Maindonald and Braun (2007), Murrell (2005), Varmuza and Filzmoser (2009), Muenchen and Hilbe (2010) to mention a few.

In the following three sections three typical applications of the programming language and system R, based on examples from the statistical practice of UNIDO will be considered. In Section 2 the abilities of R to import and export data from and to different data formats and different statistical systems is considered. This features render it a very useful tool for facilitating the collaboration in the statistical office. The second example application described in Section 3 considers the graphical excellence of R as applied to the statistical production process of UNIDO for generating publication quality graphics included in the *International Yearbook of Industrial Statistics*. The graphics together with the related text are typeset in LaTeX using the R tool for dynamic reporting `Sweave`. The third example presented in Section 4.1 illustrates the analytical and modeling functions available in R and add-on packages. These are used to implement a nowcasting tool for the Manufacturing Value Added (MVA) necessary for generating estimates to be published in the *International Yearbook of Industrial Statistics*. Functions from the package for robust statistics **robustbase** are used for this purpose. Section 5 summarizes and briefly describes other applications and topics that are not covered here, but will constitute the content of future work.

All the data sets discussed throughout the paper, which are mainly derived from the UNIDO databases as well as selections from other sources, and all the R code used are provided in an R package called **ritso** which accompanies the work. The package will be soon available on CRAN.

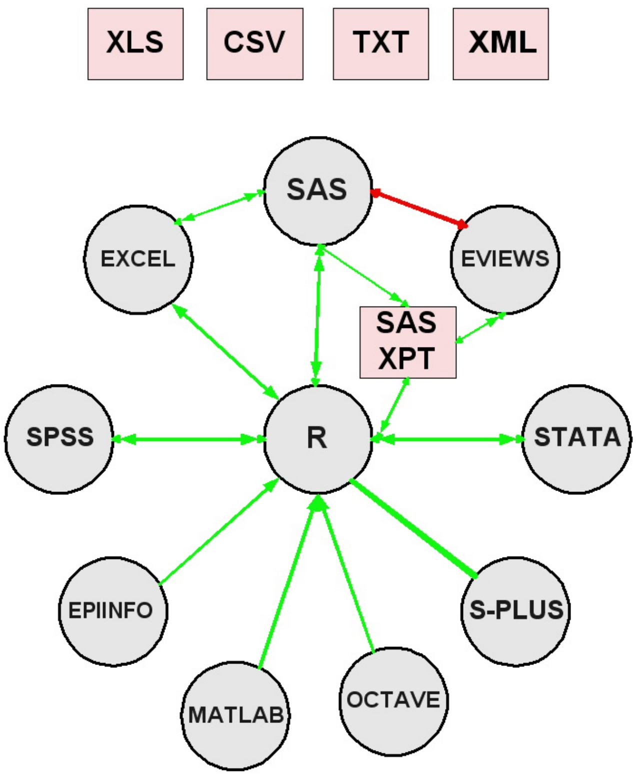


Figure 1: R interfaces to other statistical systems and data formats.

2 R as a Mediator

When using a statistical system we must have in mind that this is not done in isolation and the system must be able to communicate with other systems in order to import data for analysis, to export data for further processing (use the right tool for the right work) and to export results for report writing. When collaborating with other researches, they may have already created the necessary data sets in their favorite statistical package or may be they want to access a data set that is already available in another data format. It can also happen that within the same data process at some stage another statistical tool is more appropriate for a particular task but the final result should be produced in the original system. Examples of such task sharing are presented in the subsequent Sections 3 and 4.1. Even in a small research department like the one at UNIDO many different statistical systems like SAS, Stata, EViews, Octave, SPSS and R are in use and often the need for collaboration between the tools arises. It is not a rare case that a consultant provides the final report on a media in a binary format written by some program like SPSS or Excel or something else. In most of these cases a tool which could "mediate" among the other programs and formats would be very useful. To our experience R provides the most flexible import and export interfaces to most of the available statistical and other packages. A rich variety of facilities for data import and export as well as for communication with databases, other statistical systems and programming languages are available either in R itself or through packages available from CRAN. In Figure 1 most of the interfaces available in R are shown. We do not claim comprehensiveness of the presented relations between the different system but rather illustrate our experience in exchanging data between systems (for example the red arrow between SAS and EViews means that the link between the two packages based on a SAS ODBC driver is to our knowledge broken. The easiest data format to import into R is a simple text file but reading XML, spreadsheet-like data, e.g. from Excel is also possible. The recommended package **foreign** provides import facilities for reading data in the format of the statistical packages Minitab, SAS, S-Plus, SPSS, Stata, SYSTAT and Octave as well as export capability for writing Stata files, while the package **matlab** provides emulation for MATLAB. Further details can be found in the documentation of the packages **foreign** and **Hmisc**.

Working with large data sets could be a problem in R (if the data do not fit in the RAM of the computer) but the interface to RDBMS could help in such cases. Another limitation is that R does not easily support concurrent access to data, i.e. if more than one user is accessing, and perhaps updating, the same data, the changes made by one user will not be visible to the others. This could also be solved by using the interface to relational databases. There are several packages available on CRAN for communication with RDBMS, providing different levels of abstraction. All have functions to select data within the database via SQL queries, and to retrieve the result as a whole, as a data frame or in pieces (usually as groups of rows). Most packages are tied to a particular database—**ROracle**, **RMySQL**, **RSQLite**, **RmSQL**, **RPgSQL**, while the package **RODBC** provides a generic access to any ODBC capable relational database.

Another way to solve the large data set problem is to use the package **filehash**. It implements a simple key-value style database where character string keys are associated with data values that are stored on the disk and the utilities provided allow to treat the database much like the familiar in R environments and lists. Other packages, suitable for working with large data sets are **ff**, **bigmemory** and **biglm**.

2.1 Example: Screening tool for the IDSB Database

The Industrial Demand and Supply Balance (IDSB) Database is one of the three databases disseminated annually by the Statistics Unit of UNIDO. IDSB contains data sets based on the 4-digit level of ISIC Revision 3. The data are derived from output data reported by National Statistical Offices together with UNIDO estimates for ISIC-based international trade data, by utilizing the United Nations Commodity Trade Statistics Database (COMTRADE). IDSB contains annual time series data (in current US dollars) for eight items:

1. Domestic output

2. Total imports

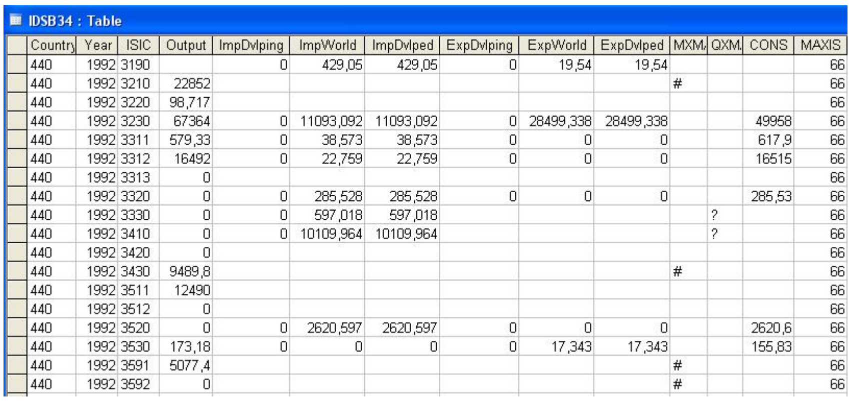
3. Total exports
4. Apparent consumption
5. Imports from industrialized countries
6. Imports from developing countries
7. Exports to industrialized countries
8. Exports to developing countries

The following relation between the above items exist:

- Total imports = Imports from industrialized countries + Imports from developing countries
- Total exports = Exports to industrialized countries + Exports to developing countries
- Apparent consumption = Domestic output + Total imports—Total exports

The data pertain to manufacturing and are arranged according to Revision 3 of ISIC at the 4 digit level, which comprises 127 industries. They are presented by country, manufacturing sector and year. The generation of the final data set involves combination of two independent data sets (INDSTAT and COMTRADE), conversion from one classification (SITC Revision 3) to another (ISIC Revision 3), conversion of the monetary values from current national currency to current USD and other minor adjustments of the data. Although each single data set is verified thoroughly and its quality is guaranteed, the verification of the synthesized data set is poses a serious challenge to the statistical staff of the Unit. A comprehensive screening data set, containing the data records marked by different flags according to a exhaustive set of conditions, is generated automatically during the production process. In the legacy mainframe system this data set was printed on the line printer and the statisticians worked tediously through the hundreds of pages, to find out which data have to be corrected or submitted for further validations. In the new system this file is stored electronically. Due to its enormous size it is not feasible to store the file in the Excel format (limited to some 64000 records), as many statisticians would prefer, but in the MS ACCESS environment. This would facilitate data processing by any user with some SQL knowledge. In Figure 2 an example of this file is shown. But the statisticians usually decline the option of SQL and prefer a statistical system and here comes R to help. Reading the complete data set or a subset of it based on a given selection criterion is as easy as the code shown bellow:

```
> ## First load the RODBC library. If not yet installed, install it by
> ## install.packages("RODBC")
> library(RODBC)
>
> ## Open the ODBC connection to the MDB file 'fname'
```



IDSB34 : Table

Country	Year	ISIC	Output	ImpDvlping	ImpWorld	ImpDvlped	ExpDvlping	ExpWorld	ExpDvlped	MXM/	QXM/	CONS	MAXIS
440	1992	3190		0	429,05	429,05	0	19,54	19,54				66
440	1992	3210	22852							#			66
440	1992	3220	98,717										66
440	1992	3230	67364	0	11093,092	11093,092	0	28499,338	28499,338			49958	66
440	1992	3311	579,33	0	38,573	38,573	0	0	0			617,9	66
440	1992	3312	16492	0	22,759	22,759	0	0	0			16515	66
440	1992	3313	0										66
440	1992	3320	0	0	285,528	285,528	0	0	0			285,53	66
440	1992	3330	0	0	597,018	597,018					?		66
440	1992	3410	0	0	10109,964	10109,964					?		66
440	1992	3420	0										66
440	1992	3430	9489,8							#			66
440	1992	3511	12490										66
440	1992	3512	0										66
440	1992	3520	0	0	2620,597	2620,597	0	0	0			2620,6	66
440	1992	3530	173,18	0	0	0	0	17,343	17,343			155,83	66
440	1992	3591	5077,4							#			66
440	1992	3592	0							#			66

Figure 2: Example of the IDSB screening file.

```
> ch <- odbcConnectAccess("C:/work/idsb34screen.mdb")
>
> ## Create an SQL query of the type:
> ##    "SELECT * FROM table_name WHERE where_condition"
> ##
> ## Execute the query and obtain the selected data in the
> ##    dataframe xdata:
>
> sql <- "Select * from idsb34 where MXMARK <>''"
> xdata <- sqlQuery(ch, sql)
> ## Show the first 5 lines of the result
> xdata[1:5,]
  Country Year ISIC   Output ImpDvlping ImpWorld ImpDvlped ExpDvlping ExpWorld
1      31 1996 3530    0.000         NA       NA        NA         NA       NA
2      31 1997 1551 3964.495         NA       NA        NA         NA       NA
3      31 1997 2927    0.000         NA       NA        NA         NA       NA
4      31 1998 1551 1912.639         NA       NA        NA         NA       NA
5      31 1999 1512 2196.757         NA       NA        NA         NA       NA
  ExpDvlped MXMARK QXMARK CONS MAXIS
1        NA      #     NA   NA    85
2        NA      #     NA   NA    85
3        NA      #     NA   NA    85
4        NA      #     NA   NA    85
5        NA      #     NA   NA    85
```

Now the data set `xdata` can be processed easily in the customary environment of the statistical programming language or it can be stored in some other format to be processed by another system.

2.2 Reading XML data

Using the package **XML** one can read XML files from within R which could be very useful in writing SDMX applications in R. After loading the package with the command `library(XML)` the functions `xmlRoot()`, `xmlSApply()`, `xmlValue()` and many more are available for parsing and navigating through the XML file. `xmlTreeParse()` performs DOM-style parsing while SAX (stream style parsing) is handled by the `xmlEventParse()` function. With `xmlTreeParse()` the channel to the file is open (by default an object of type `XMLDocument` is returned) and then functions like `xmlSApply()`, `xmlValue()` and `getNodeSet()` are called recursively to process the file.

An XML file containing ISIC classifications can be found in the *Examples* directory of the package **ritso**—the first several lines are displayed in the following listing:

```
<?xml version="1.0" encoding="Windows-1252"?>
<!--UNIDO Statistical Data Base Nomenclatures-->
<NDataSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Nomenclatures>
    <NomenclatureType Name="ISIC Rev4" IdentMode="Id" IsImplicit="False"
            HasCombinations="True">
      <Nomenclature Name="ISIC Rev4" Code="ISIC4" TypeDefault="True">
        <NomenclatureCode Code="10" Description="Food products" CompType="M"
                ValidFrom="1962" ValidUntil="9999">
          <Translations>
            <FRShort>produits alimentaires</FRShort>
            <SPShort>productos alimenticios</SPShort>
          </Translations>
        </NomenclatureCode>
...
```

The following example illustrates the usage of some functions from the package **XML**. We start by loading the **XML** and defining the `uri` of the target file (using the function `system.file()` to access the *Examples* directory of the package **ritso**). The document is parsed using `xmlTreeParse()` function. Note the parameter `useInternalNodes` which is set to `TRUE` and requires that an internal structure is used (an object of S3 class `XMLInternalDocument` is returned). The function `object.size()` shows the minimal size of the returned object—in this case it is 308 bytes, which otherwise would be of size about 1.5MB. With subsequent calls to `getNodeSet()` the codes belonging to the ISIC Revision 4 classification and their descriptions are retrieved. These are joined together to form a data frame with two columns and its first several lines are displayed. For navigating and searching in the parsed document the powerful language XPath is used (for an XPath tutorial see Nic and Jirat, 2010, which is available in many languages). Finally the allocated memory is released using the `free()` function.

```
> get.wf <- function(fname) {
+     work <- system.file("work", package = "ritso")
+     paste(work, "/", fname, sep = "")
+ }

> library(XML)
> uri <- system.file("examples", "ISIC.xml", package = "ritso")
> doc = xmlTreeParse(uri, useInternalNodes = TRUE)
> object.size(doc)

308 bytes

> mynmcl <- getNodeSet(doc, "//Nomenclature[@Code='ISIC4']")
> mynodes <- getNodeSet(mynmcl[[1]], "//NomenclatureCode")
> mycode <- getNodeSet(mynmcl[[1]], "//NomenclatureCode[@Code='1010']")
> xmlGetAttr(mycode[[1]], "Description")

[1] "Processing/preserving of meat"

> mycodes <- sapply(mynodes, xmlGetAttr, "Code")
> mydescs <- sapply(mynodes, xmlGetAttr, "Description")
> df <- cbind(mycodes, mydescs)
> head(df)

     mycodes
[1,] "10"
[2,] "1010"
[3,] "1010A"
[4,] "1020"
[5,] "1020A"
[6,] "1030"
     mydescs
[1,] "Food products"
[2,] "Processing/preserving of meat"
[3,] "1010 includes 1020, 1030, 1040, 1050, 106, 107 and 1080"
[4,] "Processing/preserving of fish, etc."
[5,] "1020 includes 1040"
[6,] "Processing/preserving of fruit,vegetables"

> free(doc)
```

Although very simple this example gives an idea how the library can be used to achieve much more sophisticated results.

2.3 Reading and Writing in Excel Format

Microsoft Excel is probably the most widely used spreadsheet program. The popular use of Excel includes entering and cleaning of data or more general basic data management, tabulation and graphics, simple data processing and statistical calculations. Different add-on packages and tools like XLSTAT or RExcel exist which tend to enhance the statistical and analytical functionality of Excel. Since Excel is available to many people as a Microsoft Office component it is often used as everyday statistics software (e.g. why not use it if I already have it). Excel is easy and intuitive to use (while statistical software is usually perceived as difficult to learn). However there are some deficiencies in the area of numerical precision, coverage of statistical techniques and graphical presentation. The quality of Excel as a statistical and graphical tool is often criticized (see Burns, 2010, and the references therein) but nevertheless much statistical data are available as Excel worksheets. It is also often necessary to create Excel worksheets as output of the statistical analysis. The reading and writing of Excel files is very important since most of our users (and data suppliers) request (or provide) data as Excel files (in one of the many Excel Versions).

R (mainly through its packages at CRAN) provides many ways to access Excel worksheets. The simplest and most straightforward method is to store the data to a comma-delimited file, and then use `read.csv()` to read the data into R. This method will be especially useful if there is additional information (metadata, like for example headings, footnotes and other descriptive material) in the spreadsheet because these information could be removed manually by editing the file derived from Excel. Of course as simple as it is, this method will not be satisfactory in most practical situations when the spreadsheets to be accessed from R are updated on a regular bases and the manual intervention is not desirable or not feasible. Also the necessity could exist that several Excel files are accessed simultaneously or the data comes from several worksheets within the same Excel file.

2.3.1 Reading and writing CSV

Let us consider a simple example showing how to create and then read a CSV file. We start by loading the data set `un86` from the package **robustbase** which contains seven socioeconomic variables observed for 73 countries. The names of the countries are stored as row names. This data set is from United Nations (1986). It was used in Daigle and Rivest (1992) to illustrate a robust biplot method (see also Todorov and Filzmoser, 2009b). After loading the package with the command `library(ritso)` and the data set with the command `data(un86)` we use the function `write.csv()` to write the data frame to a coma separated file which can be read by Excel. Note that the function `write.csv()` uses "." for the decimal point and a comma for the separator. If some other locale is used, e.g. German the data frame has to be written by `write.csv2()` which uses a comma for the decimal point and a semicolon for the separator, the Excel convention for CSV files in some Western European locales.

The result of opening the CSV file in Excel is shown in Figure 3.

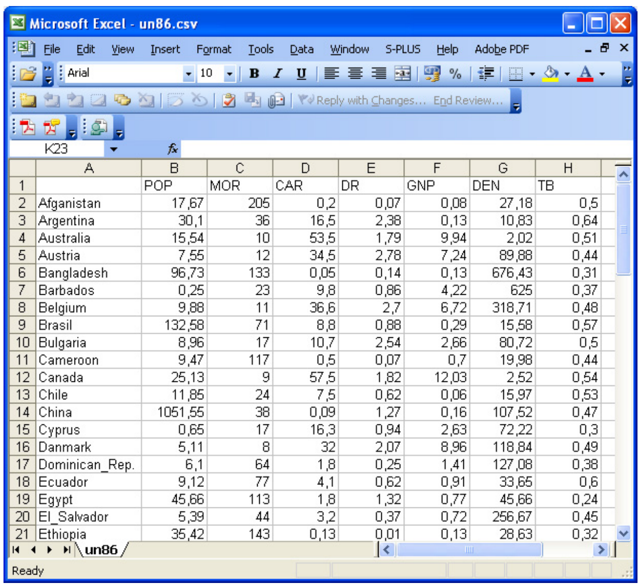
```
> library(ritso)

Scalable Robust Estimators with High Breakdown Point (version 1.2-00)
Scalable Robust Estimators with High Breakdown Point for
Incomplete Data (version 0.3-00)
R in the Statistical Office (version 0.1-00)

> data(un86)
> head(un86)

              POP MOR   CAR   DR  GNP    DEN   TB
Afganistan  17.67 205  0.20 0.07 0.08  27.18 0.50
Argentina   30.10  36 16.50 2.38 0.13  10.83 0.64
Australia   15.54  10 53.50 1.79 9.94   2.02 0.51
Austria      7.55  12 34.50 2.78 7.24  89.88 0.44
Bangladesh  96.73 133  0.05 0.14 0.13 676.43 0.31
Barbados     0.25  23  9.80 0.86 4.22 625.00 0.37

> write.csv(un86, file = get.wf("un86.csv"))
```

To read data stored by Excel in a CSV file is as easy as writing it. This is done by the function `read.csv()`. The country names are stored by `write.csv()` in the first column of the file that we wrote as CSV. Now when reading the file we can retrieve them and assign to the row names by specifying `row.names=1` in the call to `read.csv()`, otherwise they will be stored in a new column (the function argument `row.names` can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names). In order to increase the storage efficiency the function `read.csv()` (as well as `read.table()` which is the basis for `read.csv()`) if a column consists of strings (as the country names in our case) this column will be stored as a factor. While this gives advantages in terms of efficiency it could cause problems if we want to use the variable as simple strings. For this purpose we could suppress the conversion to factor by setting the argument `stringsAsFactors=FALSE`. To ensure that such conversion of string variables to factors is never performed one could set the system wide option:

```
> options(stringsAsFactors = FALSE)
```

All this is shown in the following example.

```
> xun86 <- read.csv(file = get.wf("un86.csv"), row.names = 1,
+     stringsAsFactors = FALSE)
> head(xun86)
```



	A	B	C	D	E	F	G	H
1		POP	MOR	CAR	DR	GNP	DEN	TB
2	Afganistan	17,67	205	0,2	0,07	0,08	27,18	0,5
3	Argentina	30,1	36	16,5	2,38	0,13	10,83	0,64
4	Australia	15,54	10	53,5	1,79	9,94	2,02	0,51
5	Austria	7,55	12	34,5	2,78	7,24	89,88	0,44
6	Bangladesh	96,73	133	0,05	0,14	0,13	676,43	0,31
7	Barbados	0,25	23	9,8	0,86	4,22	625	0,37
8	Belgium	9,88	11	36,6	2,7	6,72	318,71	0,48
9	Brasil	132,58	71	8,8	0,88	0,29	15,58	0,57
10	Bulgaria	8,96	17	10,7	2,54	2,66	80,72	0,5
11	Cameroon	9,47	117	0,5	0,07	0,7	19,98	0,44
12	Canada	25,13	9	57,5	1,82	12,03	2,52	0,54
13	Chile	11,85	24	7,5	0,62	0,06	15,97	0,53
14	China	1051,55	38	0,09	1,27	0,16	107,52	0,47
15	Cyprus	0,65	17	16,3	0,94	2,63	72,22	0,3
16	Danmark	5,11	8	32	2,07	8,96	118,84	0,49
17	Dominican_Rep.	6,1	64	1,8	0,25	1,41	127,08	0,38
18	Ecuador	9,12	77	4,1	0,62	0,91	33,65	0,6
19	Egypt	45,66	113	1,8	1,32	0,77	45,66	0,24
20	El_Salvador	5,39	44	3,2	0,37	0,72	256,67	0,45
21	Ethiopia	35,42	143	0,13	0,01	0,13	28,63	0,32

Microsoft Excel - un86.csv
File Edit View Insert Format Tools Data Window S-PLUS Help Adobe PDF
Arial
10
Reply with Changes... End Review...
K23
un86
Ready

Figure 3: The un86 data set read in Excel

```
             POP MOR   CAR   DR  GNP    DEN   TB
Afganistan 17.67 205  0.20 0.07 0.08  27.18 0.50
Argentina  30.10  36 16.50 2.38 0.13  10.83 0.64
Australia  15.54  10 53.50 1.79 9.94   2.02 0.51
Austria     7.55  12 34.50 2.78 7.24  89.88 0.44
Bangladesh 96.73 133  0.05 0.14 0.13 676.43 0.31
Barbados    0.25  23  9.80 0.86 4.22 625.00 0.37
```

There is no need to mention that the argument `file=...` in the call to `read.csv()` could be not only a file name on the local computer but also an URL and in this case the file will be read from Internet.

2.3.2 Packages xlsReadWrite and xlsReadWritePro

Using the functions from the package **xlsReadWrite** one can read and store a data frame, a matrix or a vector as an Excel file in Excel 97-2003 file format. The package runs only on Windows machines and uses proprietary 3rd party code.

We start by loading the package **xlsReadWrite**. If this is the first loading of the package (after installing from CRAN) a message regarding the shared library `getshlib` will be issued.

```
> library(xlsReadWrite)

xlsReadWrite version 1.5.3 (0b78c1)
Copyright (C) 2010 Hans-Peter Suter, Treetron, Switzerland.
```

```
This package can be freely distributed and used for any
purpose. It comes with ABSOLUTELY NO GUARANTEE at all.
xlsReadWrite has been written in Pascal and contains binary
code from a proprietary library. Our own code is free (GPL-2).

Updates, issue tracker and more info at http://www.swissr.org.
```

The problem is that the **xlsReadWrite** code is free, but also proprietary code is used which can only be distributed legally in pre-compiled binary form. Therefore the CRAN version of the package only distributes a place holder for the library and it is necessary to download the binary `shlib` separately. Let us do this now.

```
## Get the regular shlib (420 KB) by
##  executing the following command:
xls.getshlib()
```

Then we write the data set to an Excel file using the function `write.xls()`.

```
> write.xls(un86, file = "c:/work/un86.xls")
```

With the function `read.xls()` an Excel file can be read into R. Let us read the file which we just created.

```
> xun86 <- read.xls(file = "c:/work/un86.xls")
> head(xun86)
```

```
             POP MOR   CAR   DR  GNP    DEN   TB
Afganistan 17.67 205  0.20 0.07 0.08  27.18 0.50
Argentina  30.10  36 16.50 2.38 0.13  10.83 0.64
Australia  15.54  10 53.50 1.79 9.94   2.02 0.51
Austria     7.55  12 34.50 2.78 7.24  89.88 0.44
Bangladesh 96.73 133  0.05 0.14 0.13 676.43 0.31
Barbados    0.25  23  9.80 0.86 4.22 625.00 0.37
```

The functions in this package can only be used to read and write one worksheet. Which worksheet we want to read is specified by the argument `sheet=` and can be either the sequential number of the worksheet or its name.

In some cases we want to create a workbook consisting of several worksheets. This can be done by the *Pro* version of the package, i.e. **xlsReadWritePro**. This package is a shareware and is not available from CRAN. It can be downloaded from http://www.swissr.org/ as an evaluation license for 30 days. The license costs are indicated in Table 1.

Table 1: Pricing and license types for **xlsReadWritePro**

Single user license	65 EUR
Company/campus wide	520 EUR
Non-comercial/private	18 EUR

Apart from allowing to write in more than one worksheet the *Pro* version has many more useful functions.

- append data to existing files
- work with in-memory Excel "files" (called `xls-obj`)
- manage sheets (select, insert, copy, rename, delete, hide, info, list)
- support images (add, delete, count, list, export, info)
- read and write formula values

Formal support is also available.

We have already written the data set `un86` as an Excel file. Let us now write together with the data the corresponding metadata—the UN country codes and the country names—in a separate worksheet. The country codes and names are available in the data set `country`. We start with creating a new file using the function `xls.new()` and specifying the full path of the target Excel file. This function will return a reference to an `xls-obj` which can be used for further calls. If the file already exists we could open it using the function `xls.open()` and append to it.

```
> library(ritso)
> data(un86)
> data(country)
> library(xlsReadWritePro)

xlsReadWritePro version 1.6.3 (93a6d7)
Copyright (C) 2006 - 2010 by Hans-Peter Suter, Treetron, Switzerland.
All rights reserved.

> xls <- xls.new("C:/work/un86-meta.xls")
> xlsReadWritePro::write.xls(un86, file = xls, sheet = "un86",
+     colNames = TRUE)
> xlsReadWritePro::write.xls(country, file = xls,
+     sheet = "metadata", colNames = TRUE)
> xls.close(xls)
```

We can add images to the Excel file that we have just created. For illustrating this we create an image by plotting a pairwise scatter plot of the `un86` data set and saving it as a `jpeg` file in the working directory. Then we open the Excel file with the function `xls.open()` and using the function `xls.image()` with `action="add"` we write the image into the third worksheet of the file. The range where the image will be placed has to be specified. After that the file is stored using the function `xls.close()`.

```
> library(rrcov)
> data(un86)
> xfile <- "c:/work/un86-meta.xls"
> jpeg(file = "c:/work/un86.jpg")
> plot(un86)
> box(which = "outer")
> dev.off()

> exc <- xls.open(xfile, readonly = TRUE)
> xls.image(exc, "add", img = "c:/work/un86.jpg",
+     sheet = 3, range = c(2, 4, 40, 13))
> xls.close(exc, file = "imagedata.xls")
```

Thus, an image can be inserted into a particular worksheet (given by the argument 'sheet') and particular range - argument 'range' can be a numeric vector with 4 elements (R1,C1,R2,C2) or a character string defining a named range. The latter may change the active sheet (sheet argument will be ignored). Supported image types are: `jpg`, `bmp`, `png`, `wmf` and `emf`. Files in the `bmp` format will be stored in Excel as `png`.

Using the parameter `action` with values `"count"`, `"list"`, `"info"` one can obtain information about the images present in the file.

```
> xls.image("imagedata.xls", "count", sheet = 3)

[1] 1

> xls.image("imagedata.xls", "list", sheet = 3)

[1] "un86.jpg"

> xls.image("imagedata.xls", "info", img = "un86.jpg",
+     sheet = 3)

$idx
[1] 1

$name
[1] "un86.jpg"
```

```
$type
[1] "jpg"

$area
[1]  2  4 40 13
```

Any image from an Excel file can be exported to an image file—in the example below the first image in the third worksheet will be exported to the `jpeg` file `"un86-export.jpg"`.

```
> xls.image("imagedata.xls", "export", sheet = 3,
+     img = 1, target = "un86-export.jpg")
```

Other actions which could be performed by the function of `xls.image()` are `action=delete` and `action=deleteAll`.

2.3.3 Using SQL

The relational databases are widely used in any data related area and provide a very useful mechanism for storing data in structured tables. To access the data stored in a relational database a standard mechanism is used—the Structured Query Language (SQL). A common interface for accessing a variety of relational databases is the ODBC (Open Database Connectivity) facility which was originally developed for MS Windows (and widest variety of ODBC connectors is available on that platform). In R the ODBC interface is implemented in the **RODBC** package available from CRAN.

We already used ODBC and the package **RODBC** in the example in Section 2.1 to access a MS ACCESS database. Since ODBC drivers for Excel exist (on Windows platform) we can use the same technique to read and write Excel files. After loading the package **RODBC** we open a connection with the function `odbcConnectExcel()` by specifying the full path of the Excel file. We can query the names of the worksheets available in the file with the function `sqlTables()` and can read and write with the functions `sqlRead()` and `sqlWrite()`. Thus it is possible to work with more than one worksheets in the same excel file. Worksheets can be created or data can be appended to existing ones (using the `append=` argument).

```
> library(ritso)
> data(un86)
> data(country)
> library(RODBC)
> ch <- odbcConnectExcel(get.wf("un86odbc.xls"),
+     readOnly = FALSE)
> sqlSave(ch, un86, "un86", append = TRUE)
> sqlSave(ch, country, "country", append = TRUE)
> odbcCloseAll()
> ch <- odbcConnectExcel(get.wf("un86odbc.xls"))
> tbls <- sqlTables(ch)
> tbls

                                       TABLE_CAT
1 C:\\R\\R-2.12.0\\library\\ritso\\work\\un86odbc
2 C:\\R\\R-2.12.0\\library\\ritso\\work\\un86odbc
3 C:\\R\\R-2.12.0\\library\\ritso\\work\\un86odbc
4 C:\\R\\R-2.12.0\\library\\ritso\\work\\un86odbc
  TABLE_SCHEM TABLE_NAME   TABLE_TYPE REMARKS
1        <NA>   country$ SYSTEM TABLE    <NA>
2        <NA>      un86$ SYSTEM TABLE    <NA>
3        <NA>    country        TABLE    <NA>
4        <NA>       un86        TABLE    <NA>

> sql <- "Select * from un86"
> xun86 <- sqlQuery(ch, sql, stringsAsFactors = FALSE)
> odbcCloseAll()
> head(xun86)

    rownames   POP MOR   CAR   DR  GNP    DEN   TB
1 Afganistan 17.67 205  0.20 0.07 0.08  27.18 0.50
2  Argentina 30.10  36 16.50 2.38 0.13  10.83 0.64
3  Australia 15.54  10 53.50 1.79 9.94   2.02 0.51
4    Austria  7.55  12 34.50 2.78 7.24  89.88 0.44
5 Bangladesh 96.73 133  0.05 0.14 0.13 676.43 0.31
6   Barbados  0.25  23  9.80 0.86 4.22 625.00 0.37
```

2.3.4 Other packages

There are also other packages for reading and writing Excel data which will not be considered in detail here.

The package **WriteXLS** does not require Excel itself but requires Perl with module `Text::CSV_XS` and can write one or more data frames to Excel. Each data frame will be written to a separate named worksheet. Appending to a file is not possible and the Excel file, if it exists, is overwritten. The following example writes built-in data frames to an Excel file. The operations will be only performed if Perl and all necessary modules are properly installed—verified by the function `testPerl()`.

```
> library(WriteXLS)
> if (testPerl(verbose = FALSE)) {
+     WriteXLS("iris", "iris.xls")
+     WriteXLS(c("iris", "infert", "esoph"), "Example.xls")
+     unlink("iris.xls")
+     unlink("Example.xls")
+ }
```

The package **gdata** can read and write Excel 2003 and Excel 2007 files on all platforms but requires Perl and **dataframes2xls** requires Python. There are several packages like **RExcelXML** (depends on **ROOXML**, **Rcompression** and **XML**) and **xlsx** (depends on Java) which can read or read and write Excel 2007 files.

While most of the considered so far packages allow to access (read and/or write) Excel files from within R there is one exception, the package **RExcel** which works the other way around. The authors have recognized that despite the drawbacks and caveats of the usage of Excel for statistics it is an important and widely spread tool and have extended its functionalities by the entire R environment. **RExcel** is an add-in to Excel on MS Windows machines that allows the use of R as an extension for Excel. Data can be transferred in both directions and Excel can call R functions to perform calculations and then transfer the results to Excel (for more details see Baier and Neuwirth, 2007).

2.4 Reading Data in SDMX Format

The statistical activities of UNIDO are defined by its responsibility to provide the international community with global industrial statistics and meet internal data requirements to support the development and research programme of the Organization. Currently, UNIDO maintains an industrial statistics database, which is regularly updated with the data, collected from National Statistical Offices and OECD (for OECD member countries). For these types of activities the Statistical Data and Metadata Exchange (SDMX) technical standards and content-oriented guidelines are a natural choice as they provide common formats and nomenclatures and are developed for exchange of statistical data and metadata using modern technology. The United Nations Statistical Commission (see United Nations, 2008) recognizes SDMX as the preferred standard for dissemination of national data and metadata using web technology as a way to reduce the international reporting burden. The SDMX initiative is an international project carried out by several international organizations, including the United Nations, specialized agencies, OECD and Eurostat.

The SDMX Technical Standards Version 2.0 developed and reviewed with the goal to replace, within the context of the International Organization for Standardization (ISO), the previous version (ISO/TS 17369:2005 SDMX), which provides technical specifications for the exchange of data and metadata based on a common information model. Its scope is to define formats for the exchange of aggregated statistical data and the metadata needed to understand how the data are structured. The major focus is on *data presented as time series*, although cross-sectional XML formats are also supported. The latest work broadens the technical framework to support wider coverage of metadata exchange as well as a more fully articulated architecture for data and metadata exchange.

As it takes years for a large number of national and international agencies to adopt common standards, SDMX started with a limited number of agencies that have the required technical facilities in place and are familiar with sharing data. The practical utilization of SDMX standard is still in its infancy, not only in UNIDO but also in most international organizations. Some prominent pilot projects (not a complete list), from which lessons can be learned include:

- SDMX Open Data Interchange (SODI) is a data-sharing and exchange project within the European Statistical System. The project started with a pilot exercise involving National Statistical Offices of France, Germany, the Netherlands, Sweden and the United Kingdom. The statistical institutes of Denmark, Italy, Norway and Slovenia joined the pilot exercise in 2006, while Finland and Ireland joined in 2007.

- FAO CountrySTAT, which is based on the application of data and metadata standards of FAOSTAT and SDMX, is a web-based system being developed since May 2004 using PX-Web at FAO Headquarters. It was successfully tested in the statistical offices of Kenya, Kyrgyzstan and Ghana during 2005. Many other developing and industrialized countries have shown an interest in and are adopting it: http://www.fao.org/es/ess/countrystat/

- Data exchange between OECD and IMF: Using the IMF web service (IMF.Stat) to pull exchange rates data from IFS (SDMX-ML, using SOAP and RESTful interfaces).

There are many ways to use SDMX to exchange data characterizing this activity in simple terms. For example, a primary distinction can be made on whether the data are being sent by one counter party to another (called a ”push” scenario) or whether the data are posted in an accessible location, and then obtained when needed (called a ”pull” scenario). In the push mode, which is the traditional data-sharing mode, different means, such as e-mails and file transfers, are used to exchange data. This is how UNIDO and many other international agencies collect data from the National Statistical Offices and the other and international organizations. To use SDMX for data-reporting or data collection, which are actually two aspects of the same task, that is, the task of data exchange, at least two counterparties are required, one or more of which provides data to another. Any of these counterparties must adopt the same technical standards, have a common data structure and use common vocabulary. The lack of necessary technical facilities could be a serious stumbling block for developing countries involved in the process.

As a first step towards SDMX utilization, UNIDO is currently developing a data and metadata exchange procedure based on the web service provided at OECD.Stat. This is the central repository where validated statistical data and metadata are stored, and is intended in due course to become the sole coherent source of statistical data and related metadata for the OECD statistical publications. Utilizing the OECD web service will allow to automatically retrieve and process data for all OECD member countries, which currently is done by transferring Excel files (see Upadhyaya and Todorov, 2008). This exercise is being extended to support the data and metadata transfer with the National Statistical Offices of developing countries starting with Lebanon and Nigeria.

The OECD.Stat web service provides possibilities to retrieve all necessary data from the data set of interest (SSIS_BSC) stored in the statistical data warehouse using SDMX-ML Query format but it is also possible to get updates and revisions only or to filter by dimensions, attributes and date ranges. The data are returned in SDMX-ML Generic Data format (v. 2). Similarly, it is possible to access metadata in the form of data structure definitions, code lists and concept schemes or relevant reference metadata returned in SDMX-ML Structure format. Figure 4 shows the planned implementation of SDMX in the *UNIDO Statistical Production Process*. More information about the

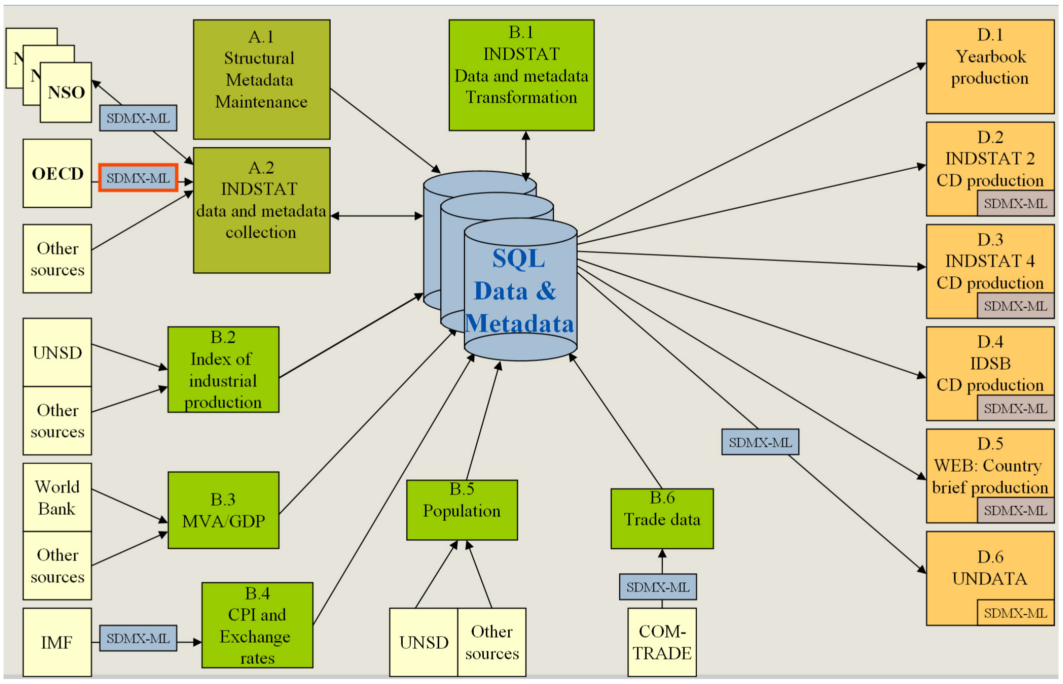


Figure 4: SDMX in the UNIDO Statistical Production Process.

UNIDO SDMX project can be found on the publicly available Wiki site of the project at http://sdmx.wikispaces.com/.

The possibilities to read and write SDMX-ML with R will be illustrated on a simple example in which exchange rates from IMF/IFS (International Monetary Fund/International Financial Statistics) file are read. On the IMF web site http://www.imf.org/oecd there are several publicly available files in SDMX format and we will consider the annual exchange rates for the period 1980 to 2007. These data are available in the file http://www.imf.org/oecd/ExchangeRatesIMFIFSAnnually1980-2007 (also available in the *Examples* directory of the package **ritso**). We start by defining the uri to the file and loading the package **sdmxer**. Currently the package **sdmxer** is under development and does not provide much SDMX-related functionality but is simply an interface to the R package **XML**.

We parse the file using the function `xmlTreeParse()` which generates an R structure representing the XML tree. This function has numerous parameters and options but in this example we use the defaults and obtain an object of type `XMLDocument`. From the returned object we extract the root with the function `xmlRoot()` which can be processed further using the functions in the **XML** package. From the first several lines of the SDMX-ML file we see that an XML Schema is defined in "http://sdmx.imf.org/oecd/IMF_KF_ER.xsd" and the file contains SDMX messages in *Compact* format. It starts by a *Header* and than continues with *Timeseries* blocks for each country (given by the attribute *CountryName*) and type of data (given by the attribute *TS_Key*).

```
> library(sdmxer)

SDMX utilities (version 0.1-00)

> uri <- "http://sdmx.imf.org/oecd/ExchangeRates_IMF_IFS_Annual_80_07.xml"
> uri <- system.file("examples", "ExchangeRates_IMF_IFS_Annual_80_07.xml",
+     package = "ritso")
> doc <- xmlRoot(xmlTreeParse(uri))
```

We will extract the time series containing the annual exchange rates (market rate period average) for Korea, i.e. will be interested in these data which have `CountryName == "Korea"` and `TS_Key == "..RF.ZF..."`. For this purpose a function is defined `select.ts()` which takes an `XMLNode` object and optional country name and record type and returns a logical value (`TRUE` or `FALSE`).

```
> select.ts <- function(x, ct = "Korea", key = "..RF.ZF...") {
+     if (xmlName(x) == "Header")
+         return(FALSE)
+     att <- as.list(xmlAttrs(x))
+     if (att$CountryName == ct & substr(att$TS_Key,
+         4, 15) == key)
+         return(TRUE)
+     return(FALSE)
+ }
```

Using this function we will extract only the data of interest—the object `kr` below will consist of only the nodes containing the required information for Korea. We can print the attributes of these data subset using the functions `xmlAttrs()`.

```
> kr <- doc[[which(xmlSApply(doc, select.ts))]]
> xmlAttrs(kr)
```

```
                         Frequency
                               "A"
                          Database
                             "IFS"
                       CountryName
                           "Korea"
                           Country
                             "542"
                            TS_Key
                   "542..RF.ZF..."
                        Descriptor
                     "MARKET RATE"
                             Units
"National Currency per US Dollar"
                             Scale
                            "None"
```

After that we will use the extracted data to create a time series object `ts` and print its values.

```
> ex <- xmlSApply(kr, function(x) as.vector(as.numeric(xmlAttrs(x))))
> colnames(ex) <- NULL
> kr.exrate <- ts(ex[2, ], start = ex[1, 1])
> kr.exrate
```

```
Time Series:
Start = 1980
End = 2007
Frequency = 1
 [1]  607.4325  681.0283  731.0842  775.7483  805.9758
 [6]  870.0200  881.4542  822.5675  731.4683  671.4558
[11]  707.7642  733.3533  780.6508  802.6708  803.4458
[16]  771.2733  804.4533  951.2892 1401.4367 1188.8167
[21] 1130.9575 1290.9946 1251.0883 1191.6142 1145.3192
[26] 1024.1167  954.7905  929.2137
```

Now we can plot the obtained time series using the following command:

```
> plot(kr.exrate, xlab = "Time", ylab = "Korea",
+     main = "Market Rate, National Currency per US Dollar")
```

The resulting time series plot is shown in Figure 5.

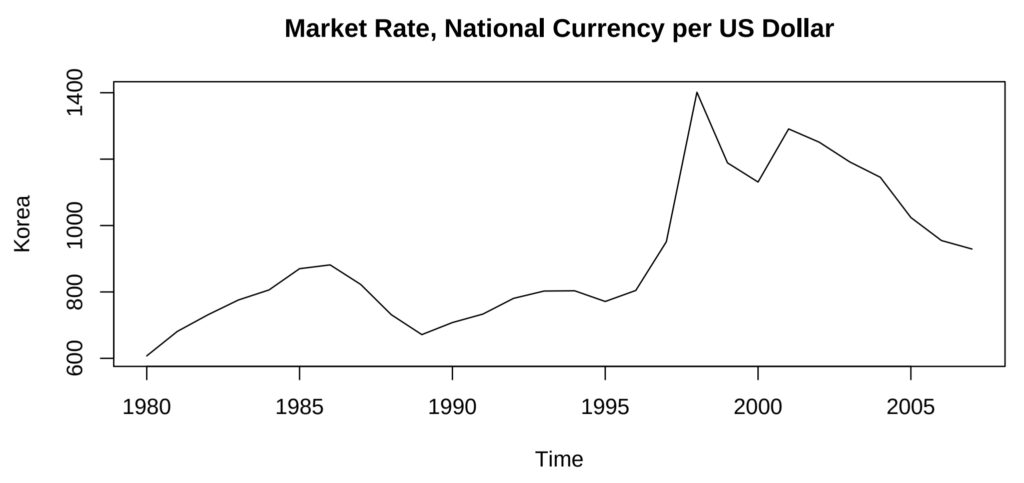


Figure 5: Example of a time series plot: Exchange rates, national currency per US dollar, for Korea, 1980-2007.

Of course even this simple example leaves many questions open. We can see that the attribute `TS_Key` is the unique identifier which could be used to specify the needed data—it identifies the country code and the record type: `TS_Key="542..RF.ZF..."`. Of course one has to know that 512 is the numerical IFS code for Korea, which in the United Nations countries and areas classification has the code 410 or the ISO code *KOR*. Also the name of the countries could be different—we have *Republic of Korea* in the UN classification. To solve such problems is the purpose of the already mentioned XML Schema—we find there the complete specification of the compact data format with all of the data types and key lists, including the entry for Korea shown below.

```
<xs:enumeration value="542" id="KOR">
<xs:annotation>
<xs:documentation xml:lang="en">Korea, Republic of</xs:documentation>
</xs:annotation>
</xs:enumeration>
```

2.5 Example: Exporting data from INDSTAT 4 Database to R

The INDSTAT4 database contains time series data from 1990 onwards. Data are available by country, year and ISIC at the 3- and 4-digit levels of ISIC (Revision 3), which comprises 151 manufacturing sectors and sub-sectors. All value data are originally stored in national currency at current prices. The system allows for data conversion from national currency into current US dollars using the average period exchange rates as given in the IMF/IFS (International Monetary Fund/International Financial Statistics).

When the INDSTAT4 database system was created, the first priority was the data requirements of cross-country economic analysis. Much of the work consisted in developing a set of data that were comparable and consistent across countries and over time. The database contains seven principle indicators of industrial statistics:

1. number of establishments
2. number of employees
3. wages and salaries
4. output
5. value added
6. gross fixed capital formation
7. number of female employees

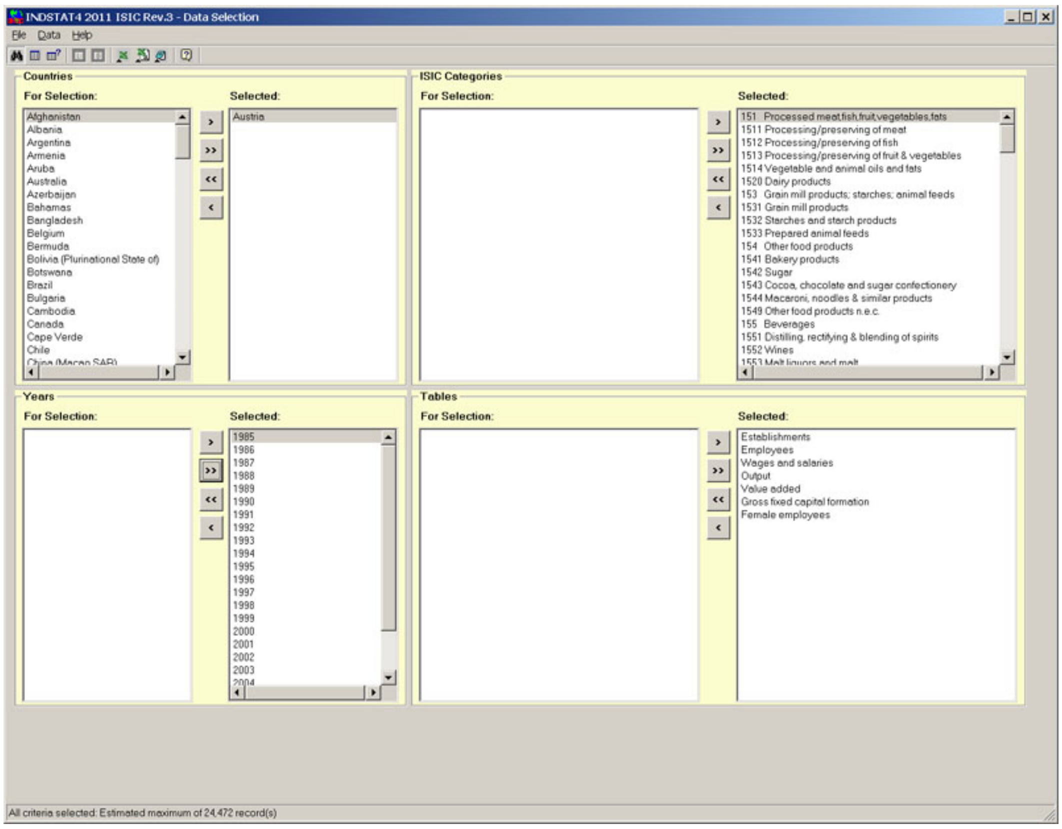
In support of the statistical data, relevant metadata information is collected from national data suppliers and OECD, which is available to users in a highly flexible form. The current edition of INDSTAT4 covers:

- Number of countries: 126
- Reference period: 1990-2007

where coverage in terms of years, as well as data items, may vary from country to country depending on data availability.

INDSTAT data are compiled in collaboration with OECD: Data for non-OECD countries are collected by UNIDO, while those for OECD member countries are collected and provided to UNIDO by OECD for inclusion in the database. The data are mostly those reported by national statistical authorities in response to UNIDO country questionnaires or UNIDO/OECD joint country questionnaires. They have been supplemented with data published in national statistical publications as well as UNIDO estimates. In recent years most countries that report their industrial statistics to UNIDO have made a switch over from ISIC Revision 2 to ISIC Revision 3 and a few of them to ISIC Revision 4. These countries now provide their industrial statistics that are based on or related either to ISIC Revision 3 or to ISIC Revision 4. As of 2010, UNIDO has discontinued the production of INDSTAT4-Rev.2, which was a co-product of INDSTAT4-Rev.3 in the past. The visualization software of the database provides graphical user interface for selection of data subsets which can be further exported to Excel or CSV format (see Figure 6).

Recently a utility was added to export the data also to R, SAS, Stata and other statistical software packages. The export selection dialogue is shown in Figure 7. The export to R (or SAS, etc.) is done by simply generating a small R (or SAS, etc.) program to read the exported `.CSV` file into the corresponding statistical package. The format of the `.CSV` file is shown below:



INDSTAT4 2011 ISIC Rev.3 - Data Selection

File Data Help

Countries

For Selection:

Afghanistan
Albania
Argentina
Armenia
Aruba
Australia
Azerbaijan
Bahamas
Bangladesh
Belgium
Bermuda
Bolivia (Plurinational State of)
Botswana
Brazil
Bulgaria
Cambodia
Canada
Cape Verde
Chile
China (Macao SAR)

Selected:

Austria

ISIC Categories

For Selection:

Selected:

151 Processed meat,fish,fruit,vegetables,fats
1511 Processing/preserving of meat
1512 Processing/preserving of fish
1513 Processing/preserving of fruit & vegetables
1514 Vegetable and animal oils and fats
1520 Dairy products
153 Grain mill products; starches; animal feeds
1531 Grain mill products
1532 Starches and starch products
1533 Prepared animal feeds
154 Other food products
1541 Bakery products
1542 Sugar
1543 Cocoa, chocolate and sugar confectionery
1544 Macaroni, noodles & similar products
1549 Other food products n.e.c.
155 Beverages
1551 Distilling, rectifying & blending of spirits
1552 Wines
1553 Malt liquors and malt

Years

For Selection:

Selected:

1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004

Tables

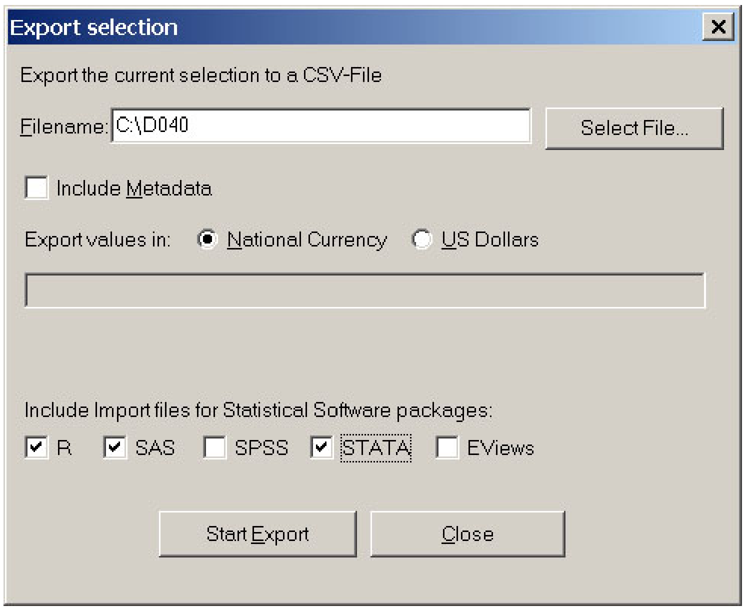
For Selection:

Selected:

Establishments
Employees
Wages and salaries
Output
Value added
Gross fixed capital formation
Female employees

All criteria selected: Estimated maximum of 24,472 record(s)

Figure 6: Data selection dialogue in the UNIDO Industrial Statistics Database INDSTAT 4.



Export selection

Export the current selection to a CSV-File

Filename: C:\D040

Select File...

☐ Include Metadata

Export values in: [✓ National Currency] US Dollars

Include Import files for Statistical Software packages:

☑ R ☑ SAS ☐ SPSS ☑ STATA ☐ EViews

Start Export Close

Figure 7: Data export dialogue in the UNIDO Industrial Statistics Database INDSTAT 4. To request export in R, SAS or other statistical software package the corresponding check box should be checked.

```
> fname <- system.file("Examples", "D040.CSV", package = "ritso")
> head(readLines(con = fname))

[1] "01,040,1990,151,151,238,01,1,2005,#"
[2] "01,040,1990,1511,1511,211,01,1,2005,#"
[3] "01,040,1990,1512,1512,...,01,0,2005,#"
[4] "01,040,1990,1513,1513,27,01,1,2005,#"
[5] "01,040,1990,1514,1514,...,01,0,2005,#"
[6] "01,040,1990,1520,1520,131,01,1,2005,#"
```

and it can be read into R with the following program:

```
> inst.classes <- c("factor", "factor", "integer",
+     "factor", "factor", "numeric", "factor", "factor",
+     "factor", "factor")
> inst.names <- c("ctable", "country", "year", "isic",
+     "isiccomb", "value", "utable", "source", "lastupdated",
+     "unit")
> inst <- read.csv(fname, header = FALSE, col.names = inst.names,
+     colClasses = inst.classes, na.strings = "...")
> head(inst)

  ctable country year isic isiccomb value utable source
1     01     040 1990  151      151   238     01      1
2     01     040 1990 1511     1511   211     01      1
3     01     040 1990 1512     1512    NA     01      0
4     01     040 1990 1513     1513    27     01      1
5     01     040 1990 1514     1514    NA     01      0
6     01     040 1990 1520     1520   131     01      1
  lastupdated unit
1        2005    #
2        2005    #
3        2005    #
4        2005    #
5        2005    #
6        2005    #
```

In a similar manner works the export to SAS, Stata and other software packages.

3 R as a Graphics Engine

A natural way to visualize data are graphs and plots and it hardly needs saying that their use is common even in non-technical documents. Usually in order to produce plots with sufficient output quality to match a well-typeset document much work is required (and rarely performed). Without going into details about graphical data visualization, for which neither the time nor the space is available within this work, we will refer to Tufte (2001) who states that publication quality displays should be both informative and aesthetically pleasing and lists several important aspects of data presentation which contribute to graphical excellence. Some of these aspects are:

- present many numbers in a small space;
- encourage the eye to compare different pieces of data.

To create a useful publication, the graphics have to be mingled with text explaining and commenting them and here comes the next challenge—how to typeset the content so that nicely formatted pages are obtained. The usual tools could be MS Word for writing the text and including the graphics and tables prepared with MS Excel. When using a word processors like MS Word, the text is placed while it is typed (which is referred to as WYSIWYG—"what you see is what you get"). It is clear that beginners like word processing but when one has to perform complex jobs the appeal fades. The advantage of WYSIWYG is at the same time its disadvantage—one has to control everything by eye—but very few people have both the knowledge and the eye to correctly typeset a page with tables, figures and equations. We will consider three tools in the rest of this section: R as a graphic engine, LaTeX as a typesetting program and `Sweave` as a tool linking the two together to produce an integrated high quality publication containing text, graphs and tables—the *International Yearbook of Industrial Statistics* of UNIDO. Another example to be considered is the *Labour Market Information Bulletin* published by the Labour Market Information Division under the Department of Employment of Ministry of Labour and Human Resources in Bhutan.

3.1 R Graphics

One of the most important strengths of R is the ease with which simple exploratory graphics as well as well-designed publication-quality plots can be produced, including mathematical symbols and formulae wherever necessary. While designing the R graphical packages, great care has been taken over the defaults for the minor design choices in graphics. Nevertheless, the user retains full control over the graphics parameters and can modify the defaults if necessary. R provides the standard statistical plots which are usual in most of the statistical packages like scatter plots, box plots, histograms, bar-plots and pie-charts as well as basic 3D plots which can be produced by a single function call. A fundamental feature of the R graphics is that graphical elements can be added sequentially to a plot in order to obtain the final result.

The spreadsheets and especially Microsoft Excel are a useful and popular tool for processing and presenting data and Microsoft Excel has become somewhat of a standard for data storage, at least for smaller data sets. This, along with the advantages of its integration within the Microsoft Office suite and the convenience of its graphical user-interface, its availability, since Excel is often being packaged with new computers, naturally encourages its use for statistical analysis and graphics. However as already mentioned, many statisticians find this unfortunate, since Excel is clearly neither a true statistical software nor a real graphical package. Due to Excel's easy availability many people (including professional statisticians) use it on a daily basis for quick and easy statistical calculations as well as for publication quality graphics. Figure 8 illustrates one of the flaws of the Excel graphical engine and mainly the defaults provided (which in most cases are used blindly by the users). In the left panel the amounts of sales for four years are presented as 3D *bar chart* (now called *column chart* in the Excel terminology) drawn with Excel. If you look at the top of the bars, e.g. the fourth bar with value 94 thousands you will see that it hardly exceeds the 90 thousands grid line. This is due to the fact that we look at the data in three dimensions. In order to read the value correctly the bar should be projected into the two-dimensional plane that contains the horizontal lines, i.e. the bars should be projected on the rear wall (see Krause, 2009). This type of 3D graphs might look "cool" but they are difficult to interpret and thus do not contribute to the very purpose they were created for—to facilitate the understanding of the problem at hand by visually presenting the data. For comparison, the same bar chart created with R is shown in the right panel. If the graphics drawn with MS

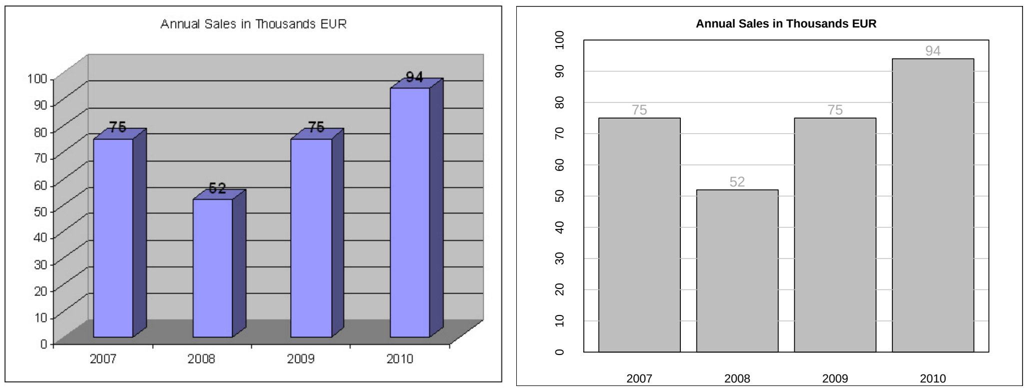


Figure 8: Bar plot drawn by Excel (left) and R (right).

Excel are widely used by the most of the users it can be said that the *pie charts* are the favorite ones. Showing information in one, two or many pie charts is almost unavoidable in any business presentation. Although a pie chart is useful when one wants to show relative proportions of a whole, they have a lot of drawbacks and are easily replaced by a nice bar chart, oriented with horizontal bars, a *dot plot* or simply a table. Without going into details one of these drawbacks will be illustrated in Figure 9. The number of employees in different manufacturing sectors are presented as a pie chart. In sector *Food and beverages* are employed one quarter of the total number of employees and this is very well seen in the chart. If we swap this sector with for example *Wood products* (so that it changes its convenient position), the perception of *a quarter* or 25% changes and now *Food and beverages* looks much larger. To quote Eduard Tufte, "the only worse design than a pie chart is several of them" (Tufte, 1997, p. 178). For more information

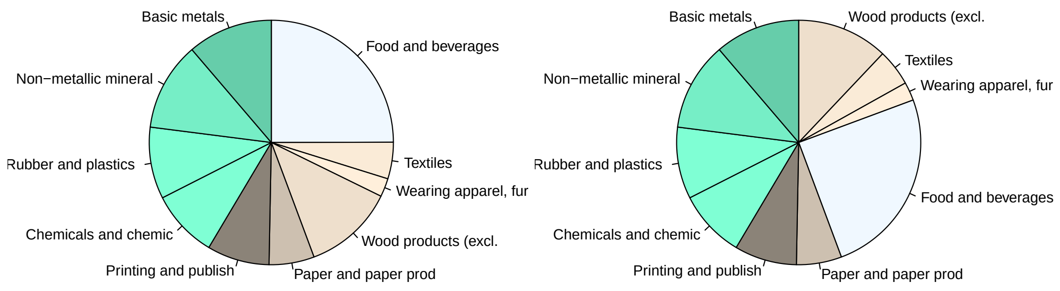


Figure 9: Two pie charts representing the number of employees in different manufacturing sectors. The only difference between the left and the right charts is that *Food and beverages* which represents 25% of the total is swaped with *Wood products*. The perception of *a quarter* or 25% changes—now *Food and beverages* looks larger.

about the problems with Excel graphics see Su (2008).

Figure 10 shows examples of different statistical plots created with the R basic graphics system. The data used are the variables EMP (number of employees) and WS (wages and salaries) for Austria in the period 1963 to 2008 for different manufacturing sectors (at 2-digit level of the ISIC Revision 3 classification).

Histogram and density line: A simple method to visualize data consisting of single variable is by dividing the range of data values into a number of intervals (bins) and plotting the frequency per interval as a bar. Such a plot is called a histogram which is an estimate of the probability density function given by

$$f(x) = \lim_{\delta \to 0} Prob(x - \delta < X \le x)/\delta \qquad (1)$$

Unfortunately the histogram can be very dependent on how the bins are formed and the number of bins. A density line, which is a smoothed histogram (or a smooth estimate of the function given in Equation 1 could be superimposed over the histogram. The R function `density()` produces kernel density estimates for a given kernel and bandwidth. By default, the Gaussian kernel is used but there is an array of other kernels available in R. The density line depends on the chosen smoothing parameter. A histogram and a density plot for the variable EMP (number of employees) in Austria are shown in the upper-left panel of Figure 10.

QQ-plot: Quantile-quantile plots are useful graphical displays for checking the distributional assumptions of the data. The display presents a plot of the quantiles of one sample versus the quantiles of another sample and overlays the points with a line that corresponds to the theoretical quantiles from the distribution of interest. When the distributions have the same shape, the points will fall roughly on a straight line. Extreme points tend to be more variable than points in the center. Therefore it could be expected to see slight departures towards the lower and upper ends of the plot. A normal QQ-plot can be created with the function `qqnorm()` which compares the quantiles of the observed data against the quantiles from a Normal distribution. Then with the function `qqline()` a line based on quantiles from a theoretical Normal distribution can be overlayed. Another useful function is `qqPlot()` from the package **car** which is used for illustration in the upper right panel of Figure 10.

Boxplot: The univariate boxplot (also known as a box-and-whiskers plot) was proposed by Tukey (1977) as a tool for exploratory data analysis. It consists of a box from the lower quartil of the data q_1 to their upper quartil q_3 with a crossbar at the median q_2. Outside of the box can be drawn the fences given by $q_2 + 4(q_3 - q_2)$ for the upper fence and $q_2 - 4(q_2 - q_1)$ for the lower fence. The whiskers are drawn as lines from the box to the most extreme points inside the fences. The observations which lie outside the fences are flagged as outliers. The boxplots are very useful for comparing the distributions of different variables or of the same variable in different groups. In the lower left panel of Figure 10 the boxplots are illustrated on the variable `EMP` for three different ISIC categories.

Bagplot: The bagplot was introduced by P. J. Rousseeuw and Tukey (1999) as a two dimensional generalization of the univariate boxplot. The main components of the bagplot are a *bag* which contains 50% of the data points, a *fence* which separates the non-outliers from the outliers and a *loop* which contains the points that are outside of the bag but inside the fence. The point with highest halfspace depth which is called the *depth median* lies in the center of the bag and is depicted by an asterisk. Similarly as the boxplot, the bagplot visualizes several characteristics of the data: the location represented by the depth median, the spread given by the size of the bag, correlation (the orientation of the bag), skewness (the shape of the bag and the loop) and the tails shown by the points lying near the boundary of the loop and the outliers. In the lower right panel of Figure 10 a bagplot of the variables `EMP` and `WS` from the industrial statistics data for Austria (Total manufacturing) is shown.

While the first three types of plots can be found in one or another form in most of the statistical packages, the *bagplot* to our knowledge is available only in R.

```
> library(ritso)
> hi <- function(x, name = "X") {
+     library(car)
+     x <- x[which(!is.na(x))]
+     hist(x, freq = FALSE, breaks = 10, xlab = name,
+         main = "Histogram")
+     lines(density(x), col = "red")
+     qqPlot(x, ylab = name, main = "Normal Q-Q Plot")
+ }
> bag <- function(x) {
+     x <- x[which(!is.na(x[, 1]) & !is.na(x[, 2])),
+         ]
+     bagplot(x, verbose = FALSE, factor = 3, show.baghull = TRUE,
+         dkmethod = 2, show.loophull = TRUE, precision = 1)
+     title("Bagplot")
+     box()
+ }
> data(aut32)
> df1 <- aut[which(aut$ISIC == 15 | aut$ISIC ==
+     17 | aut$ISIC == 31), ]
> df1$ISIC <- factor(df1$ISIC)
> x <- aut[which(aut$ISIC == "D"), ]
> name <- "EMP"
> par(mfrow = c(2, 2))
> hi(x$EMP, name = name)
> sisic <- paste("ISIC", as.character(df1$ISIC))
> boxplot(EMP ~ sisic, data = df1, col = "lightgray",
+     ylab = name, main = "Boxplot", pars = list(boxwex = 0.4))
> boxplot(EMP ~ sisic, data = df1, notch = TRUE,
+     add = TRUE, col = "blue", pars = list(boxwex = 0.4))
> bag(cbind(EMP = x$EMP, WS = x$WS/1e+06))
```

3.1.1 Multivariate plots

In addition to the traditional statistical plots R has an implementation of the *trellis plots* through the package **lattice** (see Sarkar, 2008). Lattice is a powerful and elegant high level data visualization system that is sufficient for most everyday graphics needs. Additionally it is flexible enough to be easily extended to handle demands of any research.

Scatterplot matrices: plotting continuous variables against each other in the data set (assuming that the size of the data set is not too large) will give an overview of all possible bivariate relations. The left panel of Figure 11 displays the 5 continuous socioeconomic variables from the data set `un86`. In this type of plots it is essential that the space be used very economically, therefore writing the names of the variables on the main diagonal is a very good idea. Further the space on the main diagonal can be used for displaying other information, like for example histograms. We see

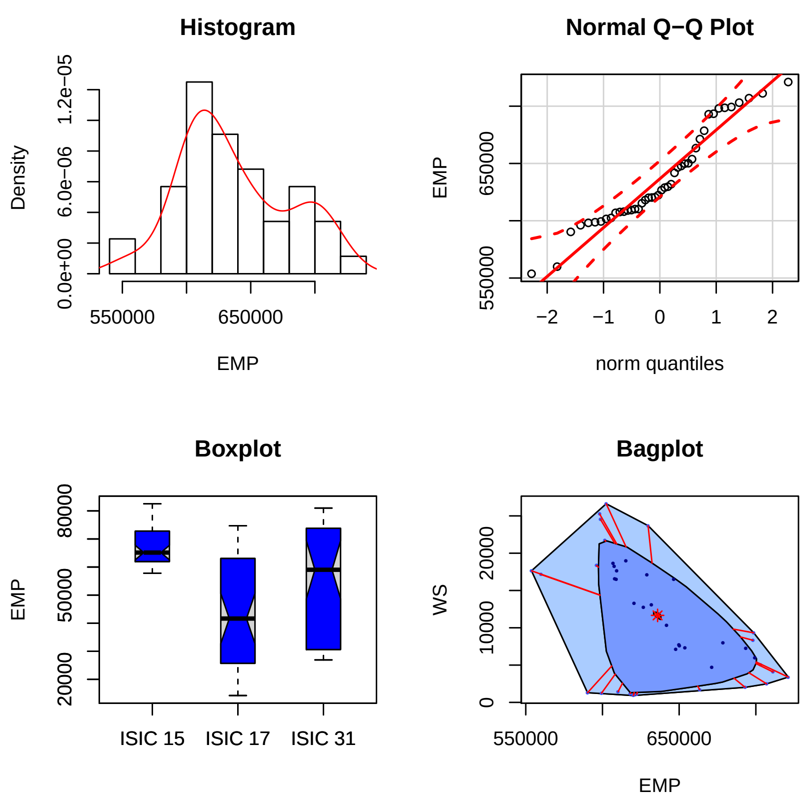


Figure 10: Example of R basic graphics: histogram, QQ-plot, boxplot and bagplot.

that the scatterplot matrix is symmetric and thus the lower diagonal could be used for other purposes too. This is illustrated in the right panel of Figure 11 which displays a specialized plot for identification of multivariate outliers. The upper triangle shows again a bivariate scatter plots but now *tolerance ellipses*, both classical and robust (computed by Minimum Covariance Determinant (MCD)) are superimposed on them. The main diagonal shows the names of the variables together with histograms and the lower diagonal presents the corresponding robust and classical correlations. We see immediately that the classical and robust correlations differ significantly (as well as the tolerance ellipses differ) which indicates that the data set contains outliers. More details about outliers and their identification will be given in a future work (see Section 5).

Parallel coordinates plot: The parallel coordinate plots (see Wegman, 1990; Venables and Ripley, 2003) help to display simultaneously a large number of continuous variables. This visualization technique for multivariate data plots individual data elements across many dimensions. In Figure 12 is shown a parallel plot of synthetic data set based on the Austrian structural business statistics data (SBS) from 2006 with 10% of outliers (for details about this simulation study see Todorov et al, 2011). The outliers are drawn with darker lines.

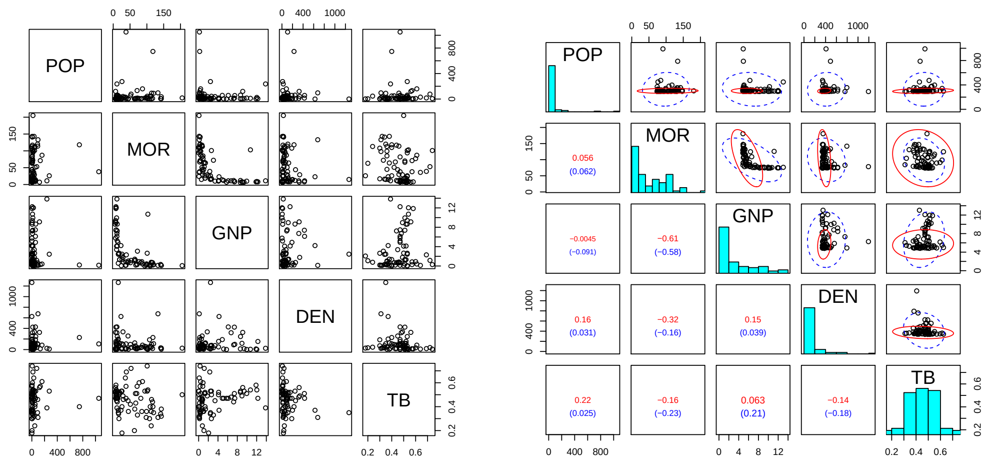


Figure 11: Example of R trellis graphics: a scatterplot matrix of 5 continuous socioeconomic variables from the data set un86(left) and MCD scatterplot matrix of the same variables (right)

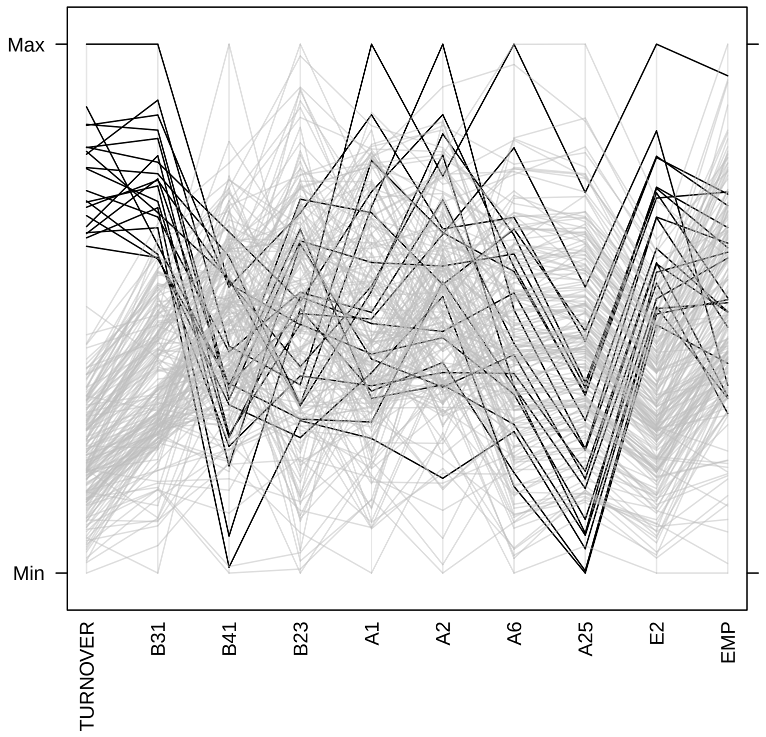


Figure 12: Parallel coordinate plot of a typical data set (from a simulation study) with 10% of outliers (represented by the dark lines).

Conditional plots: The scatterplot matrices as well as the parallel coordinate plots can be conditioned on a categorical variable (or a continuous one which is discretized). Such examples are shown in Figure 13. Three variables from the UNIDO Industrial statistics database at 2-digit level of ISIC (INDSTAT2), namely *Employment*, *Wages and Salaries* and *Gross Output* for one country (Austria) are displayed for three different manufacturing sectors (*Wood products*, *Textiles* and *Food and beverages*).

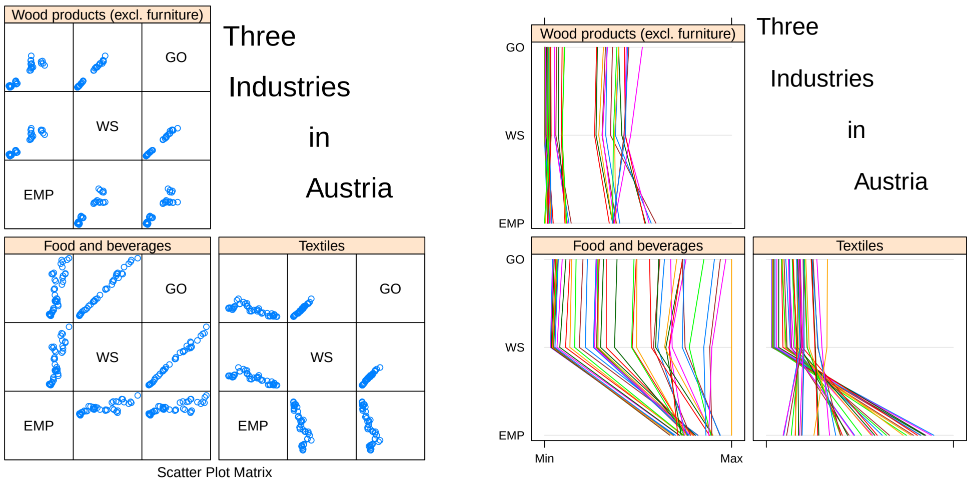


Figure 13: Example of R trellis graphics: conditional scatter plot matrices(left) and parallel coordinate plot (right)

A more sophisticated conditioning plot (see Chambers and Hastie, 1992, Chapter 3. Data for models) which can be drawn with only one call to the function `coplot()` is shown in Figure 14. It presents the *Value added* against the *Number of Employees* for Austria conditioned on the different manufacturing sectors. In the upper panel the categorical variable is shown also. The panels are ordered from lower-left to upper- right, related to the values of the conditioning variable in the upper panel (ISIC) from left to right. Thus, the lower-left plot is for the first ISIC (Food and beverages) and the upper right plot is for the last considered ISIC (Wood products). This coplot shows clearly that the relation between the number of employees and the resulting Value added is very different in the different sectors.

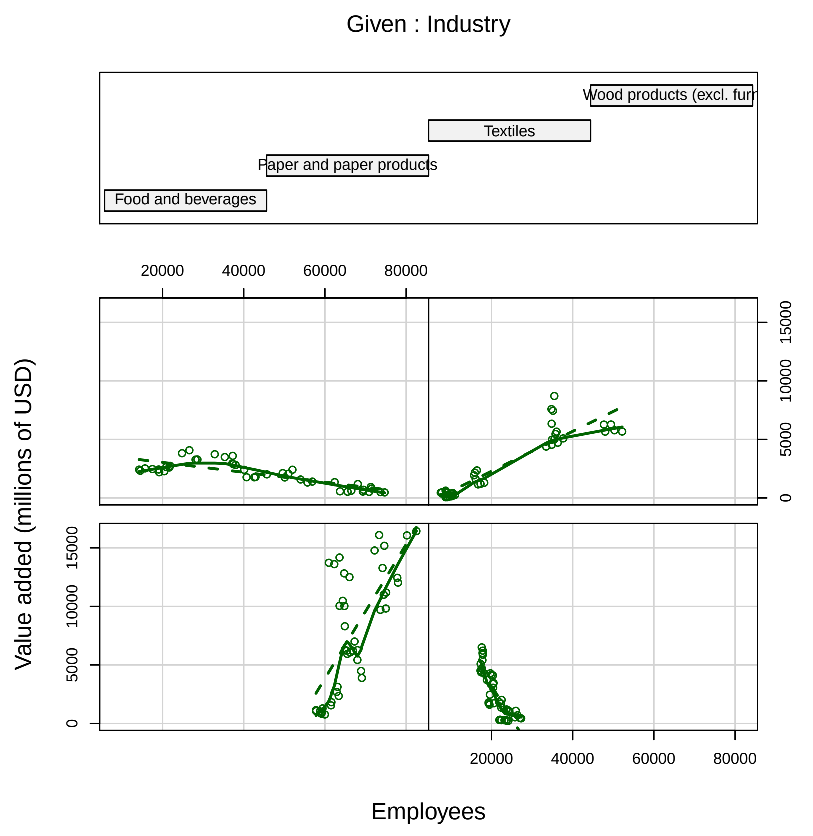


Figure 14: Example of R trellis graphics: conditioning plot (coplot)

As already mentioned it is not necessary that the conditioning variable is discrete. An example of a coplot with a numerical *given* variable is shown in Figure 15. The data are from Atkinson and Riani (1985) and represent three measurements (the girth, height and volume of timber) of 32 trees. The conditioning variable is the girth and the two other variables are presented in log scale. The following R code generates the graph:

```
> require(stats)
> require(graphics)
> coplot(log(Volume) ~ log(Girth) | Height, data = trees,
+     panel = panel.smooth)
```

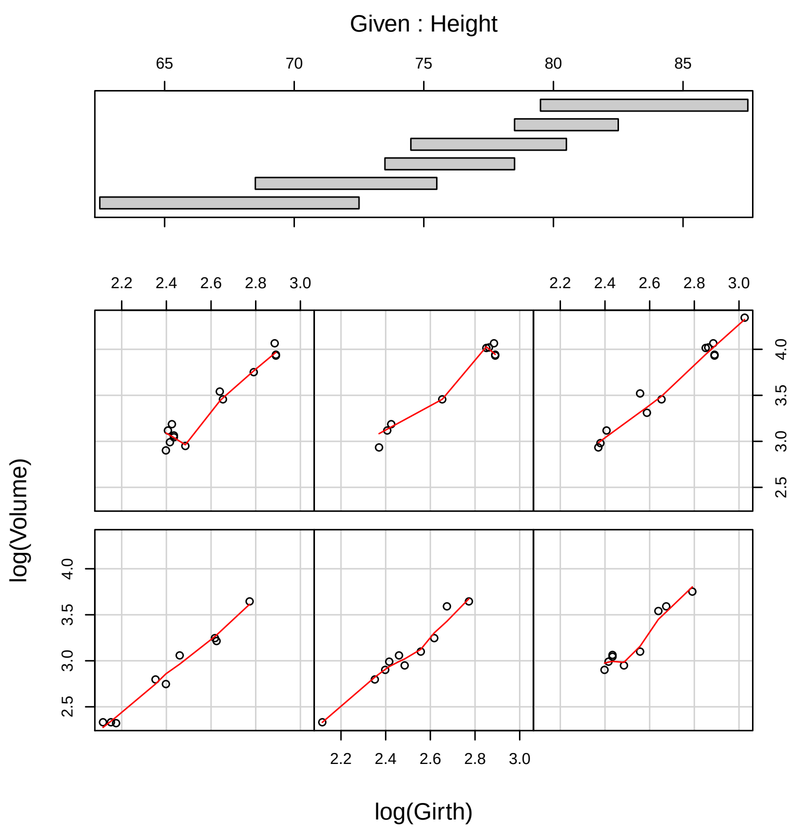


Figure 15: Example of R trellis graphics: conditioning plot (coplot) for the `trees` data set.

3.1.2 Storing the graphical output

R can produce graphics in many formats, including:

- On screen
- PS and PDF files for including in LaTeXand pdfLaTeX or for direct distribution
- PNG or JPEG bitmap formats for the WEB
- On Windows, metafiles (`wmf`—*WindowsMetaFile* and `emf`—*Enhanced MetaFiles*) can be created to be used in Word, PowerPoint, and similar programs

For example to save a plot to a PDF file the following commands are necessary.

```
> pdf(file="graph10.pdf")
> plot(graph10)
> dev.off()
```

To produce exactly the same output in `PNG` format one can use the following commands:

```
> png(file="graph10.png")
> plot(graph10)
> dev.off()
```

Alternatively one could plot the graphical result on screen and then call the function `savePlot()` to save it to a file as shown in the next example:

```
> plot(graph10)
> savePlot(file="graph10", type="pdf")
```

An excellent reference to R Graphics is the book of Paul Murrell (Murrell, 2005), a member of the R Core Development Team who has not only been the main author of the grid package but has also been responsible for several recent enhancements to the underlying R graphics engine.

The best way to visualize the potential of R when it comes to producing publication quality graphics is to type `demo(graphics)` at the R prompt and then navigate through a list of example plots. Another exciting example of the R graphics capabilities is the *R Graph Gallery*—`http://addictedtor.free.fr/graphiques/`, which aims to present many different graphs that are fully created with the programming environment R. Graphs are gathered in a MySQL database and browsable through PHP.

3.2 Example: Screening Functions for of Industrial Statistics Data

Let us consider an example from the maintenance work flow of the UNIDO statistical system. An index is defined for the ratio of the variables *Value added* (`VA`) and *Gross output* (`GO`). For each country and ISIC code, the values of the time series are transformed in the following manner

$$e_{(4)} = \Delta e^{VA/GO \cdot 10} \tag{2}$$

Here Δ denotes the differencing operator, i.e. $\Delta(x_t) = x_{t+1} - x_t$ for $t = 1, \ldots, T-1$. The editing rule is defined as follows.

For each value with necessary condition $VA/GO > 0$ a value is declared as possible error when $e_{(4)} > 30$.

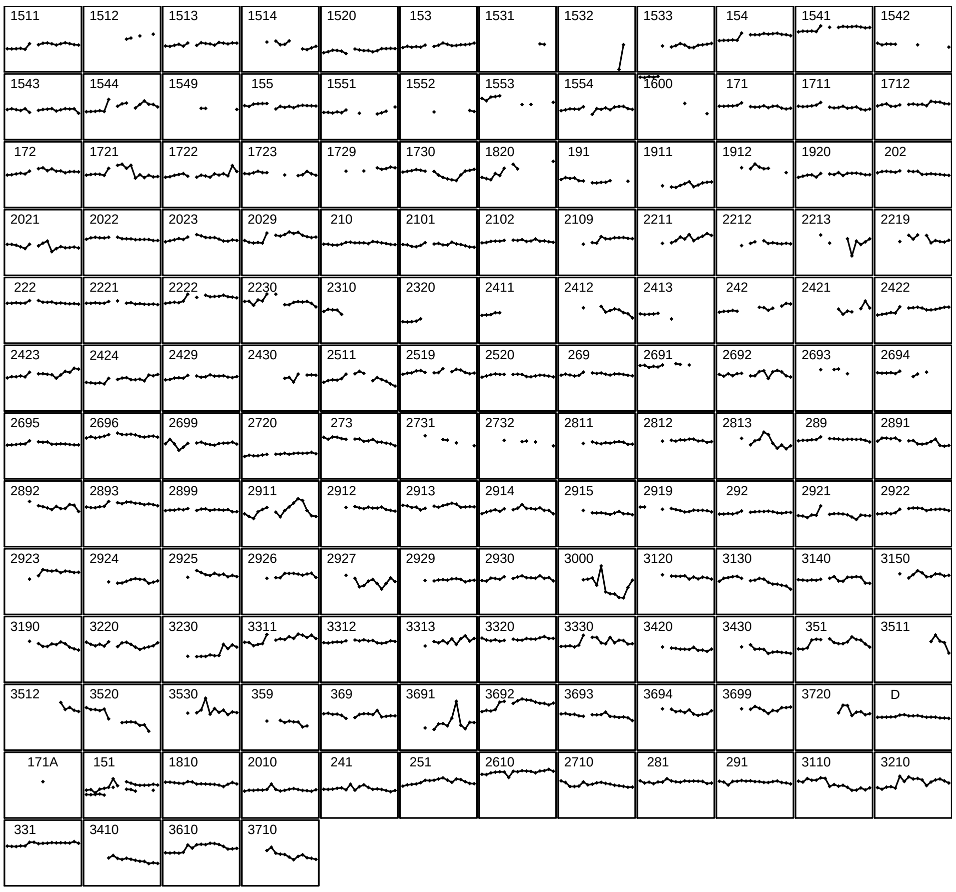


Figure 16: Time Series of the ratio VA/GO for the Austria data.

This threshold has been fixed by subject matter specialists in order to flag as few as possible "good" values as outliers and to flag possibly all erroneous values.

Figure 16 shows the VA/GO ratio for Austria from 1990 until 2006. Figure 17 shows the transformed time series as given in Equation 2. The upper line corresponds to the threshold given (30). Comparing this Figure 16 with Figure 17 it is easy to see that the possible erroneous values are somehow weighted by the transformation, i.e. possible outliers are clearly visible as outliers in Figure 17. This can be seen, for example, by looking at ISIC 3000 or 3691.

For further details about the graphical screening tools see Todorov and Templ (2010).

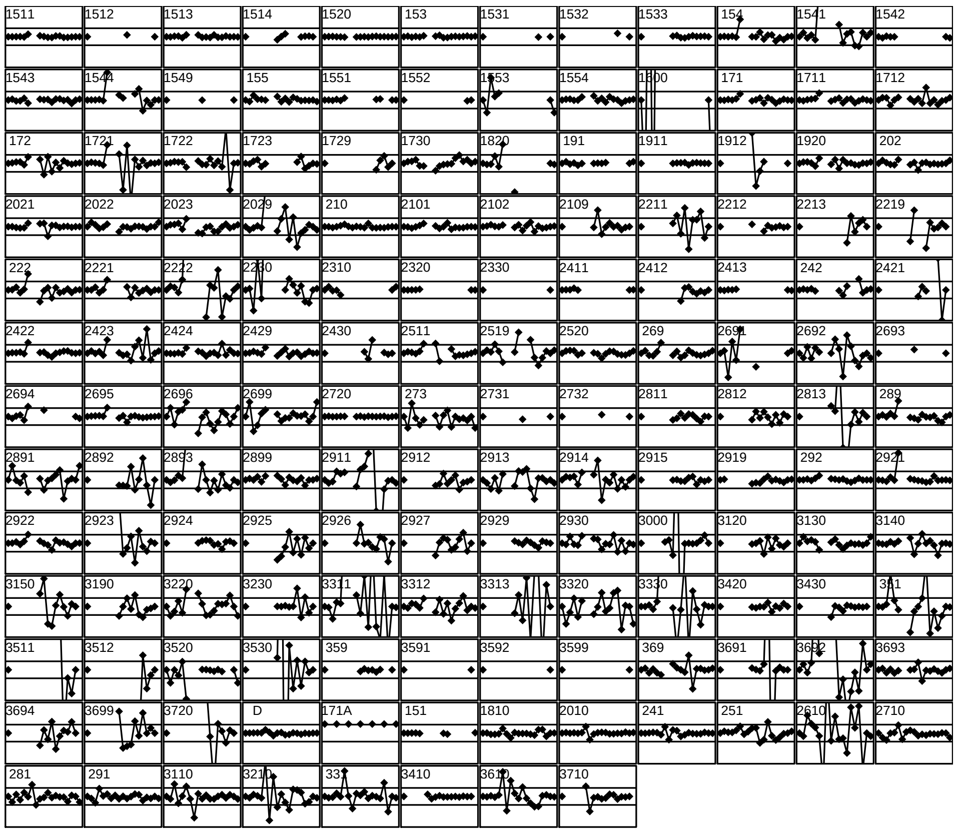


Figure 17: Transformed time series. Values outside the inner band are possible outliers.

3.3 Example: The International Yearbook of Industrial Statistics

The *International Yearbook of Industrial Statistics* published by UNIDO—see http://www.unido.org/index.php?id=o3544—is a unique and comprehensive source of information, the only international publication providing economists, planners, policymakers and business people with worldwide statistics on current performance and trends in the manufacturing sector. The Yearbook is designed to facilitate international comparisons relating to manufacturing activity and industrial development and performance. It provides data which can be used to analyze patterns of growth and related long term trends, structural change and industrial performance in individual industries. Statistics on employment patterns, wages, consumption and gross output and other key indicators are also presented. In its Part I the yearbook contains summary tables for the manufacturing sector (Part 1.1) and for the manufacturing branches (Part 1.2) and the second part consists of country tables. Recently a new section was added to Part I containing analysis of the major trends of growth and distribution of manufacturing in the world. This section is illustrated with many graphical displays based on the summary tables in Part 1.1. An example from Yearbook 2010 showing the growth of MVA in the industrialized regions as a line plot is presented in Fig. 18.

The production line for the yearbook receives as an input the 'raw' data from the UNIDO database but also from other external sources and generates as an output a ready-to-publish PDF file which is submitted to the publishing house. Most of the components of the system are developed in SAS (historically, migrated from the Mainframe) or in .Net (all new development, interfacing to the SAS-based subsystem and generating the tabular output using CrystalReports). The new graphical section consists of several

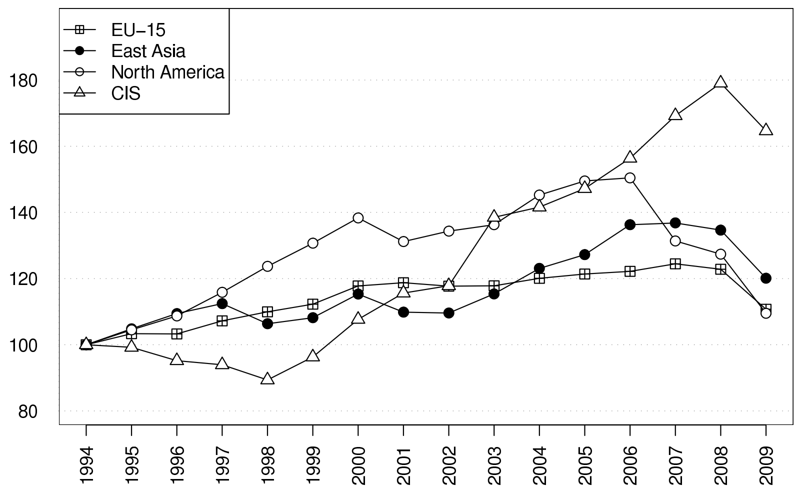


Figure 18: Growth of MVA in industrialized regions, at constant 2000 USD (1994=100)

Most of the industrialized countries have suffered a severe decline in their industrial production in the last two years or so. The worst affected region has been North America, where according to UNIDO estimates, manufacturing output has fallen by 20 per cent since 2007. CIS countries, which recovered just at the beginning of this decade from the turmoil of the 1990s and became one of the fastest growing regions, have also suffered the set back from the recent financial crisis.

pages of graphics and text typeset in two columns in landscape format. The software for the production of this output should fulfill several key requirements:

1. To be able to create publication quality graphics. Although this is possible to do with other packages too, R provides the most flexible solution.

2. To interface easily with the other components of the production line, i.e. with SAS and with the Sybase database. This types of interfaces were described in more detail in Section 2.

3. To comply with the submission guidelines of the publisher—one of the most important issues is that the final document must contain only embedded fonts. For this purpose the function `embedFonts()` is used.

4. To provide means for easy text and image placement. Whenever the data are changed the document should be (preferably automatically) regenerated.

5. To use the same fonts in figure labels as in the main document—this is desirable for reasons of consistency and aesthetics. Sometimes it is possible to match or to approximate the document font from within the data analysis program (e.g. SAS) when the figure is saved. However this would not be ideal because the document fonts themselves might not be constant and the graphics quality will never equal that of which LaTeX is capable.

6. To be easy to maintain and extend. Every year new graphs are added, other are removed and the textual explanations are rewritten to correspond to the new display.

3.4 TeX, LaTeX, BibTeX and Sweave

TeX: TeX is a typesetting system, a computer program for producing nicely printed, publication quality output, even for difficult material such as mathematics, which is freely available. TeX was invented by Professor Donald E. Knuth, at Stanford University, when he was planning to publish a series of books on computing. He discovered that the classical process of typesetting books can be replaced by a new modern computer based methodology and decided to spend a year for building the necessary system. His final solution was the program TeX which allows ordinary authors to produce beautiful output of top quality.

LaTeX: The tool actually used in our examples (and in the production line that we describe) is LaTeX which is a component designed to shield the author from the details of TeX. LaTeX was developed by Lamport (1994) who has recognized that despite all the advantages of TeX the normal user will not feel comfortable in the details of its underlying machinery. Thus LaTeX is a wrapper of TeX that allows the author for example simply to put something that is to be written exactly as it is typed between paired begin and end verbatim commands as well as frees him from the decision as to what type face to use, what size, what style, and so on. It is a high-quality typesetting system which includes features designed for the production of technical and scientific publications and is the de facto standard for the communication and publication of scientific documents. The software itself is available for free (under the terms of the LaTeX Project Public License) from `http://www.latex-project.org/ftp.html` for Linux, MacOs and Windows.

BibTeX: This is a simple tool to create a bibliography in a LaTeX document. To use BibTeX it is necessary to (i) create a bibliography database (.bib) file containing all the required references in the specific BibTeX format. There are tools for doing this and for sharing the reference database with the colleagues researches (see Todorov, 2009); (ii) define the style and the location of the bibliography within the .tex file; (iii) run LaTeX and BibTeX over the document. The main advantage is that a uniform style is achieved, which can easily be replaced by another—the author does not need to take care about this. At the same time a unified library of references is created and the same references can be used in many different publications which can be shared among the team. An example of BibTeX entries is given below:

```
todorov:book,
    TITLE = {Multivariate Robust Statistics: Methods and Computation},
    AUTHOR = {Valentin Todorov},
    YEAR = 2009,
    PUBLISHER = {Südwestdeutscher Verlag für Hochschulschriften},
    ADDRESS = {Saarbrücken},
}

todorov-oof,
    AUTHOR     = {Valentin Todorov and Peter Filzmoser},
    TITLE      = {An Object Oriented Framework for Robust Multivariate
                   Analysis},
    JOURNAL    = JSS,
    PAGES      = {1--47},
    YEAR       = {2009},
    VOLUME     = {32},
    NUMBER     = {3},
    URL        = {http://www.jstatsoft.org/v32/i03/},
}
```

`Sweave`: A suitable tool that allows to embed the code for complete data analysis in documents is the R function `Sweave()` (see Leisch, 2002). The purpose of this tool is to create dynamic reports, which can be updated automatically if data or analysis change. The report is not created statically by inserting graphics and tables prepared in advance and writing the text around them. The necessary programming code for obtaining of the graph is contained in the master document and is written in R while the text is written in LaTeX. The complete document is run through R and all the data analysis is performed on the fly generating the output—tables, graphs, etc. and inserting them into a final LaTeX document.

The idea behind `Sweave` is as follows: A document will be created and at certain positions in this document commands which calculate statistics, create tables or draw graphics will be placed. The results of these commands will be included into the document.

Whenever necessary (e.g. the input data has changed or the text of the document has changed) R will be run on the document and the output will be a ready, always up-to-date document containing all the required results, tables and graphics. The commands for R will be issued using the function `Sweave()` with a syntax resembling the *Noweb* syntax—a simple literate programming tool which allows to combine program source code and the corresponding documentation into a single file—(see Ramsey, 1998). The R commands are written in blocks (*Chunks*) which consist of three main parts: (i) begin markup "<< ... >>", (ii) command part which contains normal R code and (iii) end markup "@". The output report can be automatically updated if data or analysis change, which allows for truly reproducible document generation. There are two kind of operations: (i) *weave*: typeset documentation together with code and (ii) *tangle*: extract code chunks.

Assuming LaTeX and R are installed, there is no need for installation of `Sweave`. It is distributed with R and is included in the **utils** package (no need to load it). At the same time there is no need to learn new languages since in the documentation part LaTeX is used and in the code part the programming language R is used. More information on Sweave, as well as manuals, examples and including publication by the author of Sweave can be found at: `http://www.statistik.lmu.de/~leisch/Sweave/`.

3.5 Example: Bhutan Labour Market Information Bulletin

The Labour Market Information Division under the Department of Employment of Ministry of Labour and Human Resources in Bhutan publishes a *Labour Market Information Bulletin* (currently the fourth issue is available—Royal Government of Bhutan (2009)). The information presented in the bulletin is an outcome of an analysis carried out from the data collected and tabulated from various sources like Labour Force Survey 2009, Job Prospecting Report 2009, Nationals Statistical Bureau, Ministry of Education, Royal Civil Service Commission, Royal University of Bhutan, etc. The data was tabulated using SPSS and then the graphs were created with MS Excel. The text of the bulletin, the data tables and the graphs were built together using MS Word. As already mentioned above, this is the standard way of creating statistical reports but it is error prone and has many drawbacks the main one being that whenever something changes the whole procedure has to be repeated manually again. The framework based on the tools described above allows to create such type of reports with easiness and recreate them as often as necessary without any manual effort. We will consider one typical page of the report consisting of text, a table and a bar chart shown in Figure 19. The information necessary to build this page are two cross tables: the number of unemployed by gender and area of residence and the number of employed persons tabulated by the same two criteria. These data allows to calculate the unemployment rate in the given categories.

The two tables are shown below.

```
Number of unemployed

      Male Female Total
Urban 1700   3700  5400
Rural 2700   4800  7500
Both  4400   8500 12900

Unemplyment rate

      Male Female Total
Urban  4.2   11.6   7.5
Rural  2.1    3.8   3.0
Both   2.6    5.3   4.0
```

The LaTeX source and the R script necessary to reproduce the page are shown in Figure 20. The table could be hand-crafted in R or could be created using the package **xtable** (Dahl, 2009) but we prefer to use the function `latex()` from the package **Hmisc** (Jr and with contributions from many other users., 2009) which provides much more flexibility. The final result is shown in Figure 21.

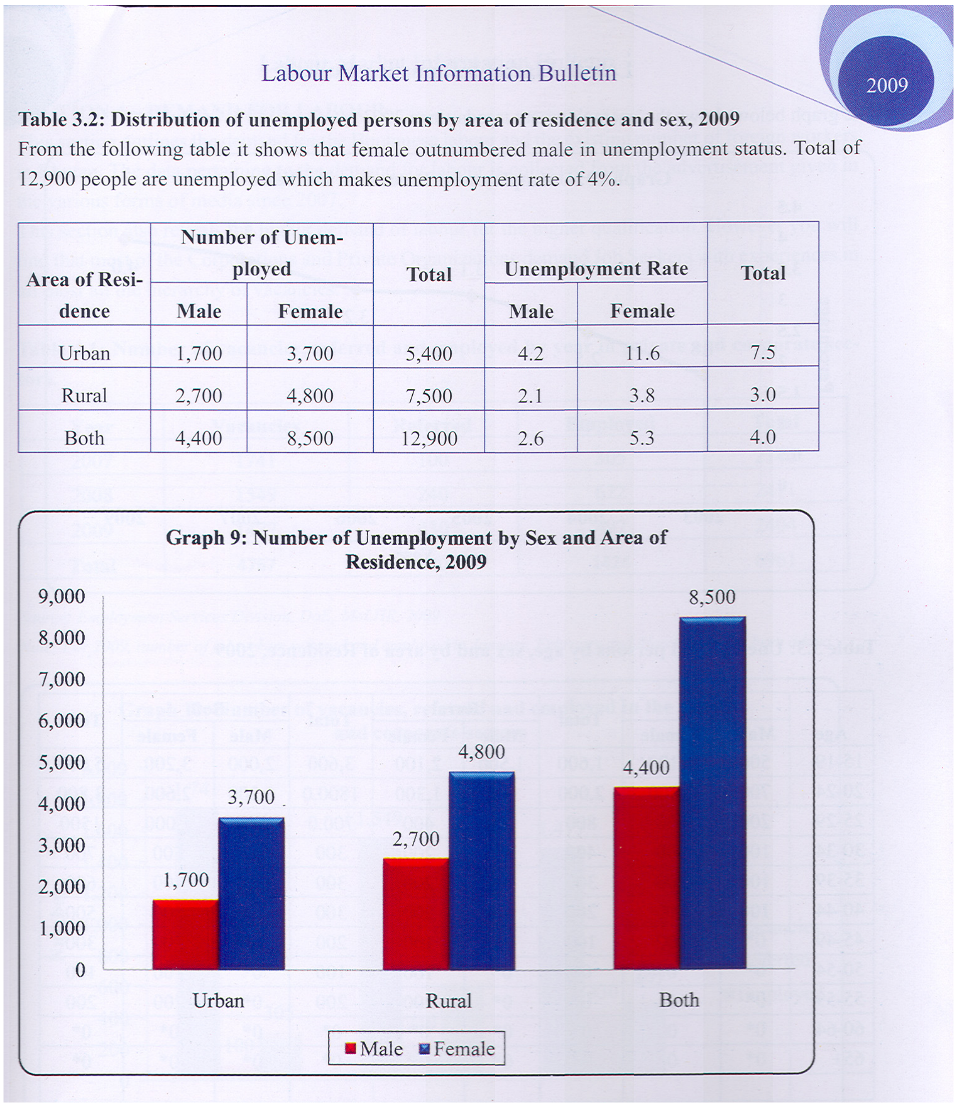
The source code for the example shown in Figure 20 is also contained in a so called vignette in the folder `~ritso/inst/doc` of the **ritso** package. The PDF document can be viewed by calling:

```
> vignette("bhutan-lmi", package = "ritso")
```

A vignette is a document which provides a task-oriented description of package functionality. An R package can contain one or more vignettes. The PDF version of a vignette for an installed package can be accessed from inside R using the `vignette()` function as shown in the above example. The function `browseVignettes()` will open a web browser with links to the vignette PDF as well as a plain-text R file containing the code used in the vignette. The vignette files, both the PDF and the Rnw sources document, are located in the doc directory of an installed package. One can discover the location of an installed package as follows:

```
> system.file(package = "rrcov")

[1] "C:/R/R-2.12.0/library/rrcov"
```

For more information about package vignettes see Leisch (2003).



Labour Market Information Bulletin

2009

Table 3.2: Distribution of unemployed persons by area of residence and sex, 2009

From the following table it shows that female outnumbered male in unemployment status. Total of 12,900 people are unemployed which makes unemployment rate of 4%.

Area of Residence	Number of Unemployed		Total	Unemployment Rate		Total
	Male	Female		Male	Female	
Urban	1,700	3,700	5,400	4.2	11.6	7.5
Rural	2,700	4,800	7,500	2.1	3.8	3.0
Both	4,400	8,500	12,900	2.6	5.3	4.0

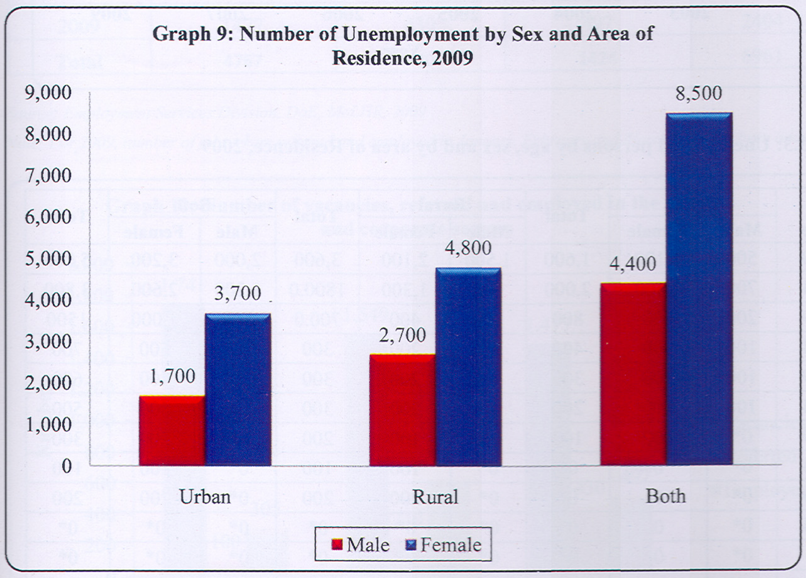


Figure 19: An example from the *Labour Market Information Bulletin, 2009* of the Royal Government of Bhutan—a typical page containing text, a table and a graph. This page will be reproduced using R, LaTeX and Sweave.

```
\documentclass[a4paper]{article}
\usepackage{ctable}
\usepackage{here}
\begin{document}
\thispagestyle{empty}
<<label=bhutan:data, echo=FALSE>>=
xemp <- matrix(c(38500, 123900, 28100, 122300), nrow=2)
xune <- matrix(c(1700, 2700, 3700, 4800), nrow=2)
sxemp <- addmargins(xemp)
sxune <- addmargins(xune)
colnames(sxemp) <- colnames(sxune) <- c("Male", "Female", "Total")
rownames(sxemp) <- rownames(sxune) <- c("Urban", "Rural", "Both")
sxtot <- sxemp + sxune
srate <- round(100*sxune/sxtot,1)
stab <-  cbind(sxune, srate)
@
\paragraph{Table 3.2: Distribution of unemployed persons
by area of residence and sex, 2009}
The following table shows that female outnumbered male in
unemployment status. Total of 12900 people are unemployed
which makes unemployment rate of 4\%.

<<label=bhutan:table, echo=FALSE, results=tex>>=
library(Hmisc)
latex(stab, file = "", cgroup = c("Number of Unemployed", "Unemplyment Rate"),
n.cgroup = c(3, 3), cgroupTexCmd="bfseries", colnamesTexCmd="bfseries",
first.hline.double=FALSE, vbar=TRUE, title="", here=TRUE)
@
\begin{figure}[h!]
\setkeys{Gin}{width=0.9\textwidth}
\begin{center}
<<label=bhutan-barplot, fig=TRUE, echo=FALSE, width=10, height=6>>=
cl <- c("red", "blue")
x <- t(sxune[, -3])
m1 <- barplot(x, beside=TRUE, col=cl, ylim = c(0,9500), axes=FALSE)
axis(2, at=seq(0,9000,1000))
legend(4, -1200, legend=rownames(x), fill=cl, ncol=2, xpd=TRUE)
text(as.vector(m1), as.vector(x)+ mean(x)/20,
    labels=format(as.vector(x)), xpd = TRUE)
title(main="Number of Unemployment by Sex and Area of
\nResidence, 2009")
box()
box(which="outer")
@
\end{center}
\end{figure}
```

Figure 20: The script used to reproduce the example page from the *Labour Market Information Bulletin, 2009* of the Royal Government of Bhutan, shown in Figure 19.

Table 3.2: Distribution of unemployed persons by area of residence and sex, 2009 The following table shows that female outnumbered male in unemployment status. Total of 12,900 people are unemployed which makes unemployment rate of 4%.

	Number of Unemployed			Unemplyment Rate		
	Male	Female	Total	Male	Female	Total
Urban	1700	3700	5400	4.2	11.6	7.5
Rural	2700	4800	7500	2.1	3.8	3.0
Both	4400	8500	12900	2.6	5.3	4.0

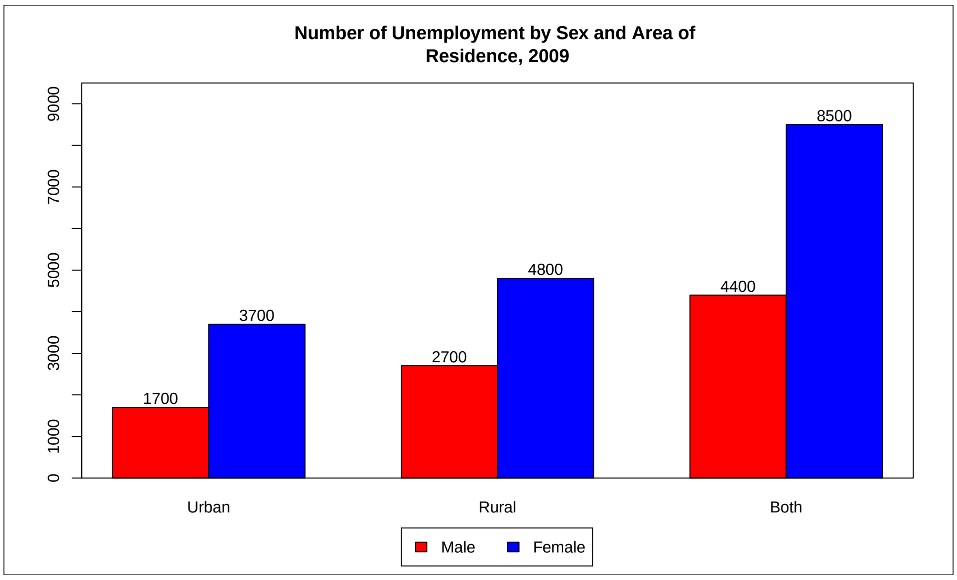


Figure 21: The example page from the Labour Market Information Bulletin, 2009 of the Royal Government of Bhutan, shown in Figure 19, reproduced using R, LaTeX and Sweave.

4 Modeling and analytical capabilities of R

One of the most popular tools for data analysis is still the classical linear regression model given by Equation 3.

$$y_i = x_i^\top \beta + e_i, \quad (i = 1, \ldots, n) \tag{3}$$

where y_i stands for the response variable and x_i are the regressors (independent or exploratory variables) collected in the design matrix $\mathbf{X}$. If the first column of the matrix $\mathbf{X}$ is a vector of ones, the constant term will be denoted by β_0. Classical theory assumes that e_i are distributed normally with mean 0 and variance σ^2. The goal is to estimate the parameters β_i and σ from the n available observations which results in the estimates $\hat{\beta}_i$ and $\hat{\sigma}$.

In R models are fitted by a model fitting function like `lm()` which gets a *formula* that describes the model and a data frame containing the data as parameters and returns an object containing the estimates. Information from this object can be obtained by the functions `summary()`, `residuals()`, `coefficients()`, `predict()`. Let us consider an example with the *Wages and Hours* data set from DASL at `http://lib.stat.cmu.edu/DASL/Datafiles/wagesdat.html` which is also available in the package **ritso** as the R data set `wages`. The data are from a 1966 national survey of 6000 households with a male head earning less than USD 15,000 annually. The households are classified into 39 demographic groups. In the data there are missing values and for the sake of this example we remove all observations where one or more of the variables have missing values. Thus we remain with a data set of 28 observations and 10 variables. The goal is to estimate the labour supply (i.e. average hours worked during the year) from the available nine independent variables. We start with a simple linear regression to investigate the relation between y representing the labour supply and x being the average age of the respondents. The formula provided to the function `lm()` is `HRS ~AGE` specifying the investigated relationship. The same formula can be used for plotting in the function `plot()` which produces a graph as the one shown in panel (a) of Figure 22.

```
> library(ritso)
> data(wages)
> names(wages)
```

```
 [1] "HRS"    "RATE"   "ERSP"   "ERNO"   "NEIN"   "ASSET"
 [7] "AGE"    "DEP"    "RACE"   "SCHOOL"
```

```
> x <- as.matrix(wages)
> ok <- is.finite(x %*% rep(1, ncol(x)))
> wages <- wages[ok, , drop = FALSE]
> wages.lm <- lm(HRS ~ AGE, data = wages)
> plot(HRS ~ AGE, data = wages)
> abline(wages.lm)
```

The function `lm()` returns an object of type `'lm'` containing the estimated model. This object is essentially a list and contains the estimated coefficients $\hat{\beta}_i$, residuals r_i, scale $\hat{\sigma}$, etc.

```
> class(wages.lm)
```

```
[1] "lm"
```

The names of the elements of the list are

```
> names(wages.lm)
```

```
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

The function `summary()` can be used for retrieving even more elements of the fitted model (standard errors, t-statistics and the corresponding p-values, R^2 and F-statistic).

```
> summary(wages.lm)

Call:
lm(formula = HRS ~ AGE, data = wages)

Residuals:
    Min      1Q  Median      3Q     Max
-106.48  -45.54   16.53   36.36  117.40

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2164.6004    85.7121  25.254   <2e-16 ***
AGE           -0.3855     2.1719  -0.178     0.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 55.39 on 26 degrees of freedom
Multiple R-squared: 0.00121,        Adjusted R-squared: -0.0372
F-statistic: 0.03151 on 1 and 26 DF,  p-value: 0.8605
```

It is clearly seen from the scatter plot shown in Figure 22a that there are two outliers (of course it is easy to spot them by eye in the case of simple regression but this could be a challenge if we had multiple independent variables)—the observations with numbers 3 and 4 represent demographic groups with too low and too high average age. Even if these observations were not typing errors, it is clear that they deviate strongly from the bulk of the data and pull the OLS line into almost perpendicular direction. It is often believed that it would be enough to perform regression diagnostics using the (standardized) regression residuals in order to identify any possible outliers in the data. Unfortunately this is seldom the case, since the outliers have the property to *mask* themselves by showing not very large standardized residuals which is seen in Figure 22b. To cope with this problem it is recommended to use *robust* regression method (see Rousseeuw and Leroy, 1987; Maronna et al, 2006). The two lower panels of Figure 22 show the results obtained by robust MM regression with the function `lmrob()` from the package **robustbase**. Information on statistical modeling in R going beyond the simple and multiple regression, like quantile regression, generalized linear models, models for time series and panel data can be found in the book Kleiber and Zeileis (2008).

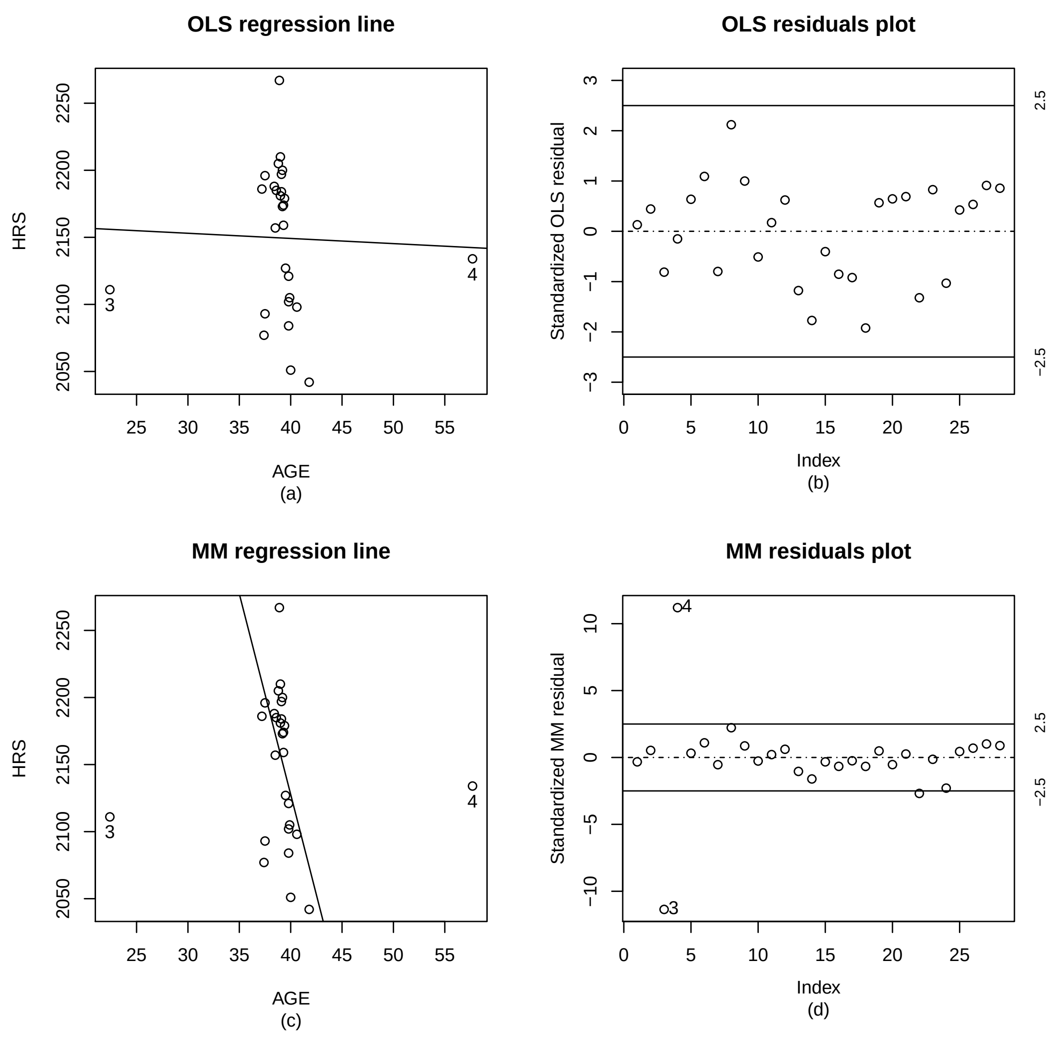


Figure 22: Classical and robust linear regression for the `wages` data set: (a) and (c) are scatter plot of the 28 observations with non-missing values with an OLS and MM line respectively; (b) and (d) OLS and MM residual plots. The latter indicates clearly the two outliers while in the former they are masked.

4.1 Example: Nowcasting MVA for Cross-country Comparison

The Research and Statistics Branch of the United Nations Industrial Development Organization (UNIDO) is responsible for implementing the international mandate of the Organization in the field of industrial statistics. It maintains a unique industrial statistics database and updates it regularly with data collected from the National Statistical Offices (NSO). A separate database at macro level is also maintained primarily for compilation of statistics related to manufacturing value added (MVA) such as its growth rate and share in gross domestic product (GDP) for various countries and regions. These figures are published in the *International Yearbook of Industrial Statistics* and posted on the statistical pages of the UNIDO web site. For current economic analysis it is crucial that the Yearbook presents MVA data for the most recent years. Because of a time-gap of at least one year between the latest year for which data are available and the year for which MVA data must be reported in the Yearbook, nowcasting methods are used to fill in the missing data up to the current year. For this purpose a parsimonious methodology was proposed exploiting the relationship between MVA and GDP, together with the availability of reliable estimates of GDP growth rates from external sources, to produce reliable nowcasts of MVA.

The standard OLS estimator may be biased because of a violation of the assumption of exogeneity of the regressors with respect to the error term and because of the presence of outliers in the data. The time series plots of the growth of ***GDP*** and ***MVA*** indicate the presence of outliers for some countries. Fig. 23 illustrates this for the 1991-2007 data for Poland. In the transition years 1991 and 1992 the MVA and GDP growth rates are extreme. In 1991 and 1992 the MVA (GDP) growth rates equal -16.3% (-7.0%) and 80.2% (2.6%), respectively. For all other years the MVA (GDP) growth rates are between -0.6% (1.2%) and 13.8% (7.1%). The estimation of the regression models using OLS is known to be problematic in the presence of outliers. As can be seen in Fig. 24, the 1992 observation tilts the OLS slope estimate to its position and yields a distorted estimate of the regression line fitting the bulk of the data. The OLS estimator thus does not satisfy the requirement that the influence of single observations on the nowcast should be small. For this reason, we also consider a robust alternative to the OLS estimator, namely the MM—estimator. The robust MM—estimator is a two-step estimator. First,

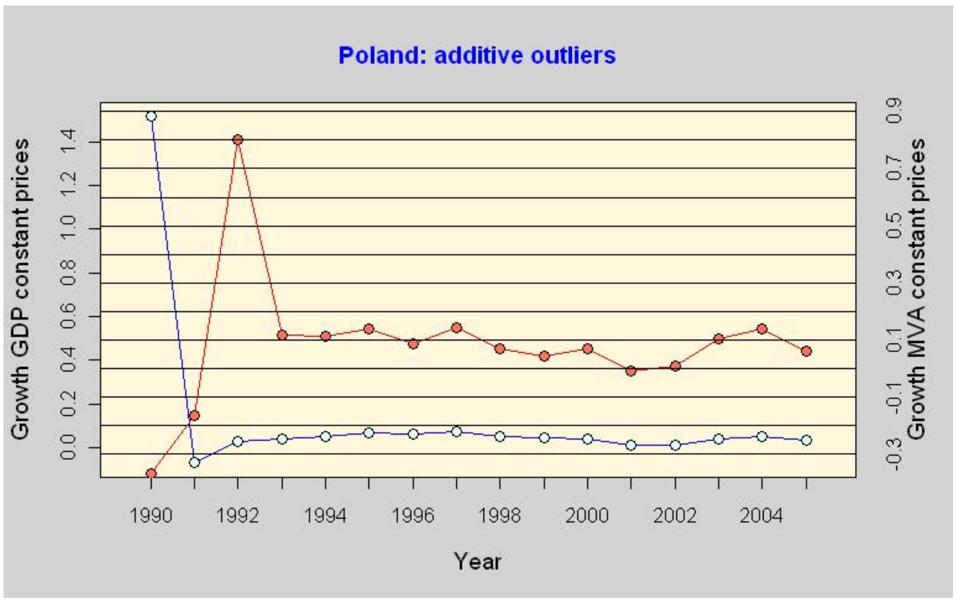


Figure 23: Time series plot of the 1990-2007 annual GDP and MVA growth rates for Poland—to illustrate the presence of additive outliers in the data.

it estimates the parameter vector that minimizes the sum of the 50% smallest squared residuals. This 50% Least Trimmed Squares (LTS) estimate then serves as the starting value for the M-estimation, where a loss function is minimized that downweights outliers. The MM—estimator has a high efficiency under the linear regression model with normally distributed errors. Because it is initialized at the LTS estimates, it is also highly robust to outliers (see e.g. Maronna et al, 2006, Chapter 5). In Fig. 24 the OLS and robust MM—regression estimates are compared. We see that, in contrast with the OLS estimator, the robust MM—estimate is rather insensitive to the outlying observations and produces an accurate fit of the bulk of the data.

The computations are performed using the package **robustbase** (see Rousseeuw et al, 2009) which was developed to provide "essential robust statistics" within R available in a single package and to provide tools that allow analyzing data with robust methods. This includes regression methodology including model selections and multivariate statistics. The goal is to cover the book Maronna et al (2006).

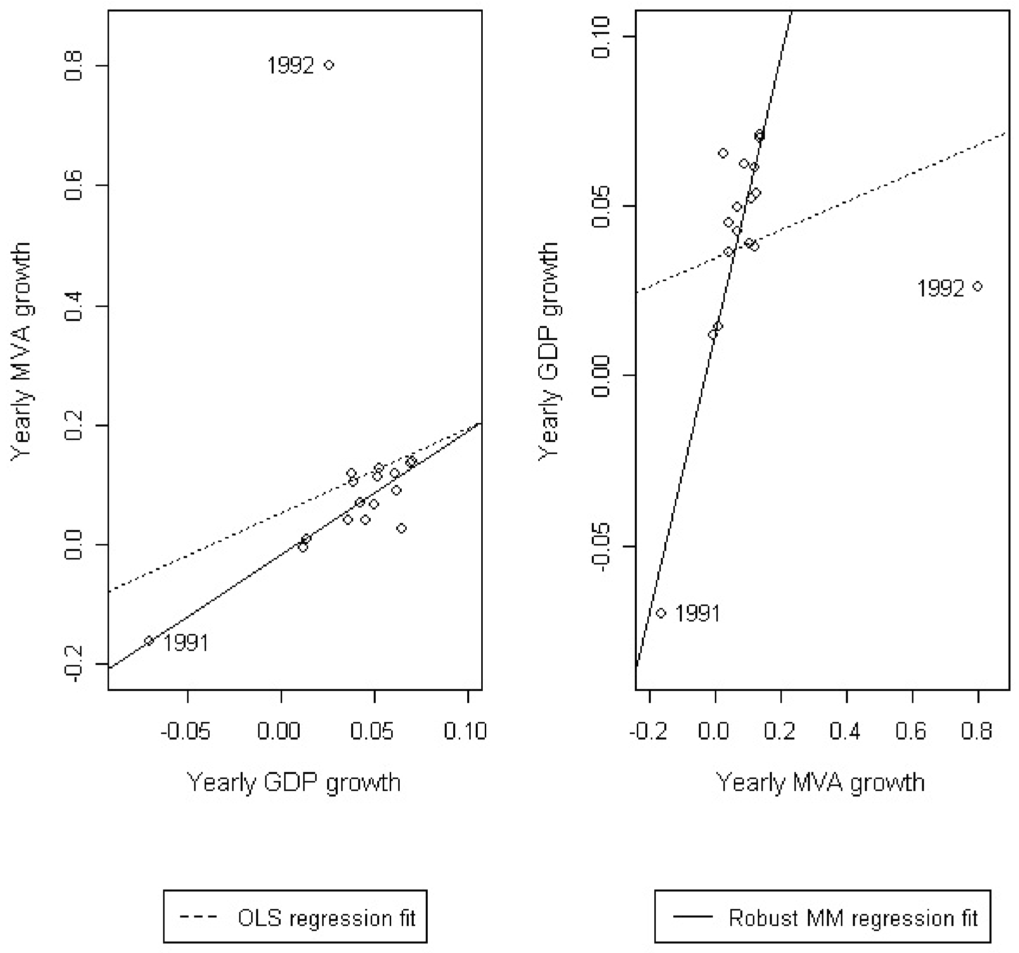


Figure 24: Scatter plots of the 1991-2007 annual GDP and MVA growth rates for Poland, together with the OLS and robust MM—regression fit.

The nowcast accuracy comparison made showed that the approach using (i) the econometric model that specifies the conditional expectation of the yearly MVA growth rate as a linear function of the contemporaneous GDP growth rate and (ii) a robust estimation method has the best performance of all considered methods. Further details about the methodology and comparison of the accuracy of the nowcasts can be found in Boudt et al (2009).

5 Summary, Conclusions and Outlook

There is an increasing demand for statistical tools which combine the ease of use of the traditional software packages with the availability of the newest analytical methods which is only possible through the flexibility provided by a statistical programming language such as R. We discussed briefly the versatility of the R programming environment and how it is possible to apply this environment in the statistical offices. This was illustrated by examples from the statistical production process of UNIDO where certain steps were either migrated or newly developed in R—the areas of data integration, automatic generation of publication quality graphics for dissemination of statistical data and nowcasting methods to fill in missing data.

There are many more interesting topics which might be worth considering when thinking about using R for working in the area of official statistics but these are out of scope of the current paper. A brief overview of such topics follows, which will constitute a future paper advocating the introduction of R in the national or international statistical offices.

5.1 Handling of missing values and outliers

Missing values and outliers are present in virtually all data sets from Official Statistics because of non-respondences, measurement errors (non-representative outliers) or correctly measured values which are far away from the main bulk of the data (representative outliers). The latter are not edited but they need special care in the estimation process since they may influence estimators too much. Missing values can originate from unit or item non-responses. While for unit non-responses the respondent does not report any value in a battery of questions or all questions, item non-responses occurs when single questions are not answered. While unit non-responses are calibrated by modification of the sampling weights in a minimalistic manner, item non-responses are typically replaced by their estimates. This is called **imputation**. The detection of outliers and values which fails on some known relationships in the data is named **editing**. A number of R packages for missing data imputation exist on CRAN (see Appendix A) but the package **rrcovNA** contains also methods for robust estimation and outlier detection.

5.2 Visualization of missing values—package VIM

Before applying models on the data or producing any statistics, the data has to be explored using graphics to look for possible problems (see Tukey, 1962, 1977). When data include missing values most visualisation tools fail because they are designed to analyze complete data. However, virtually all data sets in Official Statistics include missing values due to measurement errors or non-response. The package **VIM** allows to explore and to analyze data containing missing values. It is also possible to explore the structure of the missing values, as well as to produce high-quality graphics for publications.

5.3 Working with large data sets

One issue R is often criticized for is the fact that it loads the complete data to be processed in memory which might be a serious limitation in some cases. One way to solve this problem is to store the data set in a database and use one of the packages described in Section 2 to access it sequentially. There are also several packages on CRAN for working with large data sets like the package **filehash** which implements a simple key-value style database where character string keys are associated with data values that are stored on the disk and the provided utilities allow to treat the database much like the familiar in R environments and lists. Other packages are **ff**, **bigmemory** and **biglm**.

5.4 Statistical disclosure control

It is a fundamental principle for data providers that values of statistical units such as enterprises or persons are strictly confidential. The UN Economic Commission report on *Fundamental Principles for Official Statistics* (1992) states that:

> *Individual data collected by statistical agencies for statistical compilation, whether they refer to natural or legal persons, are to be strictly confidential and used exclusively for statistical purposes.*

This is not the only reason why organizations such as National Statistical Offices or international organizations like UNIDO should meet this requirements. Also national laws on data privacy have to be respected. On the other hand, not respecting confidentiality downwards the trust of data providers when confidence is no longer true. Statistical disclosure control consists of methods which can be categorized in two different concepts: the anonymization of micro data and the anonymization of cells in tables. However, tables generated from source perturbed micro data can also fulfil confidentiality requirements. Therefore micro data perturbation can be seen as one technique to publish confidential tables.

5.5 Graphical user interface for R

R is provided with a command line interface (CLI), which is the preferred user interface for power users because it allows direct control on calculations and it is very flexible. However, good knowledge of the language is required and the learning curve is typically longer than with a graphical user interface (GUI). The user interface remains the biggest difference between R and S-PLUS or EViews, since those implement a very sophisticated

GUIs. Many R users (or potential R users) are asking for, and would probably benefit from a R GUI. There exist several projects developing or offering the opportunity to develop alternate user interfaces. A Special Interest Group mailing list (R-SIG-GUI) exists also to freely discuss concerned issues. The R (GUI) Wiki is also there to exchange information and ideas related to the use of R GUIs and to start using R. A comprehensive list of the R GUI projects can be found at `http://sciviews.org/_rgui/` with the most prominent being RCommander which is available as a CRAN package **Rcmdr** written in Tcl/Tk (see Fox, 2005).

References

Atkinson AC, Riani M (1985) Plots, Transformations and Regression. Oxford University Press

Baier T, Neuwirth E (2007) Excel::com::r. Computational Statistics 22:91–108, URL http://dx.doi.org/10.1007/s00180-007-0023-6

Boudt K, Todorov V, Upadhyaya S (2009) Nowcasting manufacturing value added for cross-country comparison. Statistical Journal of the IAOS: Journal of the International Association of Official Statistics 26:15–20

Burns P (2010) Spreadsheet addiction. URL http://www.burns-stat.com/pages/Tutor/spreadsheet_addiction.html#addlinks

Chambers JM, Hastie TJ (1992) Statistical Models in S. Wadsworth and Brooks, Cole, Pacific Grove, CA

Dahl DB (2009) xtable: Export tables to LaTeX or HTML. URL http://CRAN.R-project.org/package=xtable, r package version 1.5-6

Daigle G, Rivest L (1992) A robust biplot. The Canadian Journal of Statistics 20(3):241–255

Fox J (2005) The R Commander: A basic-statistics graphical user interface to R. Journal of Statistical Software 14(9):1–42, URL http://www.jstatsoft.org/v14/i09/paper

Ihaka R, Gentleman R (2009) R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics 5(3):299–314

Jr FEH, with contributions from many other users (2009) Hmisc: Harrell Miscellaneous. URL http://CRAN.R-project.org/package=Hmisc, r package version 3.7-0

Kleiber C, Zeileis A (2008) Applied Econometrics with R. Springer-Verlag

Krause A (2009) Taking it to higher dimensions. Statistical Computing and Graphics Newsletter 20:11

Lamport L (1994) LaTeX: A document preparation system: User's guide and reference manual. Addison-Wesley Pub. Co., Reading, Mass.

Leisch F (2002) Sweave: Dynamic generation of statistical reports using literate data analysis. In: Härdle W, Rönz B (eds) Compstat 2002 - Proceedings in computational statistics, Physica Verlag, Heidelberg, pp 575–580

Leisch F (2003) Sweave, Part II: Package vignettes. R News 3:21–24, URL http://CRAN.R-project.org/doc/Rnews/

Maindonald J, Braun J (2007) Data Analysis and Graphics Using R, 2nd edn. Cambridge University Press, Cambridge, iSBN 978-0-521-86116-8

Maronna RA, Martin D, Yohai V (2006) Robust Statistics: Theory and Methods. John Wiley & Sons, New York

Muenchen RA, Hilbe JM (2010) R for Stata Users. Springer Series in Statistics and Computing, Springer, ISBN 978-1-4419-1317-3

Murrell P (2005) R Graphics, 1st edn. Chapman and Hall/CRC

Nic M, Jirat J (2010) Xpath tutorial. URL http://zvon.org/xxl/XPathTutorial/General/examples.html

P J Rousseeuw IR, Tukey JW (1999) The bagplot: a bivariate boxplot. The American Statistician 53:328–387

Ramsey N (1998) Noweb man page. Edu/nr/noweb.version 2.9a.

Rousseeuw P, Croux C, Todorov V, Ruckstuhl A, Salibian-Barrera M, Verbeke T, Maechler M (2009) robustbase: Basic Robust Statistics. URL http://CRAN.R-project.org/package=robustbase, r package version 0.5-0-1

Rousseeuw PJ, Leroy AM (1987) Robust Regression and Outlier Detection. John Wiley & Sons, New York

Royal Government of Bhutan (2009) Labour Market Information Bulletin. Ministry of Labour and Human Resources, URL www.molhr.gov.bt

Sarkar D (2008) Lattice: Multivariate Data Visualization with R. Springer, New York, ISBN 978-0-387-75968-5

Su YS (2008) It is easy to produce chartjunk using Microsoft Excel 2007 but hard to make good graphs. Computational Statistics & Data Analysis 52:4594Ű–4601

Todorov V (2008) R: An open source statistical environment. URL http://www.unece.org/stats/documents/2008.04.msis.htm, presented at the Meeting on the Management of Statistical Information Systems (MSIS 2008) (Luxembourg, 7-9 April 2008)

Todorov V (2009) Virtual teams: Practical guide to wikis and other collaboration tools. UNIDO Staff Working Paper (WP102009), Vienna, URL www.unido.org/fileadmin/user_media/Publications/RSF_DPR/WP102009_Ebook.pdf

Todorov V, Filzmoser P (2009a) Multivariate Robust Statistics: Methods and Computation. Südwestdeutscher Verlag für Hochschulschriften, Saarbrücken

Todorov V, Filzmoser P (2009b) An object oriented framework for robust multivariate analysis. Journal of Statistical Software 32(3):1–47, URL http://www.jstatsoft.org/v32/i03/

Todorov V, Templ M (2010) Screening: Methods and tools for the UNIDO Industrial Statistics (INDSTAT) databases. manuscript

Todorov V, Templ M, Filzmoser P (2011) Detection of multivariate outliers in business survey data with incomplete information. Advances in Data Analisys and Classification To appear, published online 27. October 2010

Tufte ER (1997) Visual Explanations: Images and Quantities, Evidence and Narrative. Graphics Press, Cheshire, Connecticut, 2nd edition

Tufte ER (2001) The Visual Display of Quantitative Information. Graphics Press, Cheshire, Connecticut

Tukey JW (1962) The Future of Data Analysis. The Annals of Mathematical Statistics 33(1):1–67

Tukey JW (1977) Exploratory Data Analysis. Addison-Wesley

United Nations (1986) World statistics in brief

United Nations (2008) International recommendations for industrial statistics (IRIS-2008

Upadhyaya S, Todorov V (2008) UNIDO data quality. UNIDO Staff Working Paper, Vienna

Varmuza K, Filzmoser P (2009) Introduction to Multivariate Statistical Analysis in Chemometrics. Taylor and Francis - CRC Press, Boca Raton, FL

Venables WN, Ripley BD (2003) Modern Applied Statistics with S, 4th edn. Springer

Wegman E (1990) Hyperdimensional data analysis using parallel coordinates. Journal of the American Statistical Association 85:664–675

A Appendix: CRAN Task View on Official Statistics and Survey Methodology

Maintainer: Matthias Templ (version 12-02-2010) This CRAN task view contains a list of packages that includes methods typically used in official statistics and survey methodology. Many packages provide functionality for more than one of the topics listed below. Therefore this list is not a strict categorization and packages can be listed more than once. Please note that not all topics that are of interest in official statistics are listed below, because functionality in R might be missing until now (e.g. for editing).

Complex Survey Design: General Comments:

- Package **sampling** includes many different algorithms for drawing survey samples and calibrating the design weights.
- Package **survey** can also handle moderate data sets and is the standard package for dealing with already drawn survey samples in R. Once the given survey design is specified within the function `svydesign()`, point and variance estimates can be computed.
- Package **simFrame** is designed for performing simulation studies in official statistics. It provides a framework for comparing different point and variance estimators under different survey designs as well as different conditions regarding missing values, representative and non-representative outliers.

Complex Survey Design: Details:

- Package **survey** allows to specify a complex survey design (stratified sampling design, cluster sampling, multi-stage sampling and pps sampling with or without replacement) for an already drawn survey sample in order to compute accurate point and variance estimates.
- Various algorithms for drawing a sample are implemented in package **sampling** (Brewer, Midzuno, pps, systematic, Sampford, balanced (cluster or stratified) sampling via the cube method, etc.).
- The **pps** package contains functions to select samples using pps sampling. Also stratified simple random sampling is possible as well as to compute joint inclusion probabilities for Sampford's method of pps sampling.
- Package **sampfling** implements the Sampford algorithm to obtain a sample without replacement and with unequal probabilities.
- Package **stratification** allows univariate stratification of survey populations with a generalisation of the Lavallee-Hidiroglou method.

Complex Survey Design: Point and Variance Estimation:

- Package **survey** allows to specify a complex survey design. The resulting object can be used to estimate (Horvitz-Thompson-) totals, means, ratios and quantiles for domains or the whole survey sample, and to apply regression models. Variance estimation for means, totals and ratios can be done either by Taylor linearization or resampling (BRR, jackkife, bootstrap or user-defined).

- Package **EVER** provides the estimation of variance for complex designs by delete-a-group jackknife replication for (Horvitz-Thompson-) totals, means, absolute and relative frequency distributions, contingency tables, ratios, quantiles and regression coefficients even for domains.

- Package **laeken** provides functions to estimate certain Laeken indicators (at-risk-of-poverty rate, quintile share ratio, relative median risk-of-poverty gap, Gini coefficient) including their variance for domains and stratas based on bootstrap resampling.

- Package **simFrame** allows to compare (user-defined) point and variance estimators in a simulation environment.

Complex Survey Design: Calibration:

- Package **survey** allows for post-stratification, generalized raking/calibration, GREG estimation and trimming of weights.

- Package **EVER** provide facilities (function `kottcalibrate()`) to calibrate either on a total number of units in the population, on mariginal distributions or joint distributions of categorical variables, or on totals of quantitative variables.

- The `calib()` function in Package **sampling** allows to calibrate for nonresponse (with response homogeneity groups) for stratified samples.

- The `calibWeights()` function in package **laeken** is a possible faster (depending on the example) implementation of parts of `calib()` from package **sampling**.

- Package **reweight** allows for calibration of survey weights for categorical survey data so that the marginal distributions of certain variables fit more closely to those from a given population, but does not allow complex sampling designs.

Imputation: A distinction between iterative model-based methods, k-nearest neighbor methods and miscellaneous methods is made. However, often the criteria for using a method depend on the scale of the data, which in official statistics are typically a mixture of continuous, semi-continuous, binary, categorical and count variables. **EM-based Imputation Methods:**

- Package **mi** provides iterative EM-based multiple Bayesian regression imputation of missing values and model checking of the regression models used. The regression models for each variable can also be user-defined. The data set may consist of continuous, semi-continuous, binary, categorical and/or count variables.
- Package **mice** provides iterative EM-based multiple regression imputation. The data set may consist of continuous, binary, categorical and/or count variables.
- Package **mitools** provides tools to perform analyses and combine results from multiply-imputated datasets.
- Package **Amelia** provides multiple imputation where first bootstrap samples with the same dimensions as the original data are drawn, and then used for EM-based imputation. It is also possible to impute longitudial data. The package in addition comes with a graphical user interface.
- Package **VIM** provides EM-based multiple imputation (function `irmi()`) using robust estimations, which allows to adequately deal with data including outliers. It can handle data consisting of continuous, semi-continuous, binary, categorical and/or count variables.
- Package **mix** provides iterative EM-based multiple regression imputation. The data set may consist of continuous, binary or categorical variables.
- Package **pan** provides multiple imputation for multivariate panel or clustered data.
- Package **norm** provides EM-based multiple imputation for multivariate normal data.
- Package **cat** provides EM-based multiple imputation for multivariate categorical data.
- Package **MImix** provides tools to combine results for multiply-imputed data using mixture approximations.
- Package **robCompositions** provides iterative model-based imputation for compositional data (function `impCoda()`).

k-Nearest Neighbor (knn) Imputation Methods

- Package **yaImpute** performs popular nearest neighbor routines for imputation of continuous variables where different metrics and methods can be used for determining the distance between observations.
- Function `SeqKNN()` in Package **SeqKnn** imputes the missing values in continuously scaled variables sequentially. First, it separates the dataset into incomplete and complete observations. The observations in the incomplete set are imputed by the order of missing rate. Once the missing values in an observations are imputed, the imputed observation is moved into the complete set.

- Package **impute** impute provides knn imputation of continuous variables.

- Package **robCompositions** provides knn imputation for compositional data (function `impKNNa()`) using the Aitchison distance and adjustment of the nearest neighbor.

- Package **rrcovNA** provides an algorithm for (robust) sequential imputation (function `impSeq()` and `impSeqRob()` by minimizing the determinant of the covariance of the augmented data matrix.

Miscellaneous Imputation Methods:

- Package **missMDA** allows to impute incomplete continuous variables by principal component analysis (PCA) or categorical variables by multiple correspondence analysis (MCA).

- Package **mice** (function `mice.impute.pmm()`) and Package **Hmisc** (function `aregImpute()`) allow predicitve mean matching imputation.

- Package **VIM** allows to visualize the structure of missing values using suitable plot methods. It also comes with a graphical user interface.

Statistical Disclosure Control: Data from statistical agencies and other institutions are in its raw form mostly confidential and data providers have to be ensure confidentiality by both modifying the original data so that no statistical units can be re-identified and by guaranting a minimum amount of information loss.

- Package **sdcMicro** can be used for the generation of confidential (micro)data, i.e. for the generation of public- and scientific-use files. The package also comes with a graphical user interface.

- Package **simPopulation** simulates synthetic, confidential, close-to-reality populations for surveys based on sample data. Such population data can then be used for extensive simulation studies in official statistics, using **simFrame** for example.

- Package **sdcTable** can be used to provide confidential (hierarchical) tabular data. It includes the HITAS and the HYPERCUBE technique and uses package **lpSolve** for solving (a large amount of) linear programs.

Seasonal Adjustment: For general time series methodology we refer to the TimeSeries, task view.

- Decomposition of time series can be done with the function `decompose()`, or more advanced by using the function `stl()`, both from the basic **stats** package. Decomposition is also possible with the `StructTS()` function, which can also be found in the **stats** package.

- Many powerful tools can be accessed via package **x12**. It provides a wrapper function and GUI for the X12 binaries (`http://www.census.gov/srd/www/x12a/`) under windows, which have to be installed first.

Statistical Record Matching:

- Package **StatMatch** provides functions to perform statistical matching between two data sources sharing a number of common variables. It creates a synthetic data set after matching of two data sources via a likelihood aproach or via hot-deck.
- Package **RecordLinkage** provides functions for linking and deduplicating data sets.
- Package **MatchIt** allows nearest neighbor matching, exact matching, optimal matching and full matching amonst other matching methods. If two data sets have to be matched, the data must come as one data frame including a factor variable which includes information about the membership of each observation.

Indices and Indicators:

- Package **laeken** provides functions to estimate popular risk-of-poverty and inequality indicators (at-risk-of-poverty rate, quintile share ratio, relative median risk-of-poverty gap, Gini coefficient). In addition, standard and robust methods for tail modeling of Pareto distributions are provided for semi-parametric estimation of indicators from continuous univariate distributions such as income variables.
- Package **ineq** computes various inequality measures (Gini, Theil, entropy, among others), concentration measures (Herfindahl, Rosenbluth), and poverty measures (Watts, Sen, SST, and Foster). It also computes and draws empirical and theoretical Lorenz curves as well as Pen's parade. It is not designed to deal with sampling weights directly (these could only be emulated via `rep(x, weights)`).
- Function `priceIndex()` from package **micEcon** allows to estimate the Paasche, the Fisher and the Laspeyres price indices.

Additional Packages and Functionalities:

- Package **rrcovNA** provides robust location and scatter estimation and robust principal component analysis with high breakdown point for incomplete data. It is therefore applicable to find representative and non-representative outliers.
- Package **samplingbook** includes sampling procedures from the book 'Stichproben. Methoden und praktische Umsetzung mit R' by Goeran Kauermann and Helmut Kuechenhoff (2010).
- Package **SDaA** is designed to reproduce results from Lohr, S. (1999) 'Sampling: Design and Analysis, Duxbury' and includes the data sets from this book.

- Package **TeachingSampling** includes functionality for sampling designs and parameter estimation in finite populations.
- Package **urn** allows sampling without replacement.
- Package **memisc** includes tools for the management of survey data, graphics and simulation.
- Package **surveyNG** is an experimental revision of the survey package that provides database interface facilities and efficient calculations with sparse matrices.
- Package **odfWeave.survey** provides support for **odfWeave** for the **survey** package.
- Package **spsurvey** includes facilities for spatial survey design and analysis for equal and unequal probability (stratified) sampling.
- Package **nlme** provides facilities to fit Gaussian linear and nonlinear mixed-effects models and **lme4** provides facilities to fit linear and generalized linear mixed-effects model, both used in small area estimation.

List of Packages: Amelia cat EVER Hmisc impute ineq laeken lme4 memisc mi mice micEcon MImix missMDA mitools mix nlme norm odfWeave.survey pan pps RecordLinkage reweight robCompositions rrcovNA sampfling sampling samplingbook SDaA sdcMicro sdcTable SeqKnn simFrame simPopulation spsurvey StatMatch stratification survey surveyNG TeachingSampling urn VIM x12 yaImpute

Other related task views: TimeSeries, SocialSciences, useR!2008 Tutorial: Small Area Estimation: `http://www.R-project.org/conferences/useR-2008/tutorials/gomez.html`

Index

Glossary

CLI	Command Line Interface, 54
CRAN	Comprehensive R Archive Network, 1
FAO	The Food and Agriculture Organization, 19
GDP	Gross Domestic Product, 49
GPL	GNU General Public License, http://en.wikipedia.org/wiki/GNU_General_Public_License, 1
GUI	Graphical User Interface, 54
IDSB	UNIDO Industrial Demand and Supply Balance Database, http://www.unido.org/index.php?id=1000311, 5
IFS	International Financial Statistics database available from the Statistics Department of the International Monetary Fund, 18
IMF	International Monetary Fund, 18
INDSTAT 4	The UNIDO Industrial Statistics Database comprises data for more than 175 countries and territories spanning the period 1963 to the latest available year. INDSTAT 4 contains the data reported according to ISIC Revision 3 at 3- and 4-digit level., 23
ISIC	United Nations International Standard Industrial Classification of All Economic Activities—the international standard for the classification of productive economic activities, http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=2, 5
LTS	Least Trimmed Squares, 49
MCD	Minimum Covariance Determinant, 31
MSIS	Joint UNECE/Eurostat/OECD Group on Management of Statistical Information Systems, http://www1.unece.org/stat/platform/display/msis/Home+Page, ii
MVA	Manufacturing Value Added, 49

NSO	National Statistical Office, 49
ODBC	Open Database Connectivity, 16
OECD	Organization for Economic Co-operation and Development, 18
OLS	Ordinary Least Squares, 49
SAB	Sharing Advisory Board, `http://www1.unece.org/stat/platform/display/msis/SAB`, ii
SDMX	Statistical Data and Metadata Exchange—an initiative sponsored by many international organizations (BIS, ECB, Eurostat, IMF, OECD, UN and World Bank) for standardization of the exchange of statistical information, `http://www.sdmx.org`, 18
SDMX-ML	XML format for the exchange of SDMX-structured data and metadata, 18
SITC	Standard International Trade Classification, `http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=14`, 6
SOAP	Simple Object Access Protocol—a protocol specification for exchanging structured information in the implementation of Web Services in computer networks, 19
SQL	Structured Query Language, 16
UNECE	United Nations Economic Commission for Europe, `http://www.unece.org/`, ii

Printed in Austria
V.11-81002—March 2011—260

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna International Centre, P.O. Box 300, 1400 Vienna, Austria
Telephone: (+43-1) 26026-0, Fax: (+43-1) 26926-69
E-mail: unido@unido.org, Internet: www.unido.org