Data for the Future

Miguel Alvarez

September 3rd–5th, 2018
Nairobi, Kenya

Contents

Preface

This workshop does not attempt to get experts on the handled topics and tools but to be the first exposure to potential applications for storage, curation and exchange of data. Keeping consistency of data and avoiding typical mistakes will not be only crucial for statistical assessment during a scientific research project but also enhance potential recycling of gained information in the context of meta-assessments. The later issue is not worth to consider for the future of data but also for current activities in the context of interdisciplinary approaches.

Enjoy this quick trip into data and databases!

Chapter 1

Introduction to the Workshop

During the sessions we will work with freeware applications. The main reasons for the use of freeware is saving costs as well as the increasing capabilities developed by freeware and open-source projects.

A disadvantage is that we do not have warranty, and some applications may not be so user-friendly as their analogous commercial software. On the other hand, exchange of data and processes will not be hampered by restricted accessibility to licenses in the context of inter-disciplinary and international research projects.

1.1 Software

The main tool for this course is **R**, which was selected because of its high popularity withing academical community worldwide. For creating tables and handling databases, we will use **Calc** and **Base** from **LibreOffice**, which are freeware alternatives to **Microsoft Office**.

Required software will be distributed in flash disks.

1.2 Data Sets

Four different data sets will be distributed for examples and exercises. One of those examples was adapted from OpenFlights and consists on three tables with information on airports, airlines and routes. A second data set is a typical example for vegetation records including a table of plot information, a table of plant species and a table for records of species on plots. The later example was extracted from Alvarez (2017).

While the previous examples will be used for relational models, there are two additional examples for dealing with time variables, namely a table of weather information recorded by an automatic weather station in Kwasunga (Tanzania) and a table with GPS track points collected in Kenya and including information on location (both decimal degrees and UTM) and time of record.

Further data set examples are installed in **R** and can be accessed by the function `data()`. Also repositories in Internet offer data, such as the case of Zenodo and gesisDataSearch.

1.3 Open Flights

Most of the examples included in this workshop will use the data set from OpenFlights. These data consist on three tables and are distributed in three formats, namely as comma separated values (*airlines.csv*, *airports.csv*, and *routes.csv*), as Open Database (*openflights.odb*) and as R Image (*openflights.rda*).

To load the last alternative in your R session, you should use the following commands:

```
setwd("your_working_directory")
load("data/openflights.rda")
str(airports)
str(airlines)
str(routes)
```

1.4 Vegetation Data

This is a common structure required for vegetation-plot databases. The data was obtained from a published work (Alvarez 2017) and exported from the database **SWEA-Dataveg**. The comma separated values are also three, namely *header.csv*, *species.csv*, and *samples.csv*.

```
load("data/vegkenya.rda")
str(header)
str(species)
str(samples)
```

Chapter 2

Quick Start on R

The following session will be a very short introduction on basics of **R**. Despite there is a lot of applications for statistics on R, this is not an statistical package. Furthermore, it is a programming language for mathematics and plotting and at the same time the interface for executing this language. Since it is a freeware and open source application, it is gaining a lot of popularity among academical community world wide.

For convenience, we are running **R** through the editor **RStudio**, which offers a lot of support while writing scripts. Basic elements of **R** that can be directly explored in the editor are the **console** (the interface), the **workspace** (memory containing objects), and the **working directory** (path used to read or write data).

2.1 Vectors and other Object Classes in R

The fundamental structure of data in R is the `vector`. Vectors will have as attribute a length (dimension) and a class. Class is the type of information contained in vectors, for instance `numeric`, `integer`, `logical`, `character`, etc.

```
A <- c(1, 0.5, -6)
A
```

```
## [1]  1.0  0.5 -6.0
```

```
length(A)
```

```
## [1] 3
```

```
class(A)
```

```
## [1] "numeric"
```

```
B <- as.integer(c(1, 2, 3))
B
```

```
## [1] 1 2 3
```

```
length(B)
```

```
## [1] 3
```

```
class(B)
```

```
## [1] "integer"
```

```
C <- 5 < c(1, 5, 10)
C
```

```
## [1] FALSE FALSE  TRUE
```

```
length(C)
```

```
## [1] 3
```

```
class(C)
```

```
## [1] "logical"
```

```
D <- c("bla", "ble", "bli")
D
```

```
## [1] "bla" "ble" "bli"
```

```
length(D)
```

```
## [1] 3
```

```
class(D)
```

```
## [1] "character"
```

A data frame (class `data.frame`) is a table, whose columns are vectors that may belong to different classes but have to be of the same length. For instance check the data set `iris`.

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

This table has 5 variables represented by columns and 150 observations represented by the number of rows of the table (the length of the single vectors). While most of them are numeric vectors (dimensions of flower elements in cm), the variable `Species` is the classification of the individuals in one out of three species of *Iris* formatted as factor.

2.2 Operators and Functions

R implements all kind of mathematical functions (including operators), as well as statistical and logical functions.

Table 2.1: Mathematical and logical operators.

Operators	Description
`+`, `-`	Addition and subtraction.
`*`, `/`	Multiplication and division.
`^` (`**`)	Power.
`==`, `!=`	Equal, unequal.
`>`, `>=`	Greater than, greater than or equal to.
`<`, `<=`	Less than, less than or equal to.
`&`, `|`	And, or.
`!`, `%in%`	Not, in.
`TRUE`, `FALSE` (`T`, `F`)	True, false.

Additional functions are suitable for descriptive statistics on **R**. More advanced functions will not be handled here.

Table 2.2: Some functions for descriptive statistics of numerical vectors.

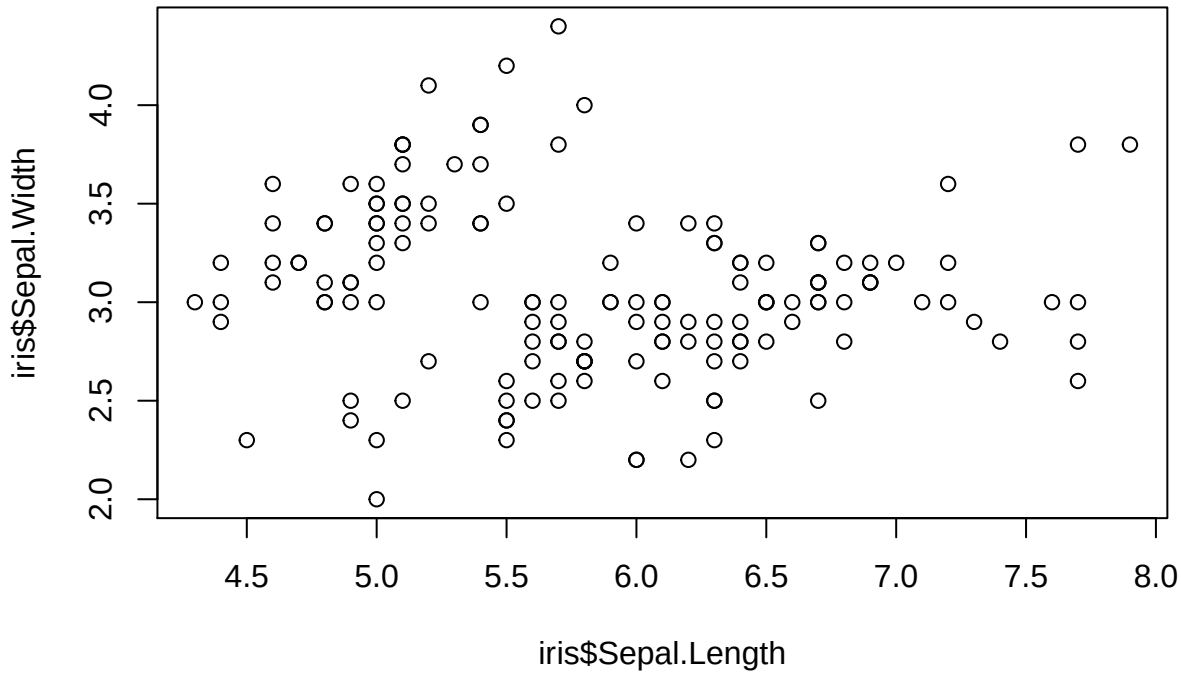
Functions	Description
`lenght()`, `sum()`	Length of vector and sum.
`mean()`, `median()`	Mean and median values.
`var()`, `sd()`	Variance and standard deviation.
`range()`, `min()`, `max()`	Extreme values.
`summary()`	Most of the descriptors at once.

Functions are also objects and have the form `foo(arg1=value1, ..., argn=valuen)`. Values can be provided by mentioning the name of argument or just in the position of argument in the definition of the function.

2.3 Plotting in R

While more details will be discussed during the workshop, just a quick view on a scatterplot.

```
data(iris)
plot(iris$Sepal.Length, iris$Sepal.Width)
```



Further functions will be summarized in the following table. Note that some functions generate full plots (*high level plotting functions*), while other only add elements to existing plot devices (*low level plotting functions*).

Table 2.3: Some basic graphical functions.

Functions	Description
`plot()`, `par()`	Plot and graphical parameters.
`barplot()`, `hist()`	Bar plot and histogram.
`boxplot()`	Box plot.
`pie()`	Pie plot.
`curve()`	Function plot.
`contour()`	Contour plot.
`persp()`	Perspective pot.
`points()`	Add points.
`text()`	Add text.
`abline()`	Add straight line.
`segments()`	Add line segments.
`lines()`	Add connected line segments.
`arrows()`	Add arrows.
`polygon()`	Add polygons.
`legend()`	Add legend.

2.4 Read Data

Several functions are implemented for the inport of diverse formal files in R (see R Data Import). Additional extensions (packages) may also implement functions for loading specific file formats. We will use the function `read.csv()` for importing tables as comma separated values. Comma separated values are text files (the extension *.csv* is just a consensus) using either commas or semilcolons to indicate the separation of columns. The symbols used for separation of cells and decimal positions will depend on the settings of your operative system and the arguments in `read.csv()` should be accordingly adapted (you can also switch to `read.csv2()` or `read.table()`).

Before reading, you have to set your *working directory* for the session. This directory will be used by R for both, reading and writting files.

```
setwd("path_to_your_directory")
```

To separate folders from sub-folders you have to use the slash symbol (`/`) or twice the backslash. Then we will read the file *airports.csv*, wich is contained in the subfolder *data*. Do not forget to assign the output to a new object.

```
airports <- read.csv("data/airports.csv")
head(airports)
```

For more details on basics and statistical applications of **R**, see Crawley (2007), Burns (2011), and Borcard, Gillet, and Legendre (2011).

Exercises

- Make box plots for the variable `Height` in the data `trees`. Check `data(iris)` and draw box plots for the variable `Sepal.Length` separated by `Species`.
- Prepare a plot for publication.

Chapter 3

Working with Data

The most important task of this workshop is to introduce you to alternatives for store digital information in a more advanced way than just using spreadsheets.

3.1 Database List versus Cross Table

Most of the tables included in publications will summarize information either by aggregating variables, crossing them or both. Since by reading scientific publications we will frequently exposed to data formated as cross tables, we will automatically attempt to save data in such formats.

Despite some specific applications, for instance in vegetation ecology, most of the statistical packages will require data formatted in column-oriented tables. The main advantages of column-oriented tables (also called database lists) are:

- Summaries and cross tables can be easily produced by almost any application
- Columns have to be restricted on the class (type) of information included there, keeping consistency in the data.

A common example in **R** for column-oriented tables is the data set `iris` from Anderson (1935):

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Aggregated tables are summaries displaying statistics for groups of data. For instance we can produce an aggregated table for the data set `iris` displaying the mean value of the petal length for every species. Average values will be then calculated from 50 observations in each species:

```
aggregate(Petal.Length ~ Species, data=iris, mean)
```

```
##      Species Petal.Length
## 1     setosa        1.462
## 2 versicolor        4.260
## 3  virginica        5.552
```

Cross tables are also known as *pivot tables* in some applications and represent counts or statistics of a numerical variable against two or more categorical variables. In the case of two categorical variables, one will be displayed in the rows and the other one in the columns. Cross tabulation can be combined with aggregation in some cases.

The next example uses a table of vegetation plot observations. We will then use the package `reshape` to produce a cross table counting plots recorded per community type and country:

```
library(reshape)
load("data/vegkenya.rda")
head(header)
```

```
##     releve_id country_code record_date plot_size cover_total
## 170       259           TZ  2012-04-25         4         100
## 208       207           TZ  2012-03-23         4          90
## 209       208           TZ  2012-03-23         4          85
## 210       209           TZ  2012-03-23         4          95
## 211       210           TZ  2012-03-23         4          70
## 212       211           TZ  2012-03-23         4          98
##           community_type longitude  latitude
## 170 Heliotropium indicum  38.32523 -5.084125
## 208    Cyperus exaltatus  38.32196 -5.085210
## 209    Cyperus exaltatus  38.32196 -5.085210
## 210    Cyperus exaltatus  38.32201 -5.085550
## 211    Typha domingensis  38.32210 -5.084843
## 212    Typha domingensis  38.32203 -5.084605
```

```
recast(header, community_type ~ country_code, fun.aggregate=length,
       measure.var="cover_total")
```

```
##            community_type KE TZ
## 1   Bolboschoenus glaucus  0  2
## 2        Cynodon dactylon  0  2
## 3       Cyperus exaltatus  0 10
## 4      Cyperus latifolius  6  0
## 5         Cyperus papyrus  0 10
## 6    Eichhornia crassipes  0 10
## 7      Ethulia conyzoides  0  2
## 8    Heliotropium indicum  0 10
## 9      Litogyne gariepina  0  3
## 10         Nymphaea lotus  0 10
## 11        Panicum maximum  0  3
## 12       Phragmites karka  0 10
## 13 Sporobolus agrostoides  0 10
## 14    Sporobolus spicatus  0  1
## 15      Typha domingensis  0 11
```

3.2 Database Approach

While most of the spreadsheet applications will be quite flexible regarding the layout and the content of tables, this flexibility is at the risk of mistakes and inconsistencies and mistakes. This issue is quite crucial when working with big data sets, where mistakes may be overseen.

Database software are more strict on it, limiting to the use of column-oriented tables but at the same time restricting the type of data contained in a column (type of variable in column have to be defined a priory in

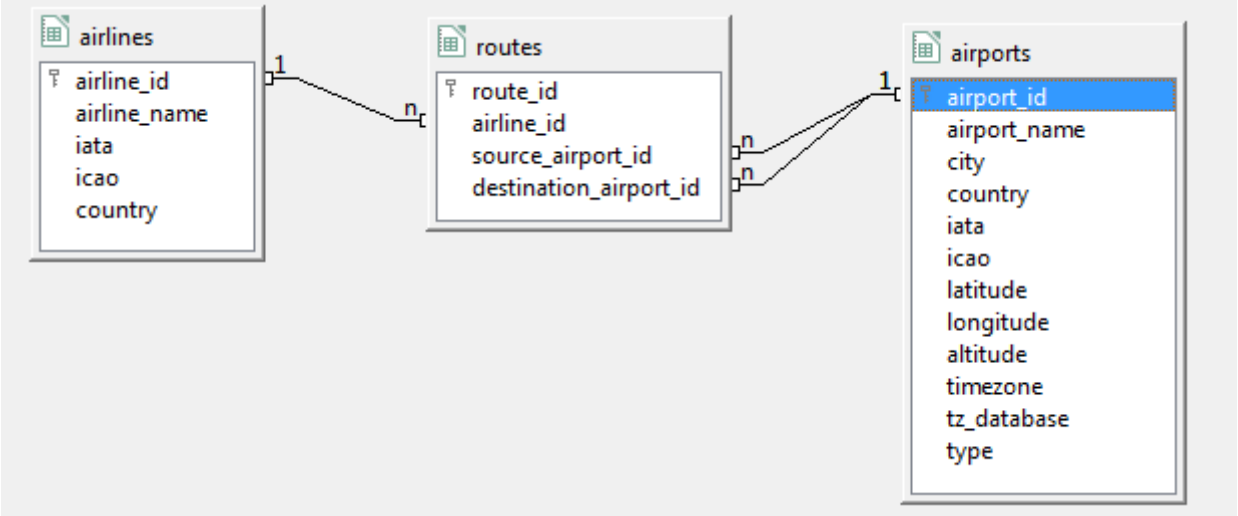


Figure 3.1: Relationships between tables in the OpenFlights data set.

the table). Note that in a database system, every record (row in the table) requires a **key** (an identifier that does not allow null values or duplicates).

Both, LibreOffice and OpenOffice share the same format for databases, which embeds HSQLDB databases. Other freeware alternatives are PostgreSQL and SQLite. Both implements extensions for GIS (PostGIS and SpatiaLite, respectively).

3.3 Relational Databases

When working with data that are related to different elements or different scales in a survey, we may struggle to get all information in just one table. In such cases the most convenient way to store data is by using more than one table and establish relationships between them. For it, we can use relational models, as in the example of flight data.

The links between tables are called **relationships** and set by **foreign keys**.

3.4 Query Building

While the database ensures consistency in the stored data, it implements a way to request information from the database. This is called **query building**. The straight-forward way to do it in **Base** is by using the **Query Design** (graphical query builder).

The graphical query builder will than also generates the respective statements in SQL-language. The Structured Query Language (**SQL**) is used to manipulate data in the database and for accessing it. For instance we can retrieve data and statistics from the example databases.

In **OpenFlights** we can select the airports located in Kenya.

```
SELECT *
FROM "airports"
WHERE "country" = 'Kenya';
```

We can also count the number of airlines starting at every airport:

```
SELECT "source_airport_id", COUNT(DISTINCT "airline_id") AS "airlines_number"
FROM "routes"
GROUP BY "source_airport_id";
```

In the following command, we can access at information stored in all three tables to get the names of airlines and airports instead of just the identifiers:

```
SELECT "airports"."airport_name", "airlines"."airline_name",
  COUNT("routes"."source_airport_id") AS "starts_number"
FROM "routes", "airlines", "airports"
WHERE "routes"."airline_id" = "airlines"."airline_id"
  AND "routes"."source_airport_id" = "airports"."airport_id"
GROUP BY "airports"."airport_name", "airlines"."airline_name";
```

Further we can extend the previous query to get the information for just one airline (**Fly540**):

```
SELECT "airports"."airport_name", "airlines"."airline_name",
  COUNT("routes"."source_airport_id") AS "starts_number"
FROM "routes", "airlines", "airports"
WHERE "routes"."airline_id" = "airlines"."airline_id"
  AND "routes"."source_airport_id" = "airports"."airport_id"
GROUP BY "airports"."airport_name", "airlines"."airline_name"
HAVING (("airlines"."airline_name" = 'Fly540'))
ORDER BY "starts_number" DESC;
```

For more details on SQL commands, visit the tutorials at **w3schools.com** and at **DataCamp**. See also in Beaulieu (2009).

3.5 Connecting R with Base

Depending on the system used for store data in a relational database, a series of packages are currently available for connecting R with your database. For instance, we can connect **HSQLDB** (HyperSQL) using the package `ODB`:

```
library(ODB)
conn <- odb.open("data/openflights.odb")
query <- "SELECT * FROM \"airports\" WHERE \"country\" = 'Kenya';"
airports_ke <- odb.read(conn, query, stringsAsFactors=FALSE)
odb.close(conn)
head(airports_ke)
```

```
##   airport_id                      airport_name    city country iata icao
## 1       1138     Eldoret International Airport Eldoret   Kenya  EDL HKEL
## 2       1140                   Kisumu Airport  Kisumu   Kenya  KIS HKKI
## 3       1141                   Kitale Airport  Kitale   Kenya  KTL HKKT
## 4       1143                   Lodwar Airport  Lodwar   Kenya  LOK HKLO
## 5       1144                   Manda Airstrip    Lamu   Kenya  LAU HKLU
## 6       1145 Mombasa Moi International Airport Mombasa   Kenya  MBA HKMO
##    latitude longitude altitude timezone    tz_database    type
## 1  0.404458   35.2389     6941        3 Africa/Nairobi airport
## 2 -0.086139   34.7289     3734        3 Africa/Nairobi airport
## 3  0.971989   34.9586     6070        3 Africa/Nairobi airport
## 4  3.121970   35.6087     1715        3 Africa/Nairobi airport
## 5 -2.252420   40.9131       20        3 Africa/Nairobi airport
## 6 -4.034830   39.5942      200        3 Africa/Nairobi airport
```

Note that the back-slash is required to escape double quotations inside of double quotations in `character` values.

Exercises

- Import the data set `iris` from **R** into **Base**.
- Create and populate the database **vegkenya** importing the csv files (species, header, and samples) into **Base**.
- Repeat the previous examples for queries in **R**.
- Create a database from own data.

Chapter 4

Time as Variable

The capabilities to manage time are important when documenting the occurrence of events. In time series and dynamic modelling, it is also crucial to use time variables in calculations and comparisons.

4.1 Time in Excel and Base

Microsoft Excel offers different format for representing date and time but the values in time variables are internally stored as numerical values. For instance the value **1** represents the date **01-01-1900**. Decimal positions will be then represent the time of the day.

In **Base** date and time will consider the use of the classes `date`, `time`, `timestamp` (combination of date and time) and `timestamp with time zone`.

While the time may become crucial for documenting events, in some disciplines it may even be one of the variables assessed during data collection.

4.2 Date and Time in R

R has also a class called `Date` for the work with dates and the implementation of time and date in the classes `POSIXct` and `POSIXlt`. Note that date and time (in the case of `POSIXct`) are handled similarly in R as in Microsoft Excel, but using different origins and resolutions.

```
day_zero <- as.Date("1970-01-01")
day_zero
```

```
## [1] "1970-01-01"
```

```
as.numeric(day_zero)
```

```
## [1] 0
```

Note that in the internal numeric value there is an origin ("01-01-1970" in class `Date`), to which the value zero is assigned.

```
as.Date(-2:2, origin=as.Date("2018-09-01"))
```

```
## [1] "2018-08-30" "2018-08-31" "2018-09-01" "2018-09-02" "2018-09-03"
```

For converting different formats of date as character into time, we can use the function `strptime()` (see the help using the command `?strptime`).

```
as.Date(strptime("01 Sep 2018", format="%d %b %Y"))
```

```
## [1] "2018-09-01"
```

```
strptime("01/09/2018 12:30:00", format="%d/%m/%Y %H:%M:%S", tz="Africa/Nairobi")
```

```
## [1] "2018-09-01 12:30:00 EAT"
```

For the names used as time zones, see following table (see also the file **timezone.csv** among distributed data sets). You can access to the current time in your computer by `Sys.Date` and `Sys.time()`.

```
today <- Sys.Date()
today
```

```
## [1] "2018-08-27"
```

```
now <- Sys.time()
now
```

```
## [1] "2018-08-27 23:38:59 CEST"
```

4.3 Calculating and Processing Time

Since `Date` and `POSIXct` classes store time as numeric values, it is possible to make calculations, for instance we can calculate time differences between time zones:

```
now <- "01-09-2018 12:30:00"
DE <- strptime(now, "%d-%m-%Y %T", tz="Europe/Berlin") # Germany
KE <- strptime(now, "%d-%m-%Y %T", tz="Africa/Nairobi") # Kenya
CL <- strptime(now, "%d-%m-%Y %T", tz="America/Santiago") # Chile
DE - KE
```

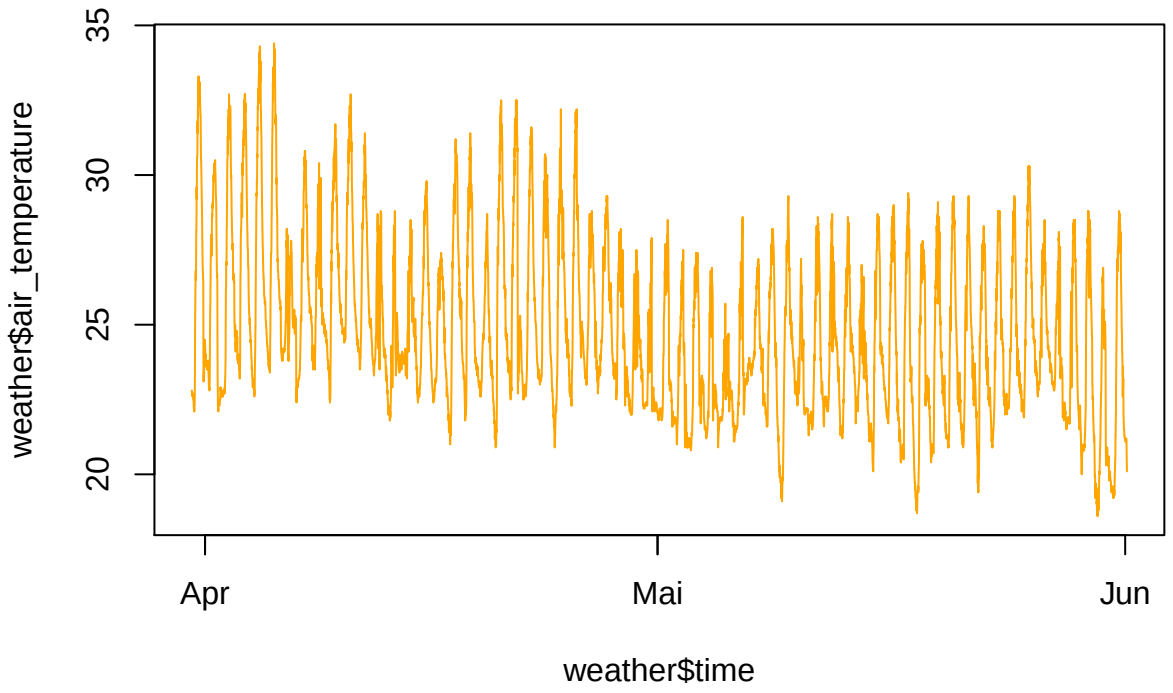
```
## Time difference of 1 hours
```

```
CL-KE
```

```
## Time difference of 6 hours
```

Further hints for the work with time variables can be found in following document. Time variables can be used to display dynamic variables. For instance we can plot temperature recorded every 10 min:

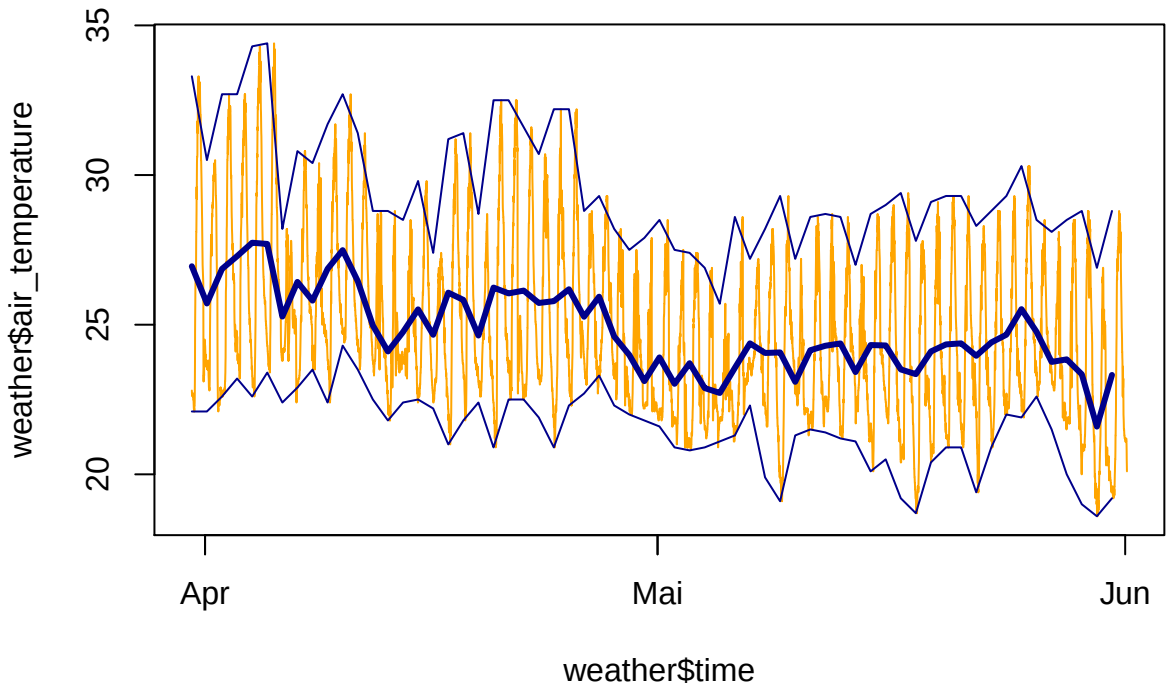
```
load("data/weather_kwasunga.rda")
plot(weather$time, weather$air_temperature, type="l", col="orange")
```



Converting time into date and producing aggregated tables, we can also display some statistics at daily resolution (average and range of values):

```
weather$date <- as.Date(weather$time)
mean_temp <- aggregate(air_temperature ~ date, weather, mean)
max_temp <- aggregate(air_temperature ~ date, weather, max)
min_temp <- aggregate(air_temperature ~ date, weather, min)

plot(weather$time, weather$air_temperature, type="l", col="orange")
lines(as.POSIXct(mean_temp$date), mean_temp$air_temperature, col="darkblue",
        lwd=3)
lines(as.POSIXct(min_temp$date), min_temp$air_temperature, col="darkblue")
lines(as.POSIXct(max_temp$date), max_temp$air_temperature, col="darkblue")
```



It is also possible to extract different components of time variables by using the function `format()`:

```
summary(weather$time)
```

```
##                  Min.               1st Qu.                Median
## "2013-03-31 03:00:00" "2013-04-15 14:57:30" "2013-05-01 02:55:00"
##                  Mean               3rd Qu.                  Max.
## "2013-05-01 02:54:59" "2013-05-16 14:52:30" "2013-06-01 02:50:00"
```

```
doy <- as.numeric(format(weather$time, "%j")) # Day of the year
summary(doy)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    90.0   105.0   121.0   120.6   136.0   152.0
```

```
weeknumber <- as.numeric(format(weather$time, "%U")) # Week of the year
summary(weeknumber)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.00   15.00   17.00   16.95   19.00   21.00
```

```
weekday <- as.numeric(format(weather$time, "%u")) # Day of the week
summary(weekday)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   3.966   6.000   7.000
```

```
month <- as.numeric(format(weather$time, "%m")) # Month number
summary(month)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    4.00    5.00    4.49    5.00    6.00
```

```
year <- as.numeric(format(weather$time, "%Y")) # Month number
summary(year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2013    2013    2013    2013    2013    2013
```

Exercises

- Search after some international flights and calculate the time duration considering time zones at departure and arrival.
- Aggregate all weather variables at daily resolution. Note that rainfall is an accumulated value.
- In the data set **track**, calculate the duration of tracks and the frequency of records.

Chapter 5

The Space

While the time gets crucial to know when data was collected, data collected in the field should also be located in the space. In this workshop we won't focus much on **GIS** (Geogrphical Information Systems), but we will mention some important aspects regarding spatial data sets.

5.1 Spatial Reference Systems

For the location of objects in the space, we will usually refer to points or nodes and provide the values in 2 or 3 dimensions. In general we will also apply for location either geographic reference systems (e.g. latitude and longitude) or projected reference systems (e.g. UTM).

Spatial reference are more than just coordinate systems, they will also apply to a datum (specific model of the Earth's surface, such as geoid or ellipsoids). For the assignment of spatial reference systems we use the function `CRS()` that works as an interface to **Proj.4**, which is a library for conversion among cartographic projections. Another important resource is the database of the **EPSG** (European Petroleum Survey Group) providing identifiers for spatial reference systems.

Among the huge amount of available spatial reference systems, the use of **WGS84** (World Geodetic System 1984) with coordinates in decimal degrees is widespread and the standard in different applications, such as GPX files produced by GPS-devices and KML files used by **Google Earth**. The identifier for **WGS84** in the EPSG database is **4623**.

```
library(sp)
library(rgdal)
load("data/openflights.rda")
coordinates(airports) <- ~ longitude + latitude
proj4string(airports) <- CRS("+init=epsg:4623")
summary(airports)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                 min      max
## longitude -179.8770 179.3410
## latitude   -54.8433  78.2461
## Is projected: FALSE
## proj4string :
## [+init=epsg:4623 +proj=longlat +ellps=intl
## +towgs84=-186,230,110,0,0,0,0 +no_defs]
## Number of points: 3245
```

```
## Data attributes:
##    airport_id    airport_name           city             country
##  Min.   :    2   Length:3245        Length:3245        Length:3245
##  1st Qu.: 1821   Class :character   Class :character   Class :character
##  Median : 3362   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 3719
##  3rd Qu.: 5832
##  Max.   :11922
##
##      iata               icao              altitude
##  Length:3245        Length:3245        Min.   :  -72.0
##  Class :character   Class :character   1st Qu.:   42.0
##  Mode  :character   Mode  :character   Median :  237.0
##                                        Mean   :  978.6
##                                        3rd Qu.: 1036.0
##                                        Max.   :14472.0
##
##     timezone        tz_database           type
##  Min.   :-12.0000   Length:3245        Length:3245
##  1st Qu.: -5.0000   Class :character   Class :character
##  Median :  1.0000   Mode  :character   Mode  :character
##  Mean   :  0.5764
##  3rd Qu.:  5.5625
##  Max.   : 13.0000
##  NA's   :1
```

In the case you like to work at a small scale and apply metric calculations (distances, surface size, etc.), then you may need to project coordinates, for instance to **UTM** (Universal Transverse Mercator). Note that the coordinate values are meters after projection.

```
airports_ke <- subset(airports, country == "Kenya")
summary(coordinates(airports_ke))
```

```
##    longitude        latitude
##  Min.   :34.35   Min.   :-4.2933
##  1st Qu.:35.18   1st Qu.:-2.3506
##  Median :36.98   Median :-0.7027
##  Mean   :37.23   Mean   :-0.6053
##  3rd Qu.:39.58   3rd Qu.: 0.6409
##  Max.   :40.91   Max.   : 4.2041
```

```
airports_ke <- spTransform(airports_ke, CRS("+proj=utm +zone=37 +north +ellps=WGS84"))
summary(coordinates(airports_ke))
```

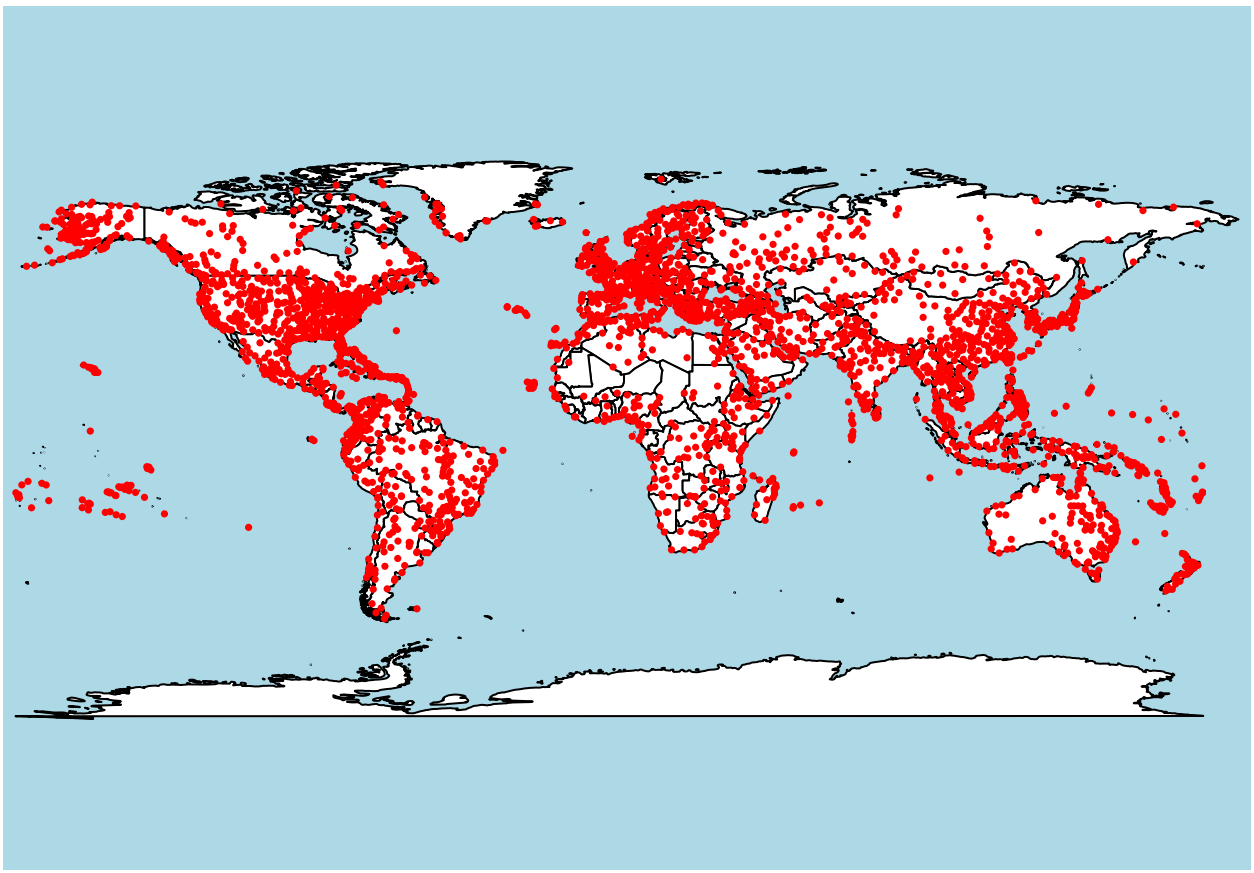
```
##    longitude         latitude
##  Min.   :-16811   Min.   :-474575
##  1st Qu.: 74785   1st Qu.:-259950
##  Median :275698   Median : -77730
##  Mean   :303376   Mean   : -66803
##  3rd Qu.:564019   3rd Qu.:  70924
##  Max.   :712757   Max.   : 466231
```

5.2 Geometries

Features can be represented in the space basically by three different ways, depending on their geometries. Points are commonly used to represent centroids of cities, lines for roads or streams, and polygons for political units or water bodies, among others.

These features will be handled as **shapefiles** in several GIS applications. In **R**, the package `sp` implements the classes `Spatial*DataFrame` (Pebesma and Bivand 2005).

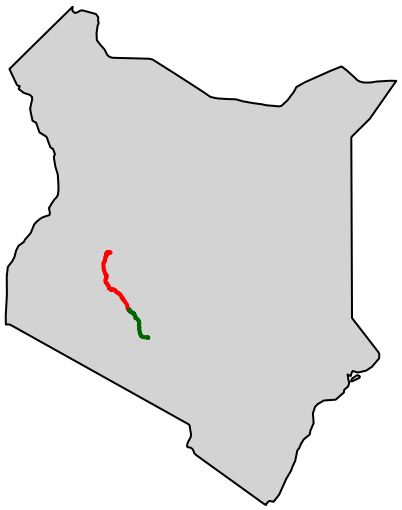
```
library(maps)
load("data/openflights.rda")
map("world", fill=TRUE, col="white", bg="lightblue", ylim=c(-90, 90), mar=c(0,0,0,0))
points(airports$longitude, airports$latitude, col="red", pch=16, cex=0.5)
```



Just to complete the excursion into vectorial data, we plot the track points as lines over the map of Kenya (a polygon).

```
load("data/tracks.rda")
track_1 <- subset(tracks, track_fid == 1)
track_2 <- subset(tracks, track_fid == 2)
map_regions <- map("world", namesonly=TRUE, plot=FALSE)

map("world", region=map_regions[grepl("Kenya", map_regions)], ylim=c(-5, 6),
        fill=TRUE, col="lightgrey")
lines(track_1$lon_deg, track_1$lat_deg, col="red", lwd=2)
lines(track_2$lon_deg, track_2$lat_deg, col="darkgreen", lwd=2)
```



5.3 Imagery

While the previous examples were restricted to vectorial data, information on surfaces can be rather represented in raster data sets. Despite raster data sets are susceptible to re-scaling and re-projecting, they are broadly applied for imagery and spatial modelling. Information usually provided in raster format are satellite imagery, aerial pictures and digital elevation models. One of the most commonly used packages for working with rasters in **R** is the package `raster`.

Further references recommended for the work with GIS data are Hengl (2009), Bivand, Pebesma, and Gómez-Rubio (2008) and Obe and Hsu (2015). See also in Analysis of Spatial Data and the R Spatial sites.

Exercises

- Calculate velocity values using the track points.
- Calculate distances between airports and ground-velocities according to flight schedules.
- Draw a landscape (altitude) profile for one of the provided tracks.

References

Alvarez, Miguel. 2017. "Classification of Aquatic and Semi-Aquatic Vegetation in Two East African Sites: Cocktail Definitions and Syntaxonomy." *Phytocoenologia* 47 (4): 345–64. https://doi.org/10.1127/phyto/2017/0078.

Anderson, E. 1935. "The Irises of the Gaspe Peninsula." *Bulletin of the American Iris Society* 59: 2–5.

Beaulieu, Alan. 2009. *Learning Sql*. O'Reilly Media, Inc, USA. http://www.r-5.org/files/books/computers/languages/sql/mysql/Alan_Beaulieu-Learning_SQL-EN.pdf.

Bivand, Roger S, Edzer J Pebesma, and Virgilio Gómez-Rubio. 2008. *Applied Spatial Data Analysis with R*. New York: Springer. https://doi.org/10.1007/978-0-387-78171-6.

Borcard, Daniel, Francois Gillet, and Pierre Legendre. 2011. *Numerical Ecology with R*. New York: Springer. https://doi.org/10.1007/978-1-4419-7976-6.

Burns, Patrick. 2011. *The R Inferno*. https://www.burns-stat.com/pages/Tutor/R_inferno.pdf.

Crawley, Michael J. 2007. *The R Book*. Chichester: Wiley. ftp://ftp.tuebingen.mpg.de/pub/kyb/bresciani/Crawley%20-%20The%20R%20Book.pdf.

Hengl, Tomislav. 2009. *A Practical Guide to Geostatistical Mapping of Environmental Variables*. Amsterdam. https://www.amazon.com/Practical-Guide-Geostatistical-Mapping/dp/9090249818?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=9090249818.

Obe, Regina O, and Leo Hsu. 2015. *PostGIS in Action*. Stamford: Manning. http://www.ebook.de/de/product/21689796/regina_o_obe_leo_hsu_postgis_in_action.html.

Pebesma, Edzer J, and Roger S Bivand. 2005. "Classes and Methods for Spatial Data in R." *R News* 5 (2): 9–13.

Notes: