Introduction to data management

with applications in hydrobiology

david.kneis@tu-dresden.de

TU Dresden, Institute of Hydrobiology

Outline

Motivation

Basics about tables

Example data set

How to arrange data properly

Software options for data storage

Working with data in base R

Topics not covered

Outline

Typical sources of data

- Monitoring
  (e.g. water quality recorded over time)
- Snapshot sampling
  (e.g. abundance of river bed organisms)
- Experiments
  (e.g. response of system to treatment; with replication)
- Model outputs
  (e.g. scenario or sensitivity analysis)

Why care about data management?

- the key to efficient data analysis
- avoids inconsistency / loss of information
- ensures re-usability by others (and yourself at a later time)
- a must for serious research (traceability of results)
- enables efficient version control and archiving

Investment in good data management **always** pays out.

What is data management about?

1. Arranging data in **tables** with proper layout
2. Selecting a software for data storage and manipulation
3. Understanding operations on tables
   - merging, filters, aggregation
4. Knowing how to create inputs for specific analysis
   - plotting, statistical tests

What is data management about?

1. Arranging data in **tables** with proper layout
2. Selecting a software for data storage and manipulation
3. Understanding operations on tables
   - merging, filters, aggregation
4. Knowing how to create inputs for specific analysis
   - plotting, statistical tests

These will be the main subjects of this course.

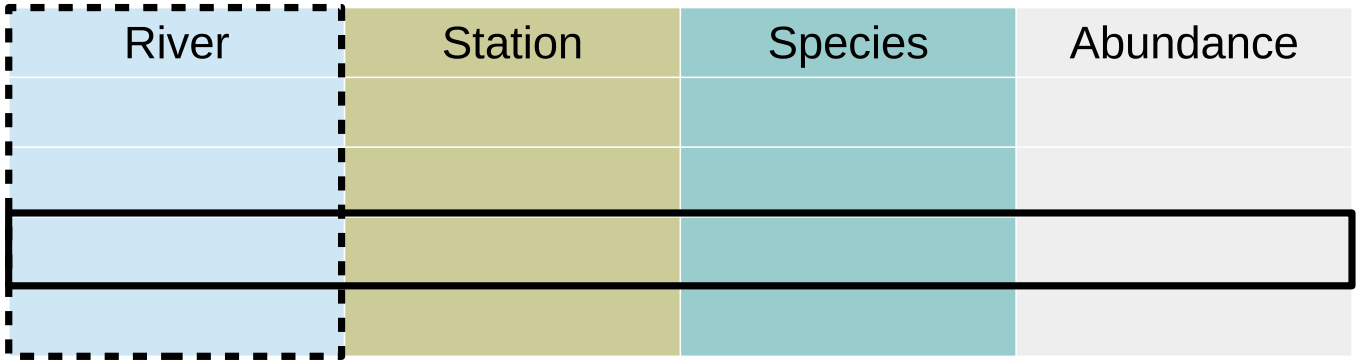
Outline

Data types

numeric	Weights, dimensions, concentrations, ...
integer	Number of offspring, ordinal and nominal data (classes), ID
character	nominal data (classes), ID
logical	All kinds of dichotomous data
special types	dates and times, images, ...

Tables

- Most common and versatile data container.
- Columns are vectors of a particular data type.
- A table row is, in general, not a vectors but a list (because types differ).



Tables

Representation of tables in R

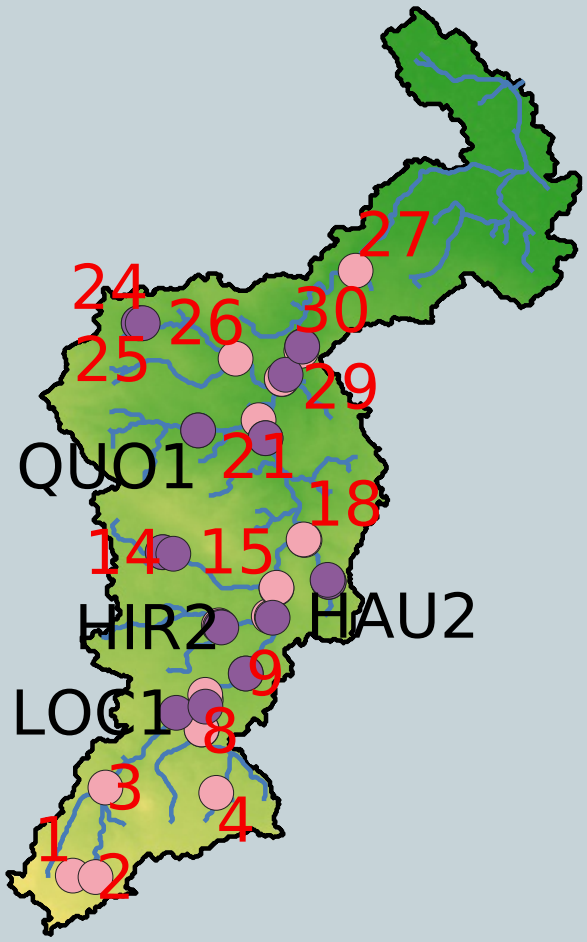
data.frame	Classic, commonly used, but 'ugly' defaults will likely confuse beginners
tibble	Good alternative
data.table	Another alternative

Exercise: A simple data frame

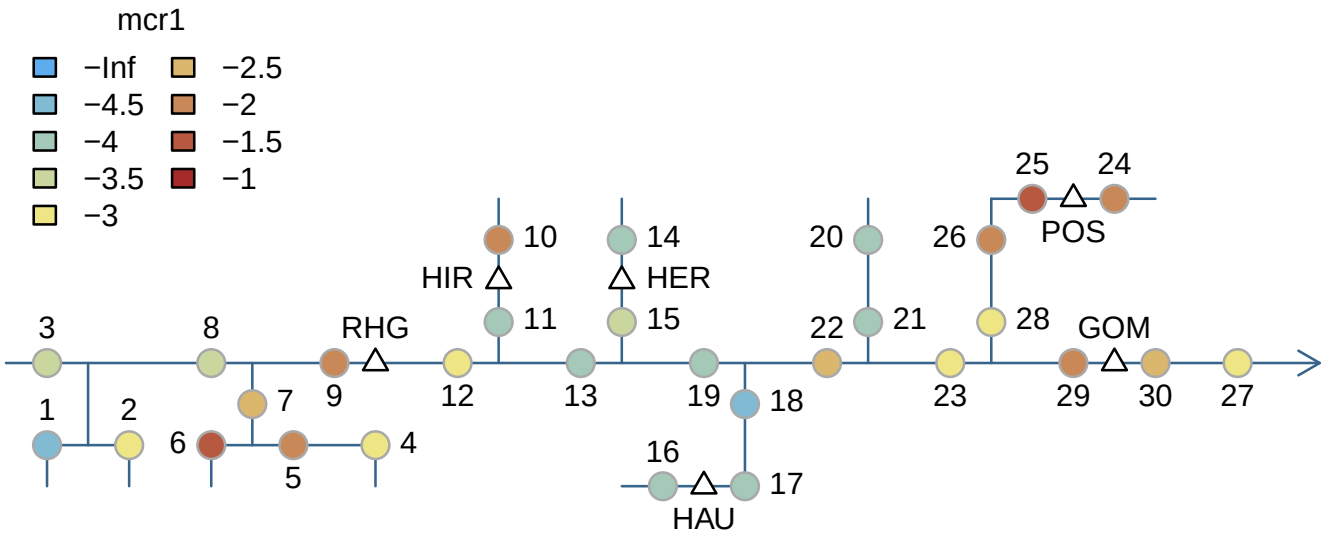

```
rm(list=ls())
options(stringsAsFactors=FALSE)

x <- read.table(file="data/lakedepth.txt",
  sep="\t", header=TRUE)

print(typeof(x))            # type of object
print(str(x))               # structure
print(lapply(x, typeof))    # type of columns
print(head(x))              # top rows
print(x$maxDepth)           # access a column
print(x["maxDepth"])        # ...
print(x[,"maxDepth"])       # ...
print(x[1,])                # access a row
```

Outline

Screening a river for AMR genes



Screening a river for AMR genes



Summary

We sampled ...

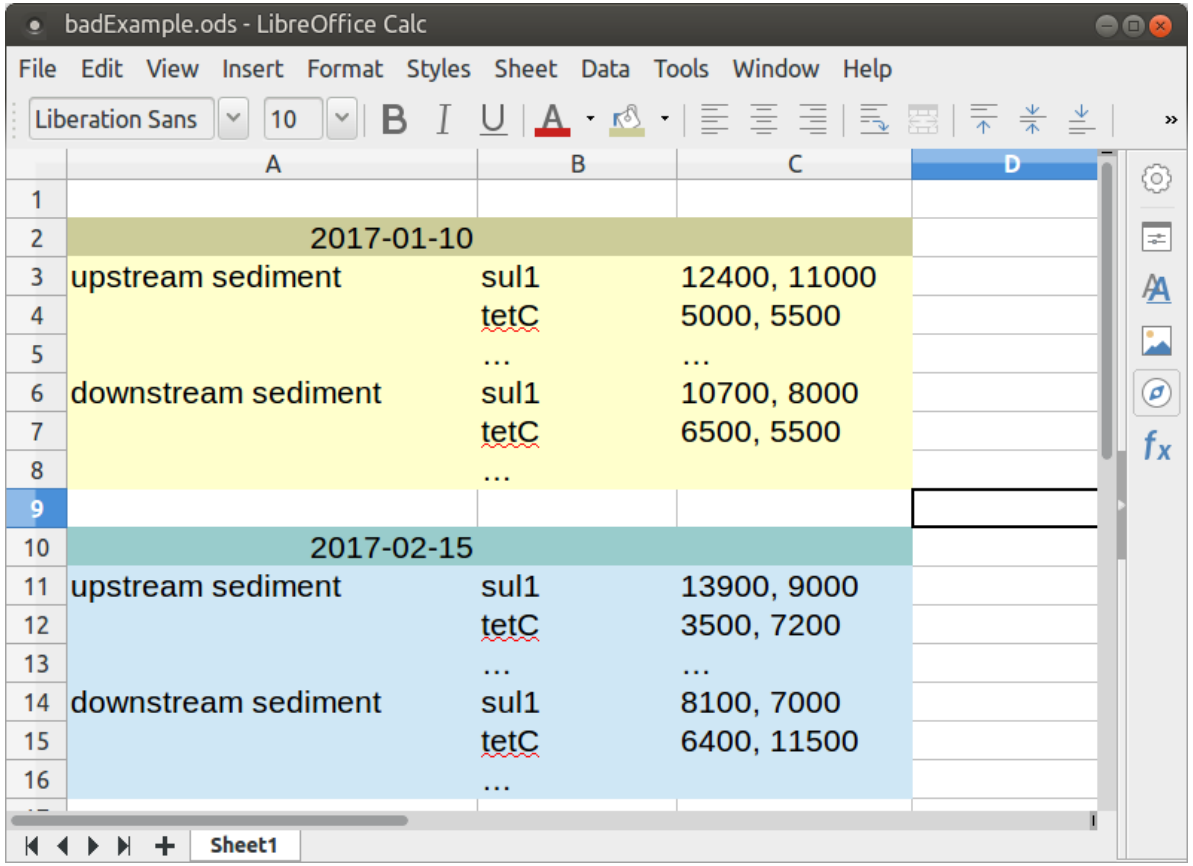
- water and bottom sediment
- at multiple locations
- repeatedly, in monthly intervals

to analyze DNA extracts for ...

- the abundance of various antibiotic resistance genes
- the abundance of marker genes (e.g. 16S rRNA)

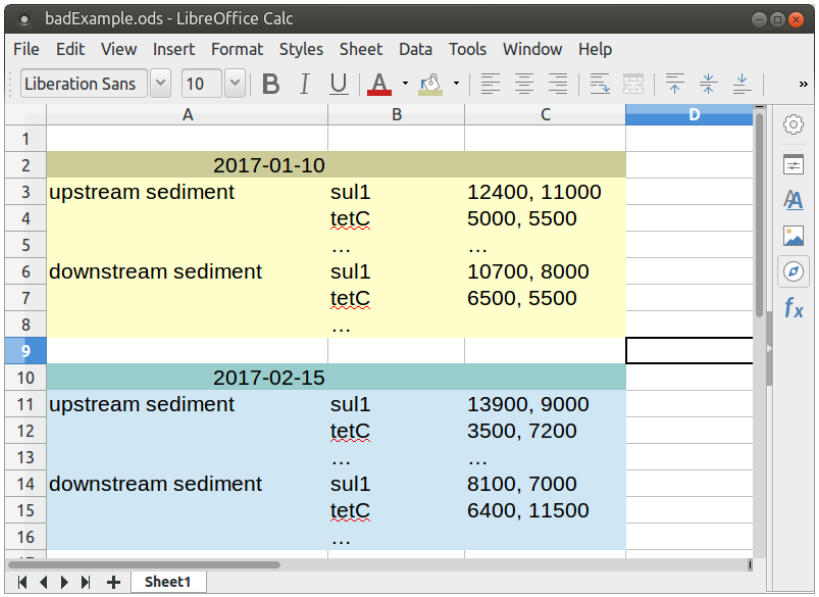
and we took physical and technical replicates.

Why is this bad practice?



	A	B	C
1			
2	2017-01-10		
3	upstream sediment	sul1	12400, 11000
4		tetC	5000, 5500
5		…	…
6	downstream sediment	sul1	10700, 8000
7		tetC	6500, 5500
8		…	
9			
10	2017-02-15		
11	upstream sediment	sul1	13900, 9000
12		tetC	3500, 7200
13		…	…
14	downstream sediment	sul1	8100, 7000
15		tetC	6400, 11500
16		…	

Why is this bad practice?



- Mixed information in column and even cells
- Multiple values per cell
- Many sub-tables on spreadsheet
- Missing headers

- No software can read this out of the box
- Data become useless soon (missing headers and meta data)

Outline

Objectives

Understand ...

- the main structure of a data set.
- how to split the data over separate tables.
- how individual tables are linked to each other.
- basic rules to achieve data integrity.

Data dimensions

Consider the example data set (page 16). What are the major dimensions of the data?

Data dimensions

Consider the example data set (page 16). What are the major dimensions of the data?

- Compartment (water, sediment)
- Space (2-dimensional, sampling locations)
- Time
- Gene

Data dimensions

Consider the example data set (page 16). What are the major dimensions of the data?

- Compartment (water, sediment)
- Space (2-dimensional, sampling locations)
- Time
- ~~Gene~~ Variable

Data dimensions

Consider the example data set (page 16). What are the major dimensions of the data?

- Compartment (water, sediment)
- Space (2-dimensional, sampling locations)
- Time
- Variable

A very common case in hydro-biological field research.

Data dimensions

Consider the example data set (page 16). What are the major dimensions of the data?

- Compartment (water, sediment)
- Space (2-dimensional, sampling locations)
- Time
- Variable

A very common case in hydro-biological field research.

If you are not sure about dimensions, imagine some plots of the data. Which item(s) would appear on the x-axis or in the legend?

Entities

Consider the example data set (page 16). What are the important entities?

Entities

Consider the example data set (page 16). What are the important entities?

- Samples
- Locations
- Compartments
- Variables
- Values (measured numerical properties)

Entities

Consider the example data set (page 16). What are the important entities?

- Samples
- Locations
- ~~Compartments~~ (Dropped for simplicity)
- Variables
- Values (measured numerical properties)

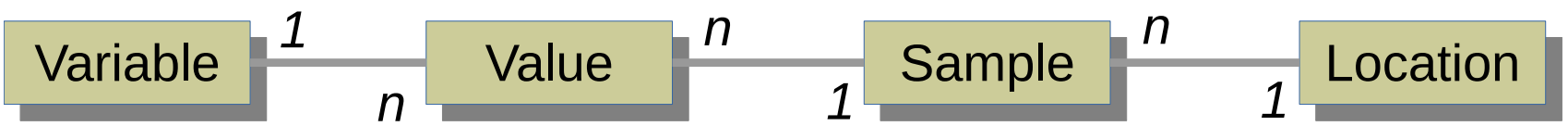
Entities

Consider the example data set (page 16). What are the important entities?

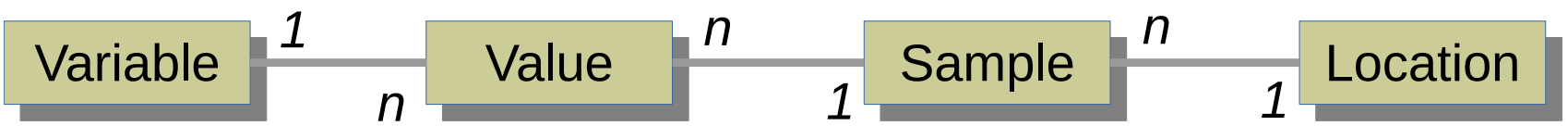
- Samples
- Locations
- ~~Compartments~~ (Dropped for simplicity)
- Variables
- Values (measured numerical properties)

This leads us to the entity-relationship model (ERM)
`https://en.wikipedia.org/wiki/Entity-relationship_model`

Entities and relations

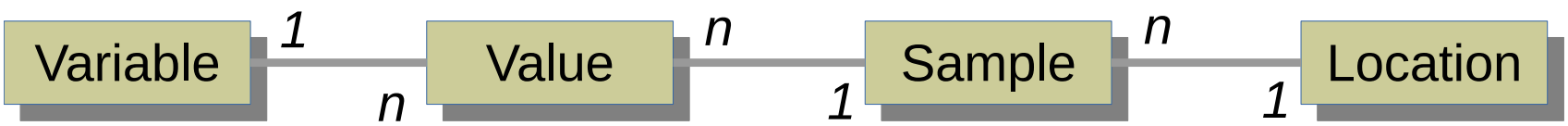


Entities and relations



- Multiple values, each measured on one particular sample
- Multiple samples, each taken at one particular location
- Each value relates to just one variable
- ...

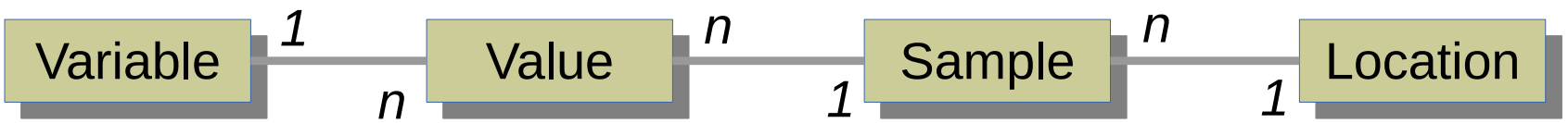
Entities and relations



- Multiple values, each measured on one particular sample
- Multiple samples, each taken at one particular location
- Each value relates to just one variable
- ...

Relations of type 1:1 and n:m also exist and those need to be resolved (not discussed here).

Attributes of entities



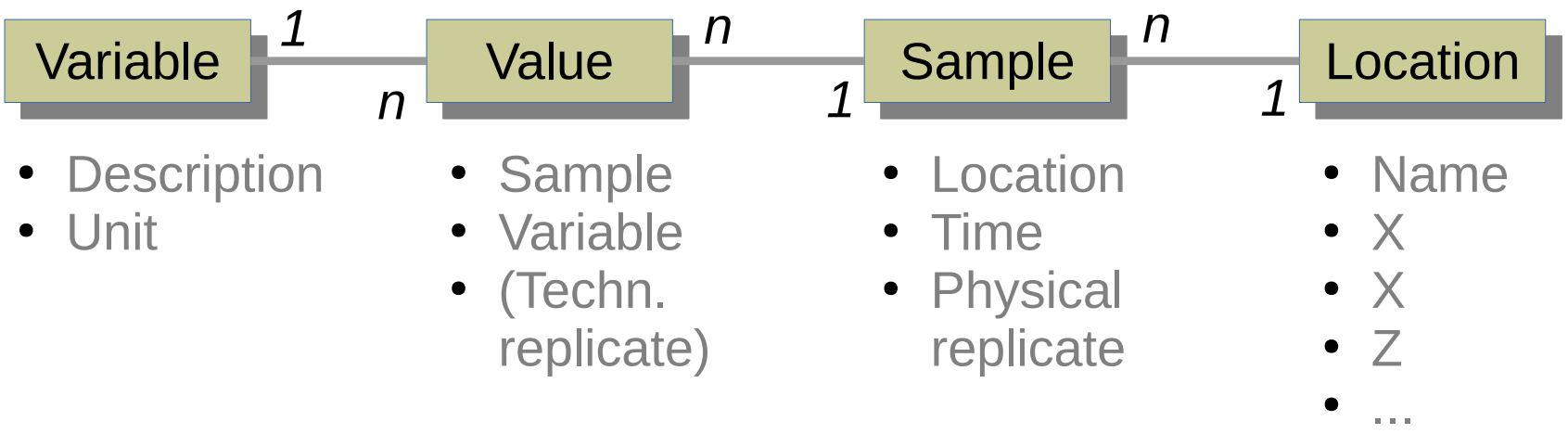
- Description
- Unit

- Sample
- Variable
- (Techn. replicate)

- Location
- Time
- Physical replicate

- Name
- X
- X
- Z
- ...

Attributes of entities



→ Attributes become table columns

Tables and relations

values

sample	*variable*	*repl*	*value*
1	sul1	1	13800
1	tetC	2	5200
1	sul1	1	12300
1	tetC	2	4900
2	sul1	1	14700
2	tetC	2	3300
2	sul1	1	9800
2	tetC	2	11900
...	...	...	...

samples

id	*time*	*location*	*repl*
1	2017-11-09	A	1
2	2017-12-09	A	1
...	...	...	...

locations

id	*name*	*x*	*y*
A	site A	...	...
B	site B	...	...
...	...	...	...

variables

id	*description*	*unit*
sul1	sulfonamide ARG	copies/ng
tetC	tetracyline ARG	copies/ng
...	...	...

Tables and relations

values

sample	*variable*	*repl*	*value*
1	sul1	1	13800
1	tetC	2	5200
1	sul1	1	12300
1	tetC	2	4900
2	sul1	1	14700
2	tetC	2	3300
2	sul1	1	9800
2	tetC	2	11900
...	...	...	...

samples

id	*time*	*location*	*repl*
1	2017-11-09	A	1
2	2017-12-09	A	1
...	...	...	...

locations

id	*name*	*x*	*y*
A	site A	...	...
B	site B	...	...
...	...	...	...

variables

id	*description*	*unit*
sul1	sulfonamide ARG	copies/ng
tetC	tetracyline ARG	copies/ng
...	...	...

- No orphaned records (e.g. only samples from known locations)
- No ambiguity (e.g. two samples cannot share the same ID)

Additional constraints

values

sample	*variable*	*repl*	*value*
1	sul1	1	13800
1	tetC	2	5200
1	sul1	1	12300
1	tetC	2	4900
2	sul1	1	14700
2	tetC	2	3300
2	sul1	1	9800
2	tetC	2	11900
...	...	...	...

samples

id	*time*	*location*	*repl*
1	2017-11-09	A	1
2	2017-12-09	A	1
...	...	...	...

locations

id	*name*	*x*	*y*
A	site A	...	...
B	site B	...	...
...	...	...	...

variables

id	*description*	*unit*
sul1	sulfonamide ARG	copies/ng
tetC	tetracyline ARG	copies/ng
...	...	...

Additional constraints

values

sample	*variable*	*repl*	*value*
1	sul1	1	13800
1	tetC	2	5200
1	sul1	1	12300
1	tetC	2	4900
2	sul1	1	14700
2	tetC	2	3300
2	sul1	1	9800
2	tetC	2	11900
...	...	...	...

samples

id	*time*	*location*	*repl*
1	2017-11-09	A	1
2	2017-12-09	A	1
...	...	...	...

locations

id	*name*	*x*	*y*
A	site A	...	...
B	site B	...	...
...	...	...	...

variables

id	*description*	*unit*
sul1	sulfonamide ARG	copies/ng
tetC	tetracyline ARG	copies/ng
...	...	...

- Each table needs a unique primary key (green color)
- Further columns may require uniqueness (blue color)
- Constraints can apply to a single column or to a set of columns

Summary of basic steps

- Identify entities, attributes, and relations
- Optimize tables following the rules of 'normalization'
- Introduce single-table constraints (primary key, unique, non-emptiness) for data integrity
- Ensure integrity of table relations (foreign key constraints)

→ Look for courses and books on 'relational database design'

Indicators of proper design

- Tables are strictly rectangular
  (well defined number of rows and columns)
- Data is self-contained (all relevant meta data included)
- Tables and columns have intuitive names
- No redundancies (eliminates risk of inconsistency)
- Limited number of explicit missing values (saves memory)
- Tables can be joined properly (no orphaned records)

Why is redundancy bad?

Location	Species	Order	Count
Elbe	Ophiogomphus cecilia	Anodonta	45
Elbe	Calopteryx splendens	Odonata	27
Rhine	Ophiogomphus cecilia	Anodonta	11
Rhine	Calopteryx splendens	Odonata	9
...	...	...	...

redundant **redundant**

Why is redundancy bad?

Location	Species	Order	Count
Elbe	Ophiogomphus cecilia	Anodonta	45
Elbe	Calopteryx splendens	Odonata	27
Rhine	Ophiogomphus cecilia	Anodonta	11
Rhine	Calopteryx splendens	Odonata	9
...	...	...	...

redundant **redundant**

Ugly: Waste of storage space

Why is redundancy bad?

Location	Species	Order	Count
Elbe	Ophiogomphus cecilia	Anodonta	45
Elbe	Calopteryx splendens	Odonata	27
Rhine	Ophiogomphus cecilia	Anodonta	11
Rhine	Calopteryx splendens	Odonata	9
...	...	...	...

redundant **redundant**

Ugly: Waste of storage space

Bad: Need to edit multiple rows to correct the order name of a single species (*Anodonta* is wrong)

Why is redundancy bad?

Location	Species	Order	Count
Elbe	Ophiogomphus cecilia	Anodonta	45
Elbe	Calopteryx splendens	Odonata	27
Rhine	Ophiogomphus cecilia	Anodonta	11
Rhine	Calopteryx splendens	Odonata	9
...	...	...	...

redundant **redundant**

Ugly: Waste of storage space

Bad: Need to edit multiple rows to correct the order name of a single species (*Anodonta* is wrong)

Evil: Danger of creating inconsistency

Why is redundancy bad?

Location	Species	Order	Count
Elbe	Ophiogomphus cecilia	Anodonta	45
Elbe	Calopteryx splendens	Odonata	27
Rhine	Ophiogomphus cecilia	Odonata	11
Rhine	Calopteryx splendens	Odonata	9
...	...	...	...

inconsistent

- Bad and evil often go together ...

Why is redundancy bad?

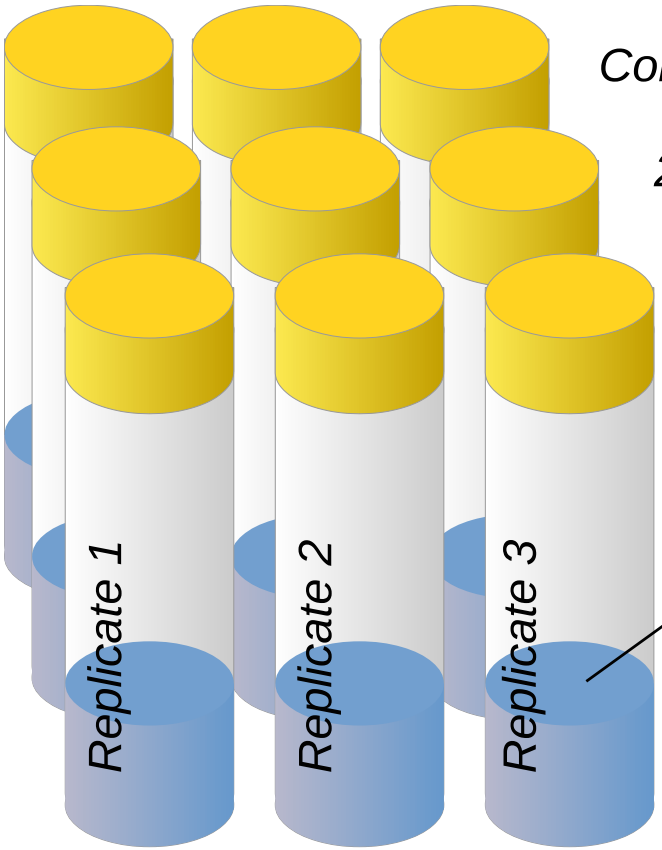
Location	ID_Species	Count
Elbe	1	45
Elbe	2	27
Rhine	1	11
Rhine	2	9
...	...	...

ID	Species	Order
1	Ophiogomphus cecilia	Odonata
2	Calopteryx splendens	Odonata
...	...	...

- Redundancy eliminated
- Inconsistency prevented by database design

Exercise: Microbiological experiments

Propose a database design for this situation



Control (20°C, medium nutrient level)

25°C, medium nutrient level

20°C, low nutrient level

Bacteria
Flagellates
Ciliates

- *Culture densities measured daily over 2 weeks*
- *Replicate microscopic counts*

Outline

Major options

- Relational database client-server software operated with the structured query language (SQL)
- 'Loose' collections of tables
  - Spreadsheet software
  - Delimited text files

SQL data bases

Pro

- Takes care of data integrity (implements tables + constraints)
- Simultaneous editing by multiple users (server + clients)
- Optimized for huge amounts of data (e.g. fast search)

SQL data bases

Pro

- Takes care of data integrity (implements tables + constraints)
- Simultaneous editing by multiple users (server + clients)
- Optimized for huge amounts of data (e.g. fast search)

Cons

- Needs special software
- Need to learn SQL
- Needs maintenance

SQL data bases

Pro

- Takes care of data integrity (implements tables + constraints)
- Simultaneous editing by multiple users (server + clients)
- Optimized for huge amounts of data (e.g. fast search)

Cons

- Needs special software
- Need to learn SQL
- Needs maintenance

→ **Gold standard but overkill for small projects**

Spreadsheet software

Pro

- Convenient data editing and viewing
- Some built-in query features (auto filter, pivot tables)

Spreadsheet software

Pro

- Convenient data editing and viewing
- Some built-in query features (auto filter, pivot tables)

Cons

- Unsafe (no way to implement table constraints)
- Not suitable for version control or archiving
- Size limitations (e.g. number of table rows)
- Software incompatibilities

Spreadsheet software

Pro

- Convenient data editing and viewing
- Some built-in query features (auto filter, pivot tables)

Cons

- Unsafe (no way to implement table constraints)
- Not suitable for version control or archiving
- Size limitations (e.g. number of table rows)
- Software incompatibilities

→ **For small single-user projects**

Delimited text files

Pro

- Maximum portability (spreadsheet, R, GIS, SQL databases, ...)
- Limited multi-user access through version control systems (e.g. git, svn)
- Hard to destroy files

Delimited text files

Pro

- Maximum portability (spreadsheet, R, GIS, SQL databases, ...)
- Limited multi-user access through version control systems (e.g. git, svn)
- Hard to destroy files

Cons

- Unsafe (table constraints to be implemented independently) see, e.g. `https://github.com/dkneis/tabular`

Delimited text files

Pro

- Maximum portability (spreadsheet, R, GIS, SQL databases, ...)
- Limited multi-user access through version control systems (e.g. git, svn)
- Hard to destroy files

Cons

- Unsafe (table constraints to be implemented independently) see, e.g. `https://github.com/dkneis/tabular`

→ **My favorite for typical projects**

Outline

Loading the plain text 'database'


```
rm(list=ls())
options(stringsAsFactors=FALSE)

# convenience function for reading delimited text
rd <- function(f, ...) {
  read.table(file=f, sep="\t", header=TRUE, ...)
}

# load individual tables as data frames
locations <- rd("data/amr_locations.txt")
variables <- rd("data/amr_variables.txt")
samples <- rd("data/amr_samples.txt")
values <- rd("data/amr_values.txt")
```

Checking data integrity


```
# example: test for violation of primary key constraint
if (any(duplicated(samples[,"id"])))
  stop("duplicate sample identifiers")

# example: test for violation of unique constraint
coord <- c("x","y")
if (nrow(unique(locations[,coord])) < nrow(locations))
  stop("multiple locations with identical coordinates")

# example: test for violation of foreign key constraint
if (!all(values[,"sample"] %in% samples[,"id"]))
  stop("values linked to unknown samples")
```

Serious work requires **ALL** constraints to be checked!

Joining tables


```
# join two tables with 'merge'
values <- merge(
  x=values,
  y=samples,
  by.x="sample",  # foreign key field in 'values'
  by.y="id",      # primary key field in 'samples'
  suffixes=c(".tech",".phys")
)

print(head(values))
```

Joining tables

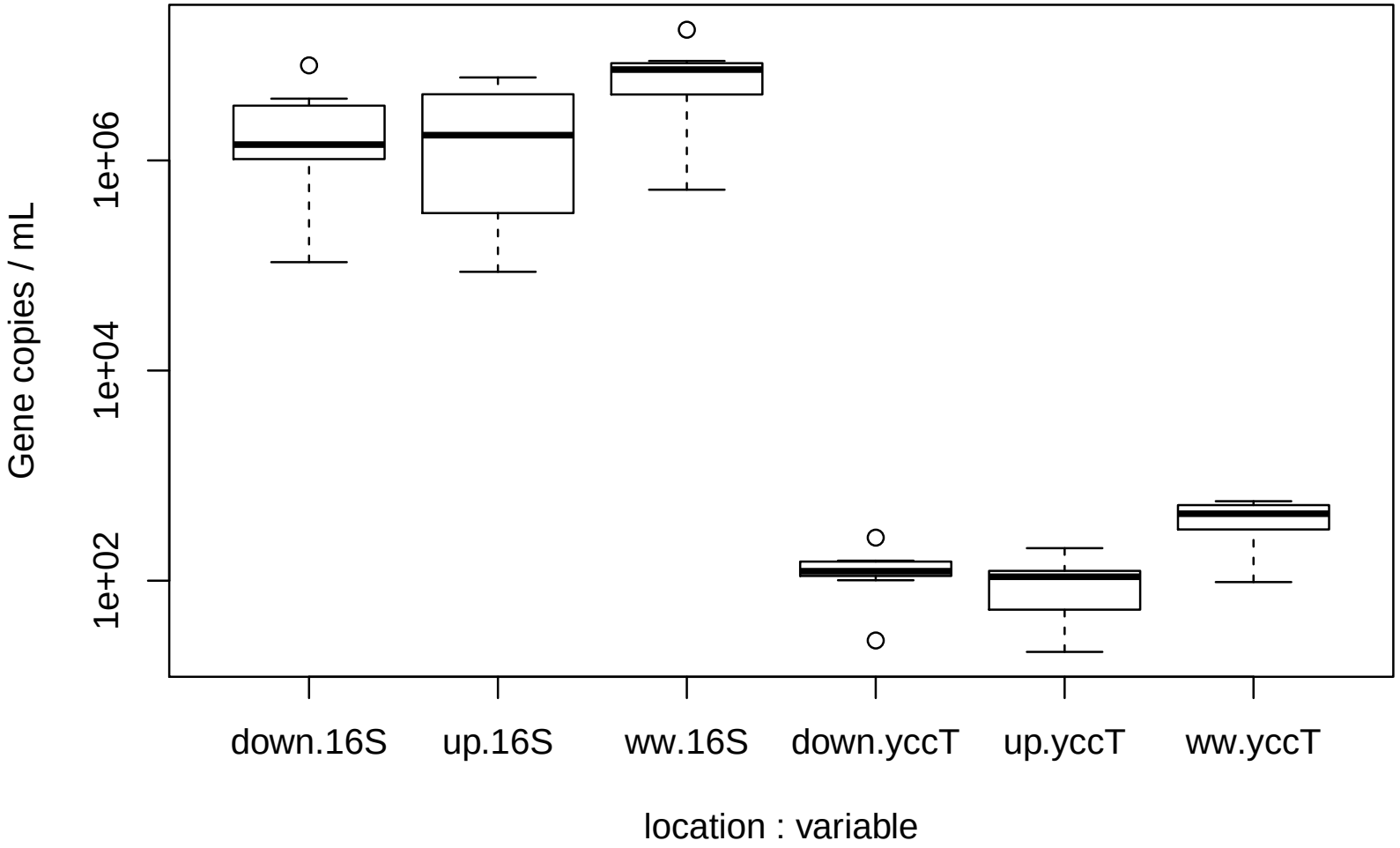

```
# join two tables with 'merge'
values <- merge(
  x=values,
  y=samples,
  by.x="sample",  # foreign key field in 'values'
  by.y="id",      # primary key field in 'samples'
  suffixes=c(".tech",".phys")
)

print(head(values))
```

Check the help page of the 'merge' function

Working with the 'long table' format

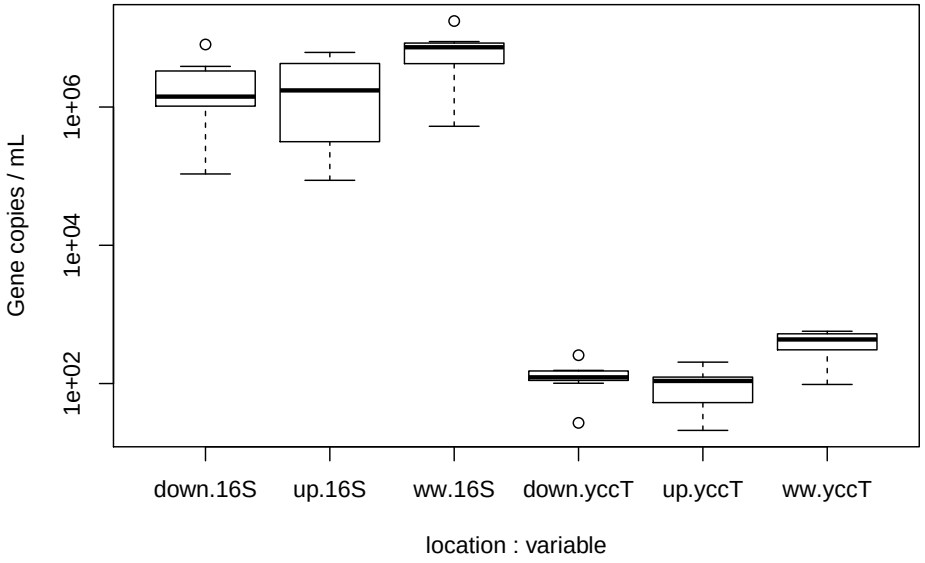

```
# inspect first records
print(head(values))

# example of using a function with formula interface;
# let's compare concentrations of bacterial marker genes
boxplot(
  formula=value ~ location + variable,
  data=values,
  subset=values[,"variable"] %in% c("16S","yccT"),
  log="y",
  ylab="Gene copies / mL"
)
```

Working with the 'long table' format



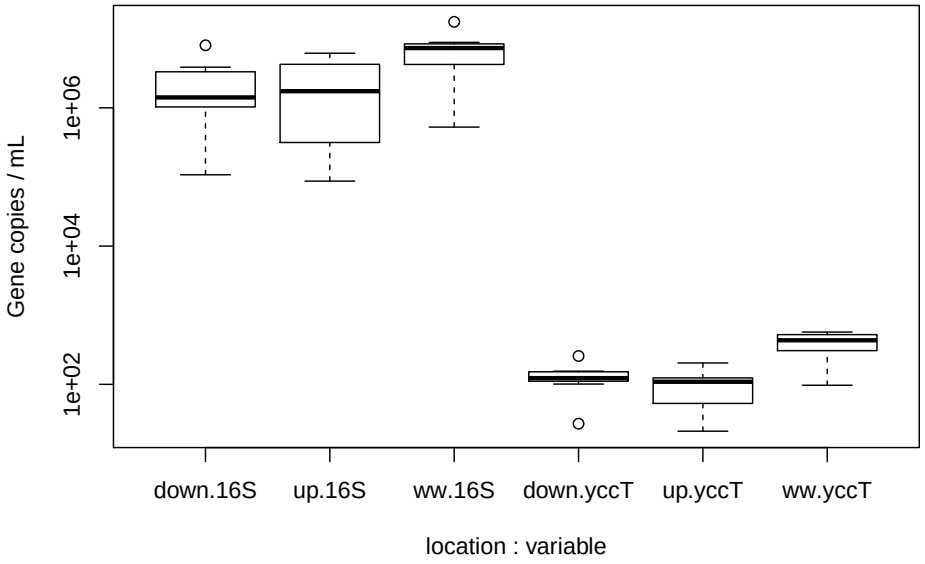
Working with the 'long table' format

What does these data tell us?



Working with the 'long table' format

What does these data tell us?



- Wastewater effluent causes (moderate) microbial pollution.
- Particularly true for *E. coli*, a member of gut communities.
- Statistical significance?

Reshaping: long → wide

Location	Time	Variable	Replicate	Value
A	1	x		
B	1	x		
A	2	x		
B	2	x		
A	1	z		
B	2	z		
...	...	...	...	...

Make data
2-dimensional

(paired observations)

Time	Location A	Location B
1		
2	Data from	column
3	”value”	
4		

Reshaping: long → wide

Aggregate over (e.g. compute mean)

Filter to get rid of this dimension

Goes into the matrix after filtering and aggregation

Location	Time	Variable	Replicate	Value
A	1	x		
B	1	x		
A	2	x		
B	2	x		
A	1	z		
B	2	z		
...	...	...	...	...

Use unique values as row headers

Use unique values as column headers

Time	A	B
1		
2		
3		
4		

'Wide format' makes pairs explicit


```
library(reshape2)              # functions for reshaping

keep <- which(values[,"variable"]=="yccT")  # filter
yccT <- acast(
  formula= time ~ location,  # set row + column headers
  data= values[keep,],       # apply filter
  fun.aggregate= mean,       # aggregate
  value.var= "value"         # define matrix content
)
print(yccT)

# Signed rank test
wilcox.test(x=yccT[,"ww"], y=yccT[,"up"], paired=TRUE)
# Rank sum test (would have worked without reshape)
wilcox.test(x=yccT[,"ww"], y=yccT[,"up"], paired=FALSE)
```

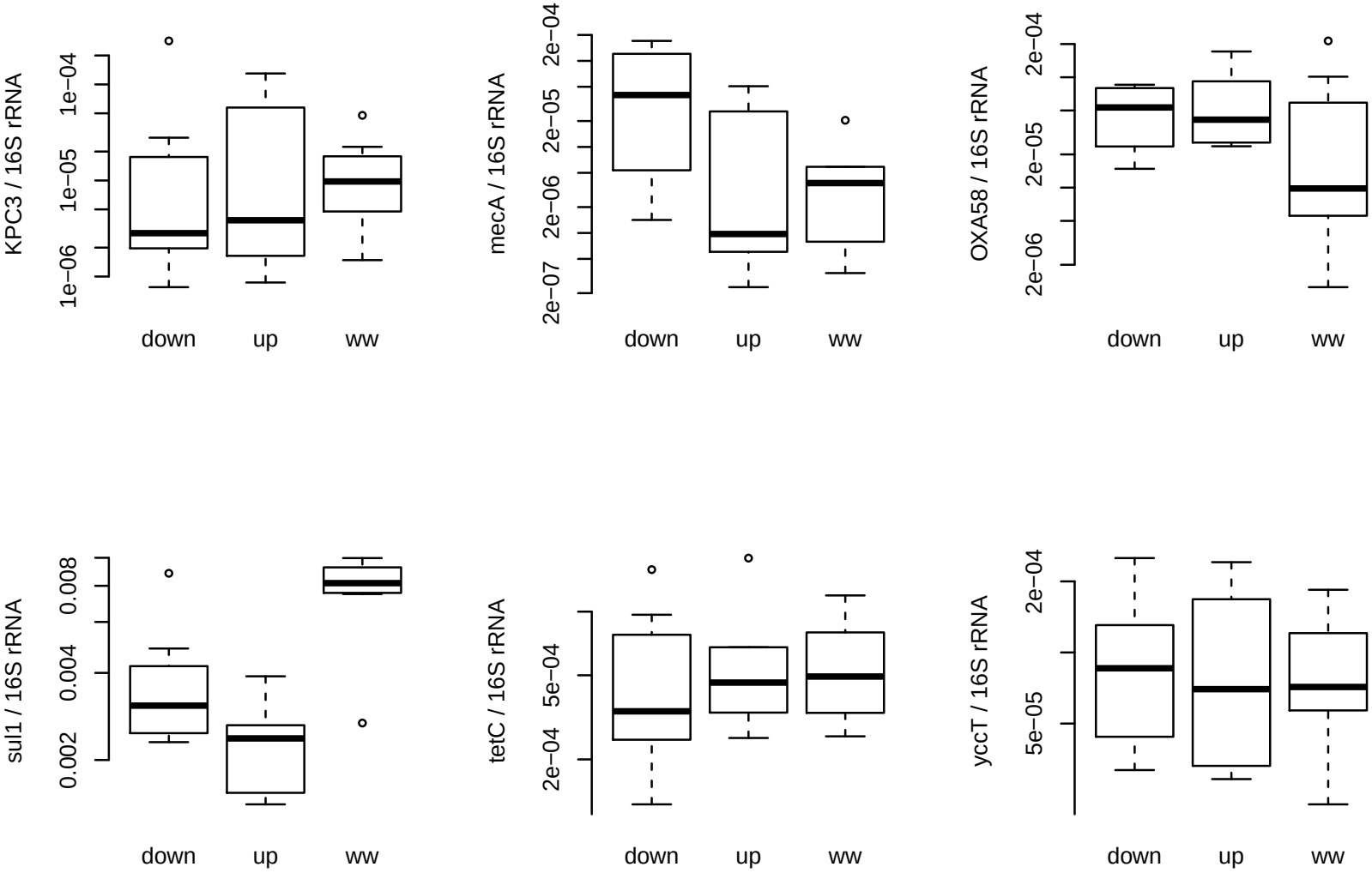
More reshaping: long → wide → long


```
# note 'dcast' instead of 'acast'; two row headers
tmp <- dcast(location + time ~ variable, data=values,
  value.var="value")
print(head(tmp))

# normalize gene abundances using info on 16S rRNA
genes <- c("KPC3","mecA","OXA58","sul1","tetC","yccT")
for (g in genes)
  tmp[,g] <- tmp[,g] / tmp[,"16S"]

# back-transformation to 'long format'
values <- melt(tmp, id.vars=c("location","time"))
values$variable <- as.character(values$variable)
```

Some custom plotting


```
# relative gene abundances in sub-figures
par(mfrow=c(2, ceiling(length(genes)/2)))
for (g in genes) {
  z <- boxplot(
    formula=value ~ location,
    data=values,
    subset=values[,"variable"] == g,
    log="y",
    xlab="", ylab=paste0(g," / 16S rRNA"),
    axes=FALSE
  )
  axis(1, at=1:length(z$names), labels=z$names, lwd=0)
  axis(2)
}
par(mfrow=c(1,1))
```

Some custom plotting



Outline

R can deal with other data sources

Some useful packages

RODBC Direct interaction with SQL databases

readODS Import spreadsheets created with LibreOffice

readxl Import spreadsheets created with ... (guess)

Modern data processing in R

- There is a trend to use high-level packages like `tidyr`, `dplyr`, `ggplot`
- I recommend to learn base R first
- ... and you may often find it sufficient