Assignment definition

- Structure of air quality (and many types of environmental) data
- Assignment description
- Lesson outline
- Data analysis
    - Examples of data models
    - Data tables
    - Example
    - Split-apply-combine (SAC) strategy

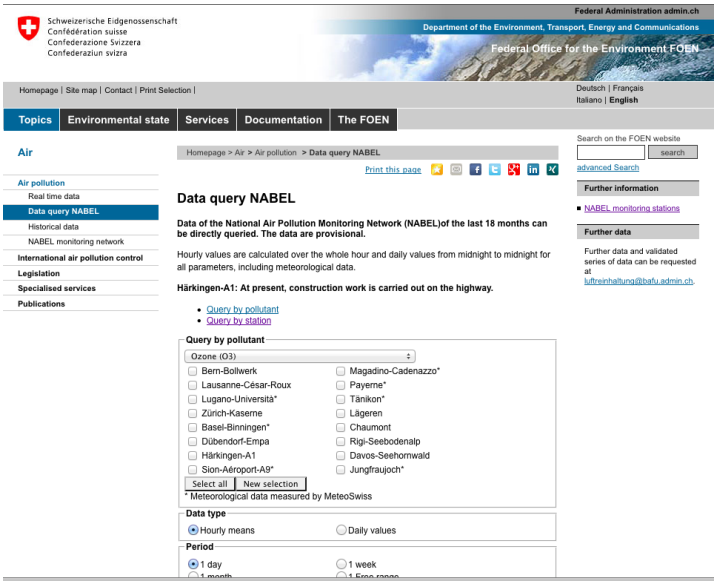
Structure of air quality (and many types of environmental) data

Air pollution data (observations or model predictions) consist of realizations of multiple variables in time and space. How can we structure this data so that we may efficiently apply simple statistical analyses on them? In general, the goal of statistical analysis can be thought of as having two products:

- statistical description (providing a few metrics which represent a large quantity of values)
- statistical inference (which includes hypothesis testing to infer properties of a population, and predictions resulting from inferred characteristics of variables or relationships among them)

In this project, we will learn methods of exploratory data analysis using relational models of data. We will cover the software package R.

Assignment description



The assignment is to analyze measurements from the Swiss Federal Office of the Environment (BAFU/OFFEV/FOEN) network. The objective is to observe seasonal and diurnal patterns in concentrations of each pollutant, and understand the interplay of emissions, photochemistry, and meteorology that gives rise to these patterns to the extent that they can be inferred. You can find many details on the BAFU website (http://www.bafu.admin.ch/luft/index.html?lang=en), and in the lectures of the course.

See course handout for exact assignment instructions.

Lesson outline

1. Motivating example
2. Assignment definition [we are here]
3. R basics
4. Visualizing time series
5. Correlations and cross-correlations
6. Autocorrelation and periodicity
7. Stochastic processes and random variables
8. Inferential statistics and hypothesis testing
9. Extreme values: detection and accommodation

Data analysis

Data analysis describes the process of reducing a set of data to a few relevant conclusions.

Code describes data (e.g., measurements) and its transformations (e.g., averaging).

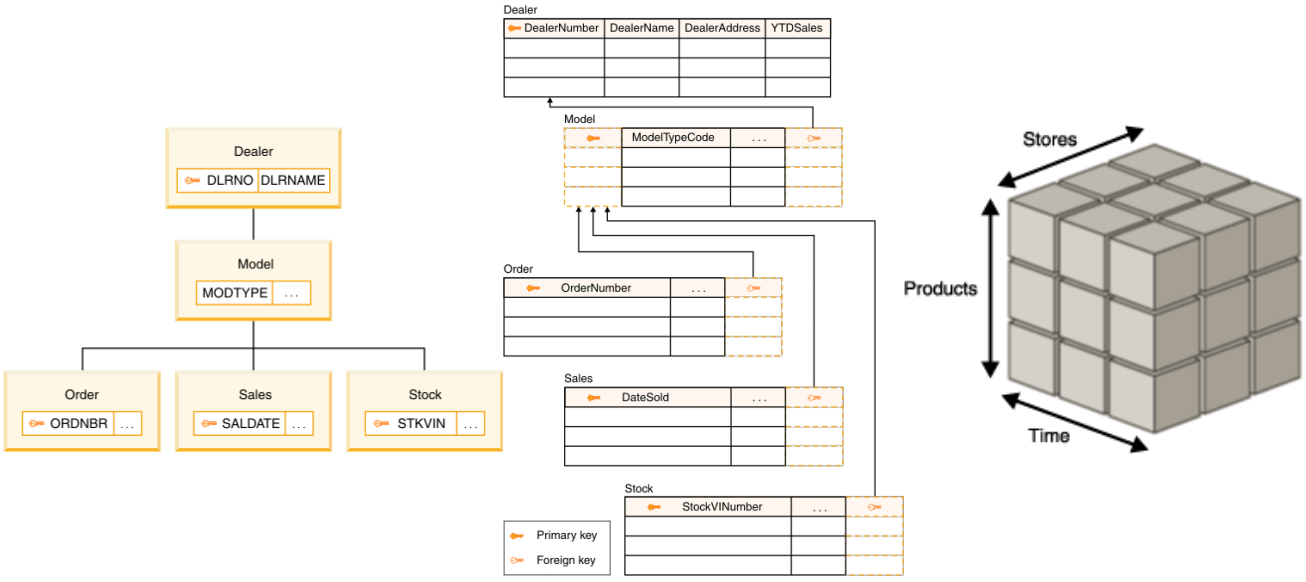
- Data are stored as variables (nouns)
- Transformations are described through functions (verbs)

Structuring your data appropriately can significantly reduce the number of steps required for many types of transformations.

Examples of data models

Data models describe entities and their relationships to other entities.

- Graph (Hierarchical, Network)
- Relational
- “Multidimensional” cube model



These models can be implemented by databases in data warehouses, or data structures residing in local memory.

Array representation in R:

```
dimnames(arr)
```

```
## $year
## [1] "2012" "2013" "2014"
##
## $month
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
##
## $day
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
## [29] "29" "30" "31"
##
## $hour
##  [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13"
## [15] "14" "15" "16" "17" "18" "19" "20" "21" "22" "23"
##
## $variable
## [1] "O3"   "NO2"  "CO"   "PM10" "TEMP" "PREC" "RAD"
```

```
arr["2013","Jul","20",,"O3"]
```

```
##     0     1     2     3     4     5     6     7     8     9    10    11
##  78.4  71.8  69.5  76.8  63.3  50.3  45.2  54.9  65.2  56.6  61.5  74.1
##    12    13    14    15    16    17    18    19    20    21    22    23
##  82.4  91.9  98.8 110.4 118.3 122.0 122.9 113.2  99.9  85.8  54.9  48.9
```

Some operations can be efficiently performed with arrays:

```
apply(arr["2013",,,,],MARGIN=c("month","variable"), FUN=mean, na.rm=TRUE)
```

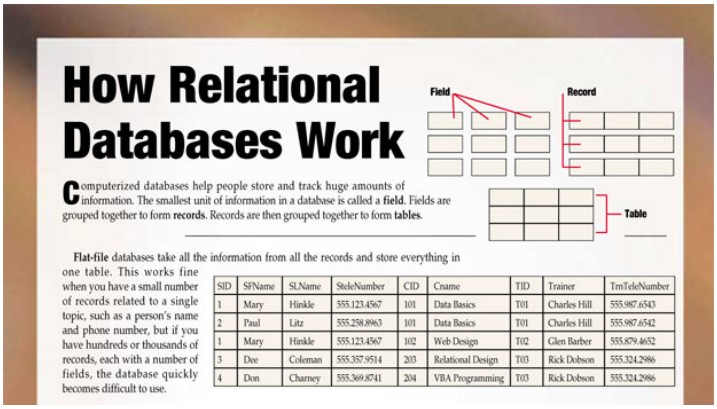
```
##      variable
## month       O3      NO2        CO     PM10       TEMP       PREC       RAD
##   Jan 22.64865 47.52375 0.4440108 22.54845  2.2271505 0.08721400  46.31774
##   Feb 40.58125 45.05580 0.4380952 28.74501  0.6016369 0.07113095  85.00015
##   Mar 35.02078 50.33841 0.4849328 35.95249  4.7498656 0.10713324  98.91492
##   Apr 50.78187 40.48799 0.3930556 25.55132 10.6093056 0.11930556 170.54847
##   May 51.85189 38.63450 0.3339166 12.29838 11.9751344 0.18561828 174.21317
##   Jun 59.61657 38.92350 0.3386111 17.55225 17.6750000 0.12517385 248.54292
##   Jul 76.19515 38.44542 0.3680108 19.92781 22.3884409 0.16424731 272.65753
##   Aug 66.49771 34.13576 0.3336022 17.26532 20.9435484 0.05397039 241.67258
##   Sep 43.20181 42.02855 0.3945833 17.52681 17.0044444 0.12472222 157.18375
##   Oct 24.74307 40.52557 0.4131720 17.46508 13.7291667 0.28721400  82.58642
##   Nov 25.85577 36.38623 0.3695833 13.65309  6.0034722 0.15724234  53.88542
##   Dec 18.55666 57.09166 0.5268817 26.03401  3.8581989 0.13575269  46.13616
```

Hierarchical representation:

```
tree[["2013"]][["Jul"]][["20"]][c("11","12")]
```

```
## $`11`
## $`11`$O3
## [1] 74.1
##
## $`11`$NO2
## [1] 31.6
##
## $`11`$CO
## [1] 0.4
##
## $`11`$PM10
## [1] 15
##
## $`11`$TEMP
## [1] 20.8
##
## $`11`$PREC
## [1] 0
##
## $`11`$RAD
## [1] 799.3
##
##
## $`12`
## $`12`$O3
## [1] 82.4
##
## $`12`$NO2
## [1] 35.1
##
## $`12`$CO
## [1] 0.4
##
## $`12`$PM10
## [1] 17.4
##
## $`12`$TEMP
## [1] 21.7
##
## $`12`$PREC
## [1] 0
##
## $`12`$RAD
## [1] 880.3
```

Data tables



How Relational Databases Work

Field Record Table

Computerized databases help people store and track huge amounts of information. The smallest unit of information in a database is called a **field**. Fields are grouped together to form **records**. Records are then grouped together to form **tables**.

Flat-file databases take all the information from all the records and store everything in one table. This works fine when you have a small number of records related to a single topic, such as a person's name and phone number, but if you have hundreds or thousands of records, each with a number of fields, the database quickly becomes difficult to use.

SID	SFName	SLName	SteleNumber	CID	Cname	TID	Trainer	TmTeleNumber
1	Mary	Hinkle	555.123.4567	101	Data Basics	T01	Charles Hill	555.987.6543
2	Paul	Litz	555.258.8963	101	Data Basics	T01	Charles Hill	555.987.6542
1	Mary	Hinkle	555.123.4567	102	Web Design	T02	Glen Barber	555.879.4652
3	Dee	Coleman	555.357.9514	203	Relational Design	T03	Rick Dobson	555.324.2986
4	Don	Charney	555.369.8741	204	VBA Programming	T03	Rick Dobson	555.324.2986

Relations with 2-D structure (“a multiset of tuples”).

- Observations along rows (identified by unique “keys”)
- Variables along columns

Like matrices, but different:

- Can be a collection of heterogeneous data types along columns: float, integer, string, Boolean, etc.
- The set of operations defined for tables contain ones which are different from matrices. For instance:
- matrices: scalar, matrix, Hadamard, and Kronecker products; transpose, determinant, decomposition (e.g., SVD), ``slice'' (equivalent of subset/select for tables)
- tables: join, group, subset (observations), select (variables)

Example

Import example data:

View in “wide” format:

```
kable(data[1:2,])
```

datetime	O3	NO2	CO	PM10	TEMP	PREC	RAD
31.12.2012 01:00	7.8	56.3	0.5	16.1	3.8	0	-2.4
31.12.2012 02:00	22.4	38.0	0.4	11.6	4.1	0	-2.3

Convert to “long” format:

```
library(reshape2) # makes melt() library available
kable(melt(data[1:2,], id.vars="datetime"))
```

datetime	variable	value
31.12.2012 01:00	O3	7.8
31.12.2012 02:00	O3	22.4
31.12.2012 01:00	NO2	56.3
31.12.2012 02:00	NO2	38.0

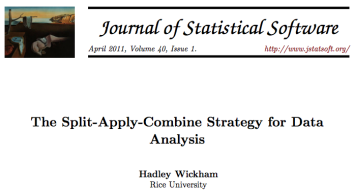
datetime	variable	value
31.12.2012 01:00	CO	0.5
31.12.2012 02:00	CO	0.4
31.12.2012 01:00	PM10	16.1
31.12.2012 02:00	PM10	11.6
31.12.2012 01:00	TEMP	3.8
31.12.2012 02:00	TEMP	4.1
31.12.2012 01:00	PREC	0.0
31.12.2012 02:00	PREC	0.0
31.12.2012 01:00	RAD	-2.4
31.12.2012 02:00	RAD	-2.3

The transformation from the wide format to long format as shown above (and vice versa) is known as “pivoting”. Other common operations with data tables include:

- extracting columns and rows
- joining (merging) two tables

Split-apply-combine (SAC) strategy

Data tables permit a set of operations common to many types of data analyses.



General strategy:

1. *split* table row-wise by a categorical variable (X)
2. *apply* function (Y) on selected variable
3. *combine* results into another table

Example (as shown in Lesson 1):

1. *split* table row-wise by month
2. *apply* mean on selected variable
3. *combine* summary statistics (i.e., mean) into a table

The basic syntax for SAC and describing a graphic in R can be similar in structure.

To return a data table after grouping rows of another data table by =variable1, variable2=; applying =cor= to each piece, and recombining:

```
library(dplyr)
data_table %>%                          # data table
  group_by(variable1,varaible2) %>%     # grouping variables
    summarize(correlation=cor(x,y))     # (aggregating) function
```

There are *many* variations on this specification, but this is the general idea for creating a graphic:

```
library(ggplot2)
ggplot(data=data_table) +               # data table
  facet_grid(variable1~variable2) +     # grouping variables
    geom_line(mapping=aes(x,y,color=z))    # aesthetics
```

The last example shows the application of a popular programming concept for graph specification. We will show more such examples in the following lessons.