

Data Frames

Data frames are used to store tabular data

- They are represented as a special type of list where every element of the list has to have the same length
- Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
- Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class
- Data frames also have a special attribute called `row.names`
- Data frames are usually created by calling `read.table()` or `read.csv()`
- Can be converted to a matrix by calling `data.matrix()`

Data Frames

```
> x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
> x
  foo  bar
1   1 TRUE
2   2 TRUE
3   3 FALSE
4   4 FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Factors

Factors are used to represent categorical data. Factors can be unordered or ordered. One can think of a factor as an integer vector where each integer has a *label*.

- Factors are treated specially by modelling functions like `lm()` and `glm()`
- Using factors with labels is *better* than using integers because factors are self-describing; having a variable that has values “Male” and “Female” is better than a variable that has values 1 and 2.

Factors

```
> x <- factor(c("yes", "yes", "no", "yes", "no"))
> x
[1] yes yes no yes no
Levels: no yes
> table(x)
x
no yes
  2  3
> unclass(x)
[1] 2 2 1 2 1
attr(,"levels")
[1] "no" "yes"
```

Factors

The order of the levels can be set using the `levels` argument to `factor()`. This can be important in linear modelling because the first level is used as the baseline level.

```
> x <- factor(c("yes", "yes", "no", "yes", "no"),  
              levels = c("yes", "no"))  
  
> x  
[1] yes yes no yes no  
Levels: yes no
```



Getting Help

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Asking Questions

- Asking questions via email is different from asking questions in person
- People on the other side do not have the background information you have
 - they also don't know you personally (usually)
- Other people are busy; their time is limited
- The instructor (me) is here to help in all circumstances but may not be able to answer all questions!

Finding Answers

- Try to find an answer by searching the archives of the forum you plan to post to.
- Try to find an answer by searching the Web.
- Try to find an answer by reading the manual.
- Try to find an answer by reading a FAQ.
- Try to find an answer by inspection or experimentation.
- Try to find an answer by asking a skilled friend.
- If you're a programmer, try to find an answer by reading the source code.


Asking Questions


- It's important to let other people know that you've done all of the previous things already
- If the answer is in the documentation, the answer will be “Read the documentation”
 - one email round wasted

Example: Error Messages


```
> library(datasets)
> data(airquality)
> cor(airquality)
Error in cor(airquality) : missing observations in cov/cor
```

Google is your friend





About 508,000 results (0.28 seconds)



[\[R\] Error in cor.default\(x1, x2\) : missing observations in cov/cor](#)

<https://stat.ethz.ch/pipermail/r-help/2008.../155523.html> - Block [stat.ethz.ch](#)

Mar 1, 2008 – [R] Error in cor.default(x1, x2) : **missing observations in cov/cor**. Daniel Malter daniel at umd.edu. Thu Feb 28 01:40:58 CET 2008. Previous message: [R] Error in ...

[\[R\] Null values in R](#) - Dec 3, 2008

[\[R\] cor\(data.frame\) infelicities](#) - Dec 3, 2007

[\[BioC\] maanova: missing observations in cov/cor](#) - Aug 25, 2006

[\[R-sig-Geo\] Some comments on calculations with 'asc' \(and 'kasc ...](#) - Mar 27, 2005

[More results from stat.ethz.ch »](#)

[EMBnet 2007 Course - Introduction to Statistics for Biologists](#)

www.ch.embnet.org/CoursEMBnet/Basel07/tp2b.html - Block [ch.embnet.org](#)

you will get an error: Error in cor(thuesen) : **missing observations in cov/cor** . [You will have seen a similar message above with sd .] This is because of the ...

[R help - Error in cor.default\(x1, x2\) : missing observations in cov/cor](#)

r.789695.n4.nabble.com/Error-in-cor-default-x1-x2-missing-... - Block

r.789695.n4.nabble.com

13 posts - 5 authors - Feb 27, 2008

Error in cor.default(x1, x2) : **missing observations in cov/cor**. Hello, I'm trying to do cor(x1,x2) and I get the following error: Error in cor.default(x1, ...

Asking Questions

- What steps will reproduce the problem?
- What is the expected output?
- What do you see instead?
- What version of the product (e.g. R, packages, etc.) are you using?
- What operating system?
- Additional information

Subject Headers

- Stupid: "Help! Can't fit linear model!"
- Smart: "R 3.0.2 lm() function produces seg fault with large data frame, Mac OS X 10.9.1"
- Smarter: "R 3.0.2 lm() function on Mac OS X 10.9.1 -- seg fault on large data frame"

Do

- Describe the goal, not the step
- Be explicit about your question
- Do provide the minimum amount of information necessary (volume is not precision)
- Be courteous (it never hurts)
- Follow up with the solution (if found)

Don't

- Claim that you've found a bug
- Grovel as a substitute for doing your homework
- Post homework questions on mailing lists (we've seen them all)
- Email multiple mailing lists at once
- Ask others to debug your broken code without giving a hint as to what sort of problem they should be searching for

Case Study: A Recent Post to the R-devel Mailing List

Subject: large dataset - confused

Message:

I'm trying to load a dataset into R, but I'm completely lost. This is probably due mostly to the fact that I'm a complete R newb, but it's got me stuck in a research project.

Response

Yes, you are lost. The R posting guide is at <http://www.r-project.org/posting-guide.html> and will point you to the right list and also the manuals (at e.g. <http://cran.r-project.org/manuals.html>, and one of them seems exactly what you need).

Analysis: What Went Wrong?

- Question was sent to the wrong mailing list (R-devel instead of R-help)
- Email subject was very vague
- Question was very vague
- Problem was not reproducible
- No evidence of any effort made to solve the problem
- RESULT: Recipe for disaster!

Places to Turn

- Class discussion board; your fellow students
- r-help@r-project.org
- Other project-specific mailing lists (This talk inspired by Eric Raymond's "How to ask questions the smart way")

Entering Input

At the R prompt we type expressions. The `<-` symbol is the assignment operator.

```
> x <- 1
> print(x)
[1] 1
> x
[1] 1
> msg <- "hello"
```

The grammar of the language determines whether an expression is complete or not.

```
> x <- ## Incomplete expression
```

The `#` character indicates a comment. Anything to the right of the `#` (including the `#` itself) is ignored.

Evaluation

When a complete expression is entered at the prompt, it is evaluated and the result of the evaluated expression is returned. The result may be auto-printed.

```
> x <- 5 ## nothing printed
> x      ## auto-printing occurs
[1] 5
> print(x) ## explicit printing
[1] 5
```

The `[1]` indicates that `x` is a vector and 5 is the first element.

Printing

```
> x <- 1:20  
> x  
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
[16] 16 17 18 19 20
```

The `:` operator is used to create integer sequences.

Matrices

Matrices are vectors with a *dimension* attribute. The dimension attribute is itself an integer vector of length 2 (nrow, ncol)

```
> m <- matrix(nrow = 2, ncol = 3)
> m
      [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
> attributes(m)
$dim
[1] 2 3
```

Matrices (cont'd)

Matrices are constructed *column-wise*, so entries can be thought of starting in the “upper left” corner and running down the columns.

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```


Matrices (cont'd)

Matrices can also be created directly from vectors by adding a dimension attribute.

```
> m <- 1:10
> m
[1] 1 2 3 4 5 6 7 8 9 10
> dim(m) <- c(2, 5)
> m
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

cbind-ing and rbind-ing

Matrices can be created by *column-binding* or *row-binding* with `cbind()` and `rbind()`.

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
      x y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
      [,1] [,2] [,3]
x         1     2     3
y        10    11    12
```

Missing Values

Missing values are denoted by `NA` or `NaN` for undefined mathematical operations.

- `is.na()` is used to test objects if they are `NA`
- `is.nan()` is used to test for `NaN`
- `NA` values have a class also, so there are integer `NA`, character `NA`, etc.
- A `NaN` value is also `NA` but the converse is not true

Missing Values

```
> x <- c(1, 2, NA, 10, 3)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```



Overview and History of R

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

What is R?

What is R?

What is R?

R is a dialect of the S language.

What is S?

- S is a language that was developed by John Chambers and others at Bell Labs.
- S was initiated in 1976 as an internal statistical analysis environment—originally implemented as Fortran libraries.
- Early versions of the language did not contain functions for statistical modeling.
- In 1988 the system was rewritten in C and began to resemble the system that we have today (this was Version 3 of the language). The book *Statistical Models in S* by Chambers and Hastie (the white book) documents the statistical analysis functionality.
- Version 4 of the S language was released in 1998 and is the version we use today. The book *Programming with Data* by John Chambers (the green book) documents this version of the language.

Historical Notes

- In 1993 Bell Labs gave StatSci (now Insightful Corp.) an exclusive license to develop and sell the S language.
- In 2004 Insightful purchased the S language from Lucent for \$2 million and is the current owner.
- In 2006, Alcatel purchased Lucent Technologies and is now called Alcatel-Lucent.
- Insightful sells its implementation of the S language under the product name S-PLUS and has built a number of fancy features (GUIs, mostly) on top of it—hence the “PLUS”.
- In 2008 Insightful is acquired by TIBCO for \$25 million
- The fundamentals of the S language itself has not changed dramatically since 1998.
- In 1998, S won the Association for Computing Machinery’s Software System Award.

S Philosophy

In “Stages in the Evolution of S”, John Chambers writes:

“[W]e wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important.”

<http://www.stat.bell-labs.com/S/history.html>

Back to R

- 1991: Created in New Zealand by Ross Ihaka and Robert Gentleman. Their experience developing R is documented in a 1996 *JCGS* paper.
- 1993: First announcement of R to the public.
- 1995: Martin Mächler convinces Ross and Robert to use the GNU General Public License to make R free software.
- 1996: A public mailing list is created (R-help and R-devel)
- 1997: The R Core Group is formed (containing some people associated with S-PLUS). The core group controls the source code for R.
- 2000: R version 1.0.0 is released.
- 2013: R version 3.0.2 is released on December 2013.

Features of R

- Syntax is very similar to S, making it easy for S-PLUS users to switch over.
- Semantics are superficially similar to S, but in reality are quite different (more on that later).
- Runs on almost any standard computing platform/OS (even on the PlayStation 3)
- Frequent releases (annual + bugfix releases); active development.

Features of R (cont'd)

- Quite lean, as far as software goes; functionality is divided into modular packages
- Graphics capabilities very sophisticated and better than most stat packages.
- Useful for interactive work, but contains a powerful programming language for developing new tools (user -> programmer)
- Very active and vibrant user community; R-help and R-devel mailing lists and Stack Overflow

Features of R (cont'd)

It's free! (Both in the sense of beer and in the sense of speech.)

Free Software

With *free software*, you are granted

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

<http://www.fsf.org>

Drawbacks of R

- Essentially based on 40 year old technology.
- Little built in support for dynamic or 3-D graphics (but things have improved greatly since the “old days”).
- Functionality is based on consumer demand and user contributions. If no one feels like implementing your favorite method, then it's *your* job!
 - (Or you need to pay someone to do it)
- Objects must generally be stored in physical memory; but there have been advancements to deal with this too
- Not ideal for all possible situations (but this is a drawback of all software packages).

Design of the R System

The R system is divided into 2 conceptual parts:

1. The “base” R system that you download from CRAN
2. Everything else.

R functionality is divided into a number of *packages*.

- The “base” R system contains, among other things, the **base** package which is required to run R and contains the most fundamental functions.
- The other packages contained in the “base” system include **utils**, **stats**, **datasets**, **graphics**, **grDevices**, **grid**, **methods**, **tools**, **parallel**, **compiler**, **splines**, **tcltk**, **stats4**.
- There are also “Recommend” packages: **boot**, **class**, **cluster**, **codetools**, **foreign**, **KernSmooth**, **lattice**, **mgcv**, **nlme**, **rpart**, **survival**, **MASS**, **spatial**, **nnet**, **Matrix**.

Design of the R System

And there are many other packages available:

- There are about 4000 packages on CRAN that have been developed by users and programmers around the world.
- There are also many packages associated with the Bioconductor project (<http://bioconductor.org>).
- People often make packages available on their personal websites; there is no reliable way to keep track of how many packages are available in this fashion.

Some R Resources

Available from CRAN (<http://cran.r-project.org>)

- An Introduction to R
- Writing R Extensions
- R Data Import/Export
- R Installation and Administration (mostly for building R from sources)
- R Internals (not for the faint of heart)

Some Useful Books on S/R

Standard texts

- Chambers (2008). *Software for Data Analysis*, Springer. (your textbook)
- Chambers (1998). *Programming with Data*, Springer.
- Venables & Ripley (2002). *Modern Applied Statistics with S*, Springer.
- Venables & Ripley (2000). *S Programming*, Springer.
- Pinheiro & Bates (2000). *Mixed-Effects Models in S and S-PLUS*, Springer.
- Murrell (2005). *R Graphics*, Chapman & Hall/CRC Press.

Other resources

- Springer has a series of books called *Use R!*.
- A longer list of books is at <http://www.r-project.org/doc/bib/R-books.html>

Creating Vectors

The `c()` function can be used to create vectors of objects.

```
> x <- c(0.5, 0.6)      ## numeric
> x <- c(TRUE, FALSE)   ## logical
> x <- c(T, F)          ## logical
> x <- c("a", "b", "c") ## character
> x <- 9:29              ## integer
> x <- c(1+0i, 2+4i)     ## complex
```

Using the `vector()` function

```
> x <- vector("numeric", length = 10)
> x
[1] 0 0 0 0 0 0 0 0 0 0
```

Mixing Objects

What about the following?

```
> y <- c(1.7, "a")    ## character  
> y <- c(TRUE, 2)     ## numeric  
> y <- c("a", TRUE)   ## character
```

When different objects are mixed in a vector, *coercion* occurs so that every element in the vector is of the same class.

Explicit Coercion

Objects can be explicitly coerced from one class to another using the `as.*` functions, if available.

```
> x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Explicit Coercion

Nonsensical coercion results in `NA`s.

```
> x <- c("a", "b", "c")
> as.numeric(x)
[1] NA NA NA
Warning message:
NAs introduced by coercion
> as.logical(x)
[1] NA NA NA
> as.complex(x)
[1] NA NA NA
Warning message:
NAs introduced by coercion
```


Matrices

Matrices are vectors with a *dimension* attribute. The dimension attribute is itself an integer vector of length 2 (nrow, ncol)

```
> m <- matrix(nrow = 2, ncol = 3)
> m
      [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> dim(m)
[1] 2 3
> attributes(m)
$dim
[1] 2 3
```

Matrices (cont'd)

Matrices are constructed *column-wise*, so entries can be thought of starting in the “upper left” corner and running down the columns.

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matrices (cont'd)

Matrices can also be created directly from vectors by adding a dimension attribute.

```
> m <- 1:10
> m
[1] 1 2 3 4 5 6 7 8 9 10
> dim(m) <- c(2, 5)
> m
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

cbind-ing and rbind-ing

Matrices can be created by *column-binding* or *row-binding* with `cbind()` and `rbind()`.

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
      x y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
      [,1] [,2] [,3]
x         1     2     3
y        10    11    12
```

Lists

Lists are a special type of vector that can contain elements of different classes. Lists are a very important data type in R and you should get to know them well.

```
> x <- list(1, "a", TRUE, 1 + 4i)
> x

[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```