Sprint session

NetCDF functionality in the R statistical software

Tom Van Engeland & Karline Soetaert

Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO-KNAW)
P.O.Box 140
4400 AC Yerseke
The Netherlands

t.vanengeland@nioo.knaw.nl

September 13, 2010

The afternoon programme

- Outline
- Installing R
- Introduction to R
- R language basics
- Exercises

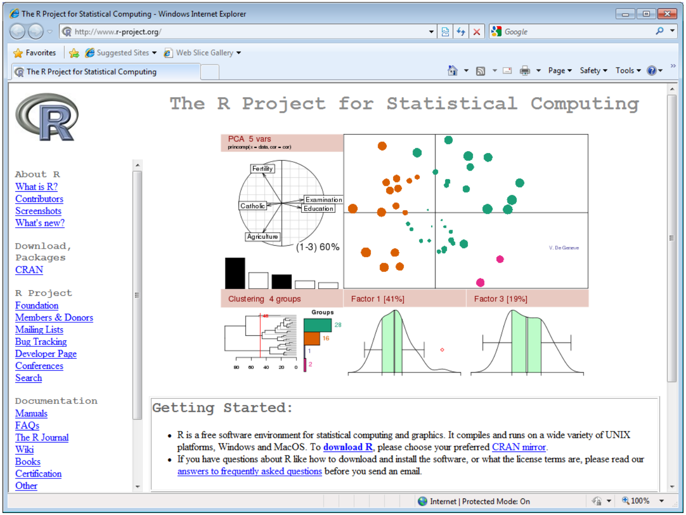
The afternoon programme

- Outline
- Installing R
- Introduction to R
- R language basics
- Exercises

- Introduction to NetCDF
- Using NetCDF format in R
- Working with your own data

Installing R Statistical Software

Downloading R from the R-project website: `http://www.r-project.org`



The R Project for Statistical Computing - Windows Internet Explorer
http://www.r-project.org/
Favorites
Suggested Sites
Web Slice Gallery
The R Project for Statistical Computing
Page
Safety
Tools
Internet | Protected Mode: On
100%

The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

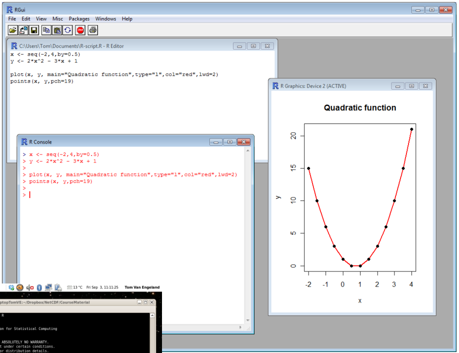
Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

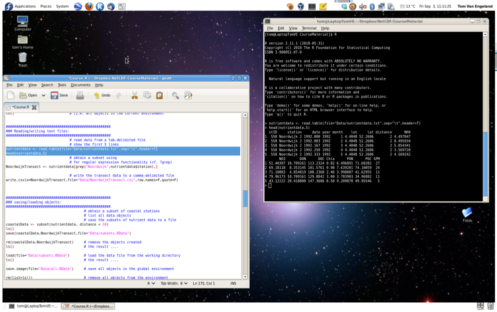
Installing R Statistical Software

- Rgui
  (Graphical User Interface)
  (!Windows only!)
- Rconsole
  (+ any lightweight text editor)
  (!All systems!)


RGui
File Edit View Misc Packages Windows Help
C:\Users\Tom\Documents\R-script.R - R Editor
R Console
R Graphics: Device 2 (ACTIVE)
Quadratic function
y
x
0
5
10
15
20
-2
-1
0
1
2
3
4

```
x <- seq(-2,4,by=0.5)
y <- 2*x^2 - 3*x + 1

plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
points(x, y,pch=19)
```

```
> x <- seq(-2,4,by=0.5)
> y <- 2*x^2 - 3*x + 1
>
> plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
> points(x, y,pch=19)
>
> |
```

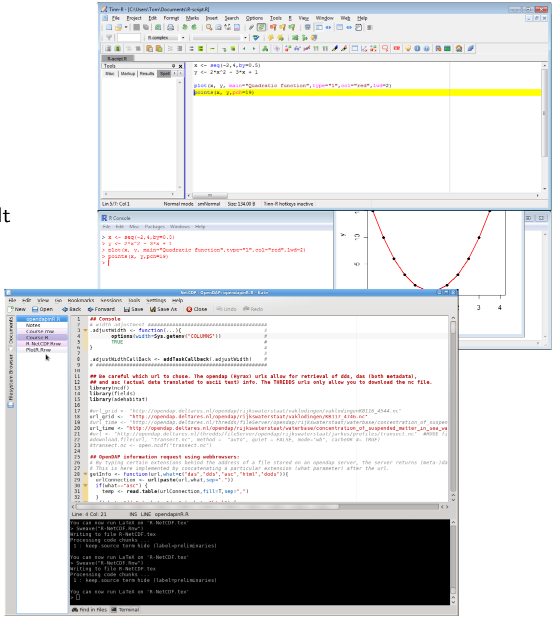

Applications Places System
13 °C Fri Sep 3, 11:11:25 Tom Van Engeland
Computer
tom's Home
Trash
*Course.R (~/Dropbox/NetCDF/CourseMaterial) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
*Course.R
R Tab Width: 8 Ln 175, Col 1 INS
tom@LaptopTomVE:~/Dropbox/NetCDF/CourseMaterial
File Edit View Terminal Help
Fotos
tom@LaptopTomVE:~...
*Course.R (~/Dropbox...

```
ls()                           # (i.e. all objects in the current environment


#####################################################
### Reading/writing text files:
#####################################################
                               # read data from a tab-delimited file
                               # show the first 5 lines
nutrientdata <- read.table(file="Data/nutrientdata.txt",sep="\t",header=T)
head(nutrientdata,5)
                               # obtain a subset using
                               # for regular expression functionality (cf. ?grep)
NoordwijkTransect <- nutrientdata[grep("Noordwijk",nutrientdata$station),]

                               # write the transect data to a comma-delimited file
write.csv(x=NoordwijkTransect,file="Data/NoordwijkTransect.csv",row.names=F,quote=F)



#####################################################
### saving/loading objects:
#####################################################
                                      # obtain a subset of coastal stations
                                      # list all data objects
                                      # save the subsets of nutrient data to a file
coastalData <- subset(nutrientdata, distance < 10)
ls()
save(coastalData,NoordwijkTransect,file="Data/subsets.RData")

rm(coastalData,NoordwijkTransect)     # remove the objects created
ls()                                  # the result ....

load(file="Data/subsets.RData")       # load the data file from the working directory
ls()                                  # the result ....

save.image(file="Data/all.RData")     # save all objects in the global environment

rm(list=ls())                         # remove all objects from the environment
```

```
[tom@LaptopTomVE CourseMaterial]$ R

R version 2.11.1 (2010-05-31)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> nutrientdata <- read.table(file="Data/nutrientdata.txt",sep="\t",header=T)
> head(nutrientdata,5)
  stID     station     date year month    lon     lat distance      NH4
1  558 Noordwijk 2 1992.000 1992     1 4.4048 52.2606        2 4.497847
2  558 Noordwijk 2 1992.083 1992     2 4.4048 52.2606        2 8.495934
3  558 Noordwijk 2 1992.167 1992     3 4.4048 52.2606        2 5.854341
4  558 Noordwijk 2 1992.250 1992     4 4.4048 52.2606        2 3.569720
5  558 Noordwijk 2 1992.333 1992     5 4.4048 52.2606        2 4.569242
       NO3       DON      DOC Chla      PON      POC SPM
1 51.40397 10.709161 113.2324 0.82 6.496891 71.60282  27
2 69.18118  8.353145 101.5761 0.98 7.639201 74.10059  24
3 71.10883  4.854819 108.2368 2.46 3.998087 41.62955  11
4 79.96173 10.709161 129.8842 3.00 3.783903 34.96882  11
5 43.12222 20.418800 147.3686 8.50 9.209878 49.95546   5
> []
```

Installing R Statistical Software

- Dedicated GUIs and IDEs
  - Windows
    - Tinn-R
    - WinEdt + RWinEdt
    - TextPad
  - Linux
    - Kate
    - gEdit
  - Multiplatform
    - Komodo Edit + SciViews-K
    - ESS for Emacs
    - Eclipse
    - jEdit
    - Alpha


Tinn-R - [C:\Users\Tom\Documents\R-script.R]
File Project Edit Format Marks Insert Search Options Tools R View Window Web Help
R-script.R
Tools
Misc Markup Results Spell
Ln 5/7: Col 1 Normal mode smNormal Size: 134.00 B Tinn-R hotkeys inactive
R Console
File Edit Misc Packages Windows Help
NetCDF / OpenDAP: opendapinR.R - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
Documents
Filesystem Browser
opendapinR.R
Notes
Course.rnw
Course.R
R-NetCDF.Rnw
PlotR.Rnw
Line: 4 Col: 21 INS LINE opendapinR.R
Find in Files
Terminal
y
10
5
3
4

```
x <- seq(-2,4,by=0.5)
y <- 2*x^2 - 3*x + 1

plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
points(x, y,pch=19)
```

```
> x <- seq(-2,4,by=0.5)
> y <- 2*x^2 - 3*x + 1
> plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
> points(x, y,pch=19)
> |
```

```
## Console
# width adjustment ##########################################
.adjustWidth <- function(...){                              #
        options(width=Sys.getenv("COLUMNS"))                 #
        TRUE                                                 #
}                                                            #
                                                             #
.adjustWidthCallBack <- addTaskCallback(.adjustWidth)       #
# ##########################################################

## Be careful which url to chose. The opendap (Hyrax) urls allow for retrieval of dds, das (both metadata),
## and asc (actual data translated to ascii text) info. The THREDDS urls only allow you to download the nc file.
library(ncdf)
library(fields)
library(adehabitat)

#url_grid <- "http://opendap.deltares.nl/opendap/rijkswaterstaat/vaklodingen/vaklodingenKB116_4544.nc"
url_grid <-  "http://opendap.deltares.nl/opendap/rijkswaterstaat/vaklodingen/KB117_4746.nc"
#url_time <- "http://opendap.deltares.nl/thredds/fileServer/opendap/rijkswaterstaat/waterbase/concentration_of_suspen
url_time <- "http://opendap.deltares.nl/opendap/rijkswaterstaat/waterbase/concentration_of_suspended_matter_in_sea_wa
#url <- "http://opendap.deltares.nl/thredds/fileServer/opendap/rijkswaterstaat/jarkus/profiles/transect.nc"  #HUGE fi
#download.file(url, "transect.nc", method =  "auto", quiet = FALSE, mode="wb", cacheOK #= TRUE)
#transect.nc <- open.ncdf("transect.nc")

## OpenDAP information request using webbrowsers:
# By typing certain extensions behind the address of a file stored on an opendap server, the server returns (meta-)da
# This is here implemented by concatenating a particular extension (what parameter) after the url.
getInfo <- function(url,what=c("das","dds","asc","html","dods")){
  urlConnection <- url(paste(url,what,sep="."))
  if(what=="asc") {
    temp <- read.table(urlConnection,fill=T,sep=",")
  }
```

```
You can now run LaTeX on 'R-NetCDF.tex'
> Sweave("R-NetCDF.Rnw")
Writing to file R-NetCDF.tex
Processing code chunks ...
 1 : keep.source term hide (label=preliminaries)

You can now run LaTeX on 'R-NetCDF.tex'
> Sweave("R-NetCDF.Rnw")
Writing to file R-NetCDF.tex
Processing code chunks ...
 1 : keep.source term hide (label=preliminaries)

You can now run LaTeX on 'R-NetCDF.tex'
> 
```

GNU's S

- Statistics and graphics
- Open source and distributable under GPL
- R and S code are largely exchangeable

GNU's S

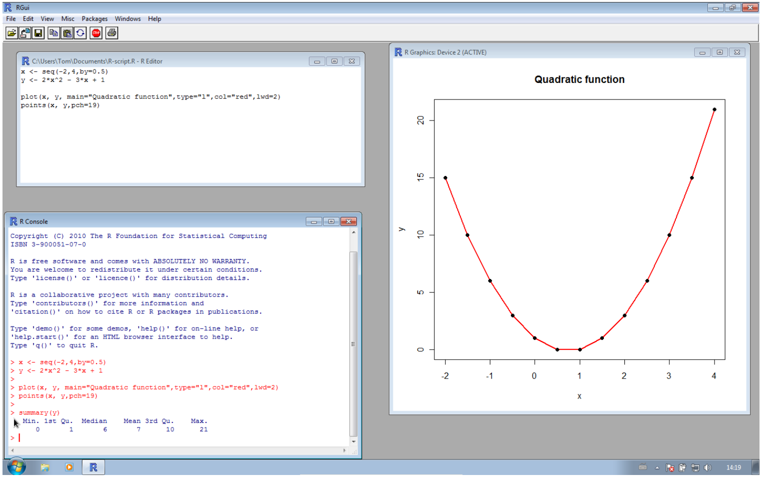
- Statistics and graphics
- Open source and distributable under GPL
- R and S code are largely exchangeable
- Plugin packages
- good connectivity with other software packages
- platform independent (Unix-alikes, MacOS, Windows)

Learning R

- Learning R consumes more time than for commercial packages
- Still many facilities for learning R
  - R help files
  - Tutorials
  - R Site Search (search engine)
  - Forums and mailinglists
  - Large community, producing and sharing code and small package-related tutorials
  - Best accessible with standard search engine (Google, ...)

Talking to R

- A matter of question and answer


RGui
File Edit View Misc Packages Windows Help
C:\Users\Tom\Documents\R-script.R - R Editor
R Graphics: Device 2 (ACTIVE)
Quadratic function
y
x
R Console
14:19

```
x <- seq(-2,4,by=0.5)
y <- 2*x^2 - 3*x + 1

plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
points(x, y,pch=19)
```

```
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> x <- seq(-2,4,by=0.5)
> y <- 2*x^2 - 3*x + 1
>
> plot(x, y, main="Quadratic function",type="l",col="red",lwd=2)
> points(x, y,pch=19)
>
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       1       6       7      10      21
> 
```

Built-in R help facilities

- ?<command>
  Examples:

```
> ?help
> ?plot
> help(plot)
```

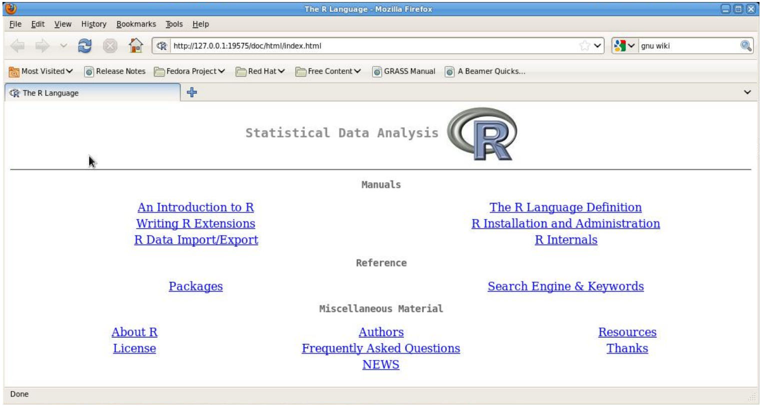
- ??<command>, which searches for string matches in help files. The function names are returned:

```
> ??plot
> ??line
> help.search("line")
```

- R help starting page launched in a web browser:

```
> help.start()
```

Built-in R help facilities


The R Language - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1:19575/doc/html/index.html
gnu wiki
Most Visited
Release Notes
Fedora Project
Red Hat
Free Content
GRASS Manual
A Beamer Quicks...
The R Language
Done

Statistical Data Analysis

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

Reference

Packages
Search Engine & Keywords

Miscellaneous Material

About R
License
Authors
Frequently Asked Questions
NEWS
Resources
Thanks

Figure: Start page for the R help files.

The beginning

- Assignments, comments, and retrieving variables

```
> var1 <- "some value"
> var1                   # type varname to retrieve content
[1] "some value"
> counter <<- 1
> counter
[1] 1
> temp = TRUE
> temp
[1] TRUE
```

- Exact meaning of different assignments: `help(assignOps)`

Modes, types, and classes

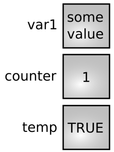
- Data type determined upon assignment
- Internal storage mode
- Simple types combined in more complex data structures
- Single mode data structures (vectors, matrices, arrays)
- Multiple mode data structures (lists, data frames, externally defined classes)

Table: R data types and internal storage modes:

types	**modes**
logical	logical
integer	numeric
double	numeric
complex	complex
character	character
symbol	name
raw	raw
list	list
NULL	NULL
...	

Single values

```
> var1 <- "some value"
> counter <<- 1
> temp = TRUE
```



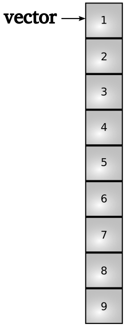
This is actually a single valued vector:

```
> length(var1)
[1] 1
```

Vectors

Extension in one dimension:

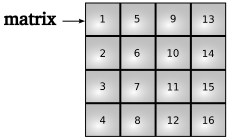
```
> b <- 1:9
> b
[1] 1 2 3 4 5 6 7 8 9

> c <- seq(0,3,by=0.5)
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0

> temp = c(T,F,F,T)
> temp
[1]  TRUE FALSE FALSE  TRUE

> rep(c(1,3:1),times=2)
[1] 1 3 2 1 1 3 2 1
```



Matrices

Extension to two dimensions:

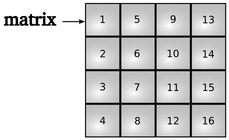
```
> mat <- matrix(data=seq(1,9,by=1),nrow=3)
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> dim(mat)
[1] 3 3

> mat[1,]
[1] 1 4 7

> mat[,1]
[1] 1 2 3
```



Arrays

Extension to n dimensions:

```
> mat4d <- array(data=seq(1,36,by=1),dim=c(3,3,2,2))
> mat4d
, , 1, 1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

 , , 2, 1
      [,1] [,2] [,3]
 [1,]   10   13   16
 [2,]   11   14   17
 [3,]   12   15   18

 , , 1, 2
      [,1] [,2] [,3]
 [1,]    1    4    7
 [2,]    2    5    8
 [3,]    3    6    9

 , , 2, 2
     [,1] [,2] [,3]
 [1,]   10   13   16
 [2,]   11   14   17
 [3,]   12   15   18
```

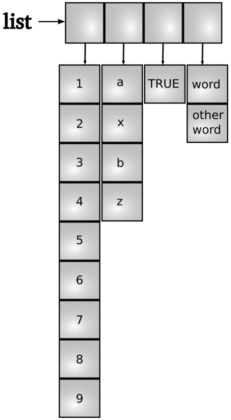


Lists

- Combine different data types:

```
> datarecord <- list(name="Tom Van Engeland",
+                    Department="Ecosystem Studies",
+                    Functions=c(PhD=4,Postdoc=1),
+                    Carpooling=T
+                   )
```

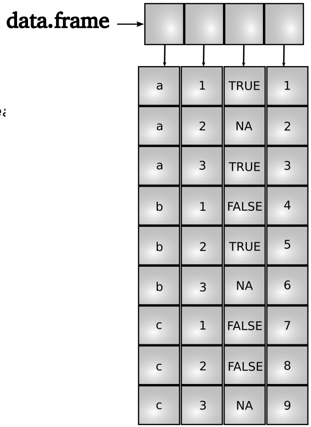
- Different types
- Different sizes



Data frames

- List elements of equal size - a data table

```
> dataset <-
+   data.frame(treatment=
+             rep(c("control","level 1","level 2"),ea
+             replicate=rep(1:3,times=3),
+             value=c(2,6,4,8,9,8,3,2,1)
+           )
> dataset
  treatment replicate value
1   control         1     2
2   control         2     6
3   control         3     4
4   level 1         1     8
5   level 1         2     9
6   level 1         3     8
7   level 2         1     3
8   level 2         2     2
9   level 2         3     1
```



Accessing objects

- to see the content:

```
varname
```

- to see the structure:

```
str(object)
ls.str()
```

- to acquire a more specific part of the information:

```
summary(object)
length(vector)
ncol(matrix)
nrow(matrix)
colnames(matrix)
rownames(matrix)
names(list)
dimnames(array)
subset(dataframe, rowselection, select=columnnames)
```

- indexing

```
matrix[rownumber,columnnumbers]
matrix[rownames,columnnames]
list[indexnumber]       (pointer to list element)
list[[indexnumber]]     (list element at place <indexnumber>)
list$listelement
```

- See help files on package specific functions

Operators

Arithmetic operators	
`x + y`	addition
`x - y`	subtraction
`x * y`	multiplication (element-wise)
`x / y`	division
`x %/% y`	integer division
`x %% y`	modulo
`x ^ y`	power
`x %*% y`	matrix multiplication
Logic operators (element-wise)	
`! x`	negation
`x & y`	AND
`x | y`	OR
`xor(x,y)`	XOR

Type `?Arithmetic`,`?Logic`,`?Special`,or `?Syntax` for more information !

Managing Your R Environment

- Get all objects from an environment:

```
> ls()
```

- Search for objects in an environment:

```
> find("search string")
> apropos("search string")
```

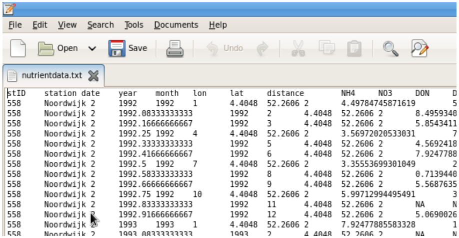
- Remove objects

```
> rm(object1, object2)
> rm(list=ls())
```

Text files

Provide a flexible means for exchange of small datasets

```
> nutrientdata <- read.table(file="Data/nutrientdata.txt",sep="\t",header=T)
> NoordwijkTransect <- nutrientdata[grep("Noordwijk",nutrientdata$station),]
> write.csv(x=NoordwijkTransect,file="Data/NoordwijkTransect.csv",
+           row.names=F,quote=F)
```



```
stID    station date    year    month   lon     lat     distance        NH4     NO3     DON     D
558     Noordwijk 2     1992    1992    1       4.4048  52.2606 2       4.49784745871619        5
558     Noordwijk 2     1992.08333333333        1992    2       4.4048  52.2606 2       8.4959340
558     Noordwijk 2     1992.16666666667        1992    3       4.4048  52.2606 2       5.8543411
558     Noordwijk 2     1992.25 1992    4       4.4048  52.2606 2       3.56972020533031        7
558     Noordwijk 2     1992.33333333333        1992    5       4.4048  52.2606 2       4.5692418
558     Noordwijk 2     1992.41666666667        1992    6       4.4048  52.2606 2       7.9247788
558     Noordwijk 2     1992.5  1992    7       4.4048  52.2606 2       3.35553699301049        2
558     Noordwijk 2     1992.58333333333        1992    8       4.4048  52.2606 2       0.7139440
558     Noordwijk 2     1992.66666666667        1992    9       4.4048  52.2606 2       5.5687635
558     Noordwijk 2     1992.75 1992    10      4.4048  52.2606 2       5.99712994495491        3
558     Noordwijk 2     1992.83333333333        1992    11      4.4048  52.2606 2       NA      N
558     Noordwijk 2     1992.91666666667        1992    12      4.4048  52.2606 2       5.0690026
558     Noordwijk 2     1993    1993    1       4.4048  52.2606 2       7.92477885583328        1
558     Noordwijk 2     1993.08333333333        1993    2       4.4048  52.2606 2       NA      N
```

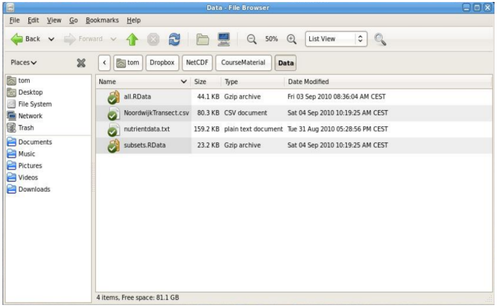
.RData files

- Write individual data objects to a R-specific binary data format (.RData):

```
> coastalData <- subset(nutrientdata, distance < 10)
> save(coastalData,NoordwijkTransect, file="Data/subsets.RData")
> load(file="Data/subsets.RData")
```

- Save all objects from an environment to an .RData file:

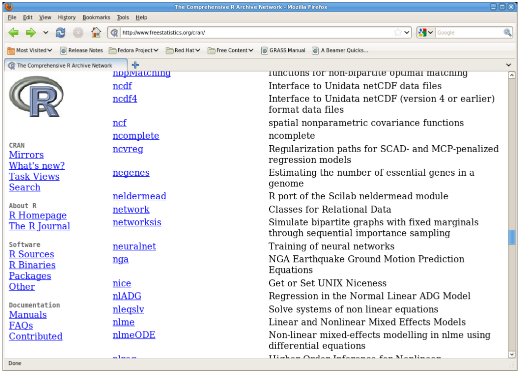
```
> save.image(file="Data/all.RData")
> load(file="Data/all.RData")
```


Data - File Browser
File Edit View Go Bookmarks Help
Back Forward 50% List View
Places
tom Dropbox NetCDF CourseMaterial Data
tom
Desktop
File System
Network
Trash
Documents
Music
Pictures
Videos
Downloads

Name	Size	Type	Date Modified
all.RData	44.1 KB	Gzip archive	Fri 03 Sep 2010 08:36:04 AM CEST
NoordwijkTransect.csv	80.3 KB	CSV document	Sat 04 Sep 2010 10:19:25 AM CEST
nutrientdata.txt	159.2 KB	plain text document	Tue 31 Aug 2010 05:28:56 PM CEST
subsets.RData	23.2 KB	Gzip archive	Sat 04 Sep 2010 10:19:25 AM CEST

4 items, Free space: 81.1 GB

Packages


The Comprehensive R Archive Network - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.freestatistics.org/cran/
Most Visited Release Notes Fedora Project Red Hat Free Content GRASS Manual A Beamer Quicks...
The Comprehensive R Archive Network
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed
Done

nbpMatching	functions for non-bipartite optimal matching
ncdf	Interface to Unidata netCDF data files
ncdf4	Interface to Unidata netCDF (version 4 or earlier) format data files
ncf	spatial nonparametric covariance functions
ncomplete	ncomplete
ncvreg	Regularization paths for SCAD- and MCP-penalized regression models
negenes	Estimating the number of essential genes in a genome
neldermead	R port of the Scilab neldermead module
network	Classes for Relational Data
networksis	Simulate bipartite graphs with fixed marginals through sequential importance sampling
neuralnet	Training of neural networks
nga	NGA Earthquake Ground Motion Prediction Equations
nice	Get or Set UNIX Niceness
nlADG	Regression in the Normal Linear ADG Model
nleqslv	Solve systems of non linear equations
nlme	Linear and Nonlinear Mixed Effects Models
nlmeODE	Non-linear mixed-effects modelling in nlme using differential equations
nlreg	Higher Order Inference for Nonlinear

- Only loaded when needed
- About 25 default packages
- Other packages managed on the CRAN website
- Install binary or from source after compilation

Packages

- Installation

```
> install.packages("nlme",dependencies=T)
```

- Loading

```
> library(nlme)
> require(nlme)
```

  Require is mainly intended to be called inside other functions, to allow for exception handling and a decent termination of the relevant function.

- Other functionalities:

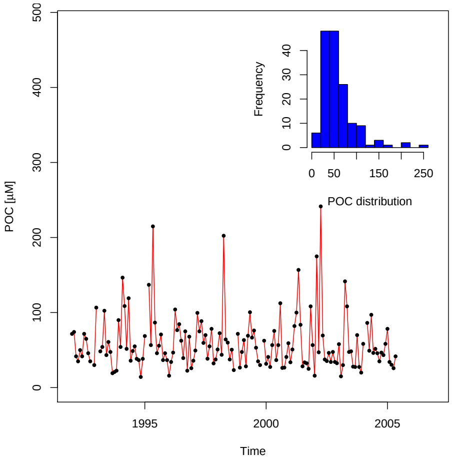
```
> update.packages()
> available.packages()
> installed.packages()
> remove.packages()
```

Statistics and Graphs with R

- main purpose = data analysis
- production of publication quality graphs

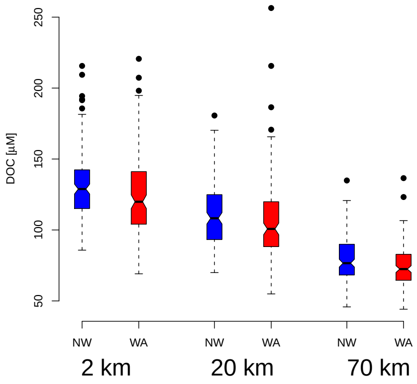
Example 1:

```
> Noordwijk <- subset(coastalData,
+                     station=="Noordwijk 2")
> par(mar=c(4,4,0.5,0.5))
> plot(Noordwijk$date,Noordwijk$POC,
+      xlab="Time",
+      ylab=expression("POC ["*mu*"M]"),
+      type="l",col="red",
+      ylim=c(0,max(Noordwijk$POC,na.rm=T)*2))
> points(Noordwijk$date,Noordwijk$POC,
+        pch=19,cex=0.6)
> par(new=T,fig=c(0.55,0.95,0.55,0.95),
+     mar=c(4,4,0,0))
> hist(Noordwijk$POC,main="", col="blue",
+      xlab="POC distribution")
```



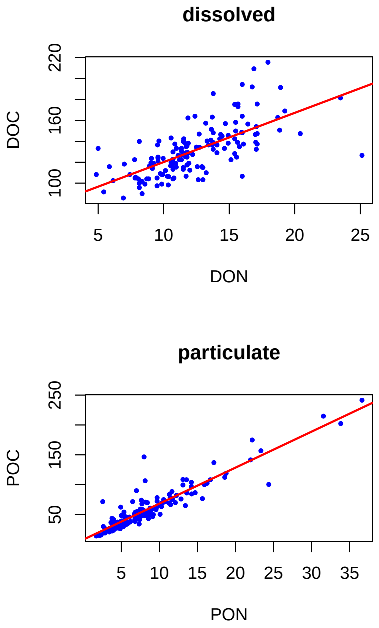
Example 2:

```
> pdf(file="Illustrations/boxplots.pdf",
+     width=6.5,height=6.5)
> boxplot(DOC~station*distance,
+         data=nutrientdata,notch=T,
+         col=c("blue","blue","blue",
+               "red","red","red"),
+         axes=F,
+         ylab=expression("DOC ["*mu*"M]"),
+         pch=19)
> axis(2)
> axis(1,at=c(1,4,8,11,15,18),
+      labels=c("NW","WA","NW",
+               "WA","NW","WA"))
> mtext(1,outer=T,line=-2,
+       at=c(0.23,0.53,0.83),
+       text=c("2 km","20 km","70 km"),
+       cex=2)
> dev.off()
```



Example 3:

```
> Dmodel <- lm(DOC ~ DON, data=Noordwijk)
> Pmodel <- lm(POC ~ PON, data=Noordwijk)
> par(mfrow=c(2,1))
> plot(Noordwijk$DON,Noordwijk$DOC,
+      main="dissolved", xlab="DON",ylab="DOC",
+      type="p", pch=19,cex=0.5,col="blue")
> abline(coef(Dmodel),col="red",lwd=2)
> plot(Noordwijk$PON,Noordwijk$POC,
+      main="particulate", xlab="PON",ylab="POC",
+      type="p",pch=19,cex=0.5,col="blue")
> abline(coef(Pmodel),col="red",lwd=2)
> mtext(side=1,outer=T,line= 1, at=0.5,
+       text="organic matter C/N ratios")
```



Writing your own functions

- to repeat certain tasks

```
someFunctionName <- function (set, of, paramaters) {
                      code block
                    }
```

- Example:

```
CNratioExt <- function(carbon, nitrogen, inverse=F) {
                if(!(is.vector(carbon) & is.vector(nitrogen))) {
                 print("This function only accepts vectors")
                 return()
                }                   # Test vector types
                if(length(carbon) != length(nitrogen)) {
                 print("carbon and nitrogen dataset not of same size")
                 return()
                }                   # Checking equality of their lengths
                if(inverse) {
                 return(nitrogen/carbon)
                }
                else {
                 return(carbon/nitrogen)
                }
              }
```

Flow control

- for-loop

```
for (i in seq(2,10,by=0.01) {code block}
```

- while-loop

```
while (test == TRUE) {code block}
```

- if-then-else construct

```
if (test == TRUE) {code block}
else {other code block}
```

- switch construct

```
switch(expression,
choice1 = {code block},
choice2 = {code block},
choice3 = {code block}
```

- ?Control for more information

R Exercise

NetCDF and OpenDAP

NetCDF

NetCDF = Network Common Data Form

Unidata (UCAR; University Corporation for Atmospheric Research) `http://www.unidata.ucar.edu`

Set of

- data formats
- programming interfaces
- software libraries (abstract data type)

NetCDF (Network Common Data Form)

- binary file
- regular N-dimensional arrays
- contains metadata (self-descriptiveness)
  = data on the data (crucial side-information)
- platform-independent (portability)

The data model

- dimensions
  - describe the extent of the variables' arrays
  - axes of variability
  - declare lengths of variables
  - name + length
  - fixed or unlimited
    a dynamically adjustable dimension that can change throughout the creation of the file with no performance penalty
- variables
  - the actual n-dimensional arrays of data
  - fixed data type (char, byte, short, int, float, double)
  - coordinate variables: data counterpart of dimensions
    (same name)
  - and just "variables"
- attributes
  - metadata
  - scalar or 1D-array (vector)
  - associated with variable or entire dataset (global)
  - some reserved

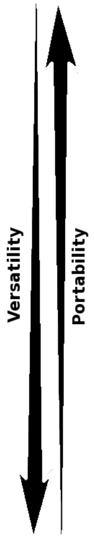
NetCDf versions

- Classical netCDF data model
  - data model as presented here
  - severe size restrictions
    (mainly for physico-chemical data; less for biological data)
  - only one unlimited dimension
  - only predefined set of base types
- Version 3.6.0
  - allowed for 64-bit based addresses
    => max size doubled (4 GB)
- Common Data Model and HDF5 (NetCDF 4.1.1)
  - more data structuring (grouping; hierarchical structuring)
  - ?addition information on the data?
  - more unlimited dimensions
  - new base types
  - user-defined data types

NetCDf versions

- Classical netCDF data model
  - data model as presented here
  - severe size restrictions
    (mainly for physico-chemical data; less for biological data)
  - only one unlimited dimension
  - only predefined set of base types
- Version 3.6.0
  - allowed for 64-bit based addresses
    => max size doubled (4 GB)
- Common Data Model and HDF5 (NetCDF 4.1.1)
  - more data structuring (grouping; hierarchical structuring)
  - ?addition information on the data?
  - more unlimited dimensions
  - new base types
  - user-defined data types

Backward compatible

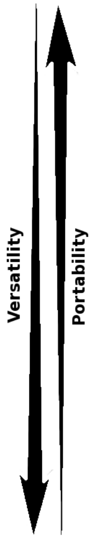


NetCDf versions

Default

- Classical netCDF data model
  - data model as presented here
  - severe size restrictions
    (mainly for physico-chemical data; less for biological data)
  - only one unlimited dimension
  - only predefined set of base types
- Version 3.6.0
  - allowed for 64-bit based addresses
    => max size doubled (4 GB)
- Common Data Model and HDF5 (NetCDF 4.1.1)
  - more data structuring (grouping; hierarchical structuring)
  - ?addition information on the data?
  - more unlimited dimensions
  - new base types
  - user-defined data types

Backward compatible



What about relational databases?

- no relationships between data points
- no efficient storage and access to gridded data
- extra costs involved in DBMS do not pay-off
  as useful functionality (concurrency control, access rights, views, ...)

NetCDF is intended for gridded data which are more efficiently presented and managed using (ir)regular grids than by relationship (relational databases)

BUT Relational databases to link dataset through metadata?

Metadata

- Rules of application partially delineated but not enforced
  tension between self-descriptiveness, dedication, and flexibility
- Provision left to user
- Standards and conventions needed
- Examples: `http://www.unidata.ucar.edu/netcdf/conventions.html`
- Which metadata needed for ecological variables?

Advantages & disadvantages

- freely available
- data + variable description + creation history + other attributes
- reasonably compact portable binary format
- allows extraction of parts of large data file
- possibility to write your own interface (e.g. R-packages)

BUT

- no 'ragged' objects
- extensive use of special 'missing values' may be needed

NetCDF with R

R packages

- Available packages
  - ncdf
  - RNetCDF
    Not for OSX
  - ncdf4
    - Has netCDF 4 capability
    - Has OpenDAP capability
      if compiled properly
    - Does currently not work on Windows
- Installation
  - Windows: binaries at CRAN
  - Linux and OSX: need to install NetCDF libraries first

```
R CMD INSTALL --configure-args="
-with-netcdf_incdir=/path/to/netcdf/include
-with-netcdf_libdir=/path/to/netcdf/lib"
ncdf_xx.yy.tar.gz
```

The ncdf package

- reading open.ncdf(...) get.var.ncdf(...)
- writing open.ncdf(...,write=T) put.var.ncdf(...)
- creation create.ncdf(...) dim.def.ncdf(...) var.def.ncdf(...)

Downloading and opening a NetCDF file

```
> url_grid <-  "http://opendap.deltares.nl/opendap/
+               rijkswaterstaat/vaklodingen/KB117_4746.nc"
> download.file(url=url_grid,
+               destfile="vaklodingenKB117_4746.nc",
+               method =  "auto",
+               quiet = FALSE,
+               mode="wb",
+               cacheOK = TRUE)
grid.nc <- open.ncdf("vaklodingenKB117_4746.nc")
```

Querying a local NetCDF file

```
names(grid.nc)
```

Name	Description
id	a file id
ndims	number of dimensions
natts	number of global attributes
unlimdimid	id of unlimited dimension or -1 if not present
filename	filename on disk
writable	file opened for writing of not
dim	dimensions (named list of objects of class dim.ncdf)
nvars	number of non-coordinate variables
var	variables (named list of objects of class var.ncdf)

Querying a local NetCDF file

Finding the variables and dimensions:

```
> names(grid.nc$var)
[1] "crs" "z"   "lon" "lat"
> names(grid.nc$dim)
[1] "time" "y"    "x"
> grid.nc$dim$time
$name
[1] "time"
$len
[1] 60
$unlim
[1] TRUE
$id
[1] 1
$dimvarid
[1] 2
$units
[1] "days since 0001-01-01 00:00:00 +1:00"
$vals
 [1] 713684 714415 714780 715145 715510 715876 716241 716606 716971 717337 717702 718067
[13] 718432 718798 719163 719528 719893 720259 720624 720989 721354 721720 722085 722450
[25] 722815 723546 724276 724642 725007 725372 725737 726103 726468 726833 727198 727564
[37] 727929 728294 728659 729025 729030 729176 729390 729395 729541 729755 729760 729906
[49] 730120 730125 730272 730486 730851 730856 731002 731216 731221 731367 731581 731947
$create_dimvar
[1] TRUE

attr(,"class")
[1] "dim.ncdf"
```

Querying a local NetCDF file

Extraction of the variables:

Try `grid.nc$var$z`

... There is no data?

```
> x <- get.var.ncdf(nc=grid.nc,varid="x")
> y <- get.var.ncdf(grid.nc,varid="y")
> time <- get.var.ncdf(grid.nc,"time")
> z <- get.var.ncdf(grid.nc,"z")
> list(x=dim(x),y=dim(y),z=dim(z))

$x
[1] 500

$y
[1] 625

$z
[1] 500 625  60
```

Variables are read from the local file...

Querying a local NetCDF file

- What about the attributes?
- No function to get a list of attributes?
- Attributes are scatter all over the object...

```
library(RNetCDF)
alternative <- open.nc("/home/tom/Dropbox/NetCDF/vaklodingenKB117_4746.nc")
print.nc(alternative)
```

Querying a local NetCDF file

- What about the attributes?
- No function to get a list of attributes?
- Attributes are scatter all over the object...

```
library(RNetCDF)
alternative <- open.nc("/home/tom/Dropbox/NetCDF/vaklodingenKB117_4746.nc")
print.nc(alternative)
```

- Or ask them from the OpenDAP server...

Querying a local NetCDF file

```
> getInfo(url_grid,what="dds") # cf. course
Dataset {
     Int32 crs;
     Grid {
      ARRAY:
         Float64 lat[y = 625][x = 500];
      MAPS:
         Float64 y[y = 625];
         Float64 x[x = 500];
     } lat;
     Grid {
      ARRAY:
         Float64 z[time = 10][y = 625][x = 500];
      MAPS:
         Float64 time[time = 10];
         Float64 y[y = 625];
         Float64 x[x = 500];
     } z;
     Grid {
      ARRAY:
         Float64 lon[y = 625][x = 500];
      MAPS:
         Float64 y[y = 625];
         Float64 x[x = 500];
     } lon;
     Float64 y[y = 625];
     Float64 x[x = 500];
     Float64 time[time = 10];
 } opendap/rijkswaterstaat/vaklodingen/vaklodingenKB117_4746.nc;
```

Try getInfo(url_grid,what="das") ...

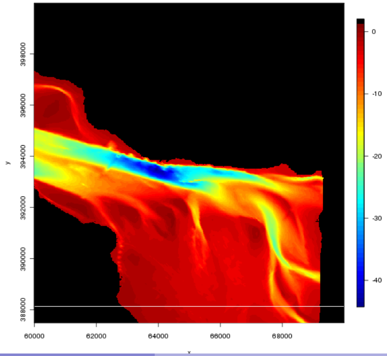
Querying a local NetCDF file

Other extensions for OpenDAP querying:

- .help : generates a web based help document
- .html : generates a web page containing information and action buttons over the netcdf file
- .info : returns the netcdf file metadata
- .dds : returns information about the variables
- .das : returns information about the variable attributes
- .ver : returns the OpenDAP server version
- .asc : returns the netcdf file in ASCII format

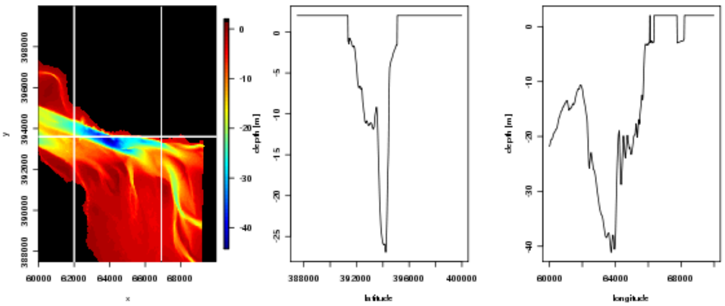
Plotting NetCDF data

```
> # The plot function that is used needs variable values in ascending order
> y <- rev(y)
> z <- z[,(ncol(z):1),]
> # select only the slices, containing data
> selection <- which(apply(z,MARGIN=3,
+                          FUN=function(a) diff(range(a,na.rm=T)) != 0,
+                    arr.ind=T)
> z <- z[,,selection]
> # Replace -9999 with some value closer to the relevant range
> z[z == -9999] <- max(z,na.rm=T)+1
> # the dimensions as headers in the depth array is more informative
> # Standard calendar has 365.242 days in a year...
> dimnames(z) <- list(x,y,round(
+                 time[selection]/365.242))
> # Plot the first non-empty time slice
> image.plot(x,y,z[,,1],
+            col = c(tim.colors(),"black"))
```



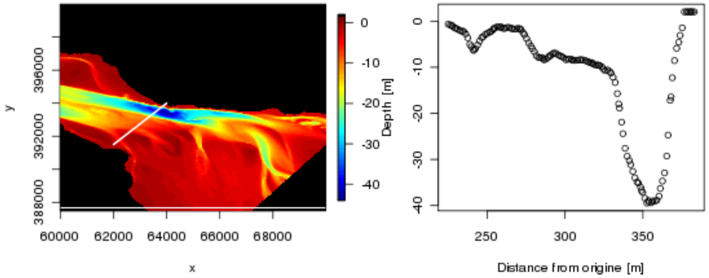
Working with a data subset

```
> par(mfrow=c(1,3))
> image.plot(x,y,z[,,1],col = c(tim.colors(),"black"))
> abline(v=62000,h=393600,col="white",lwd=2)
> plot(y,z["62000",,1],type="l",xlab="latitude",ylab="depth [m]")
> plot(x,z[,"393600",1],type="l",xlab="longitude",ylab="depth [m]")
```



Working with a data subset

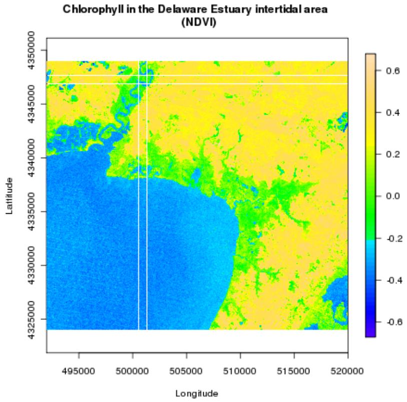
```
> depthProfile <- transect(datamatrix=z[,,"1988"],
+                          startpoint = c(62000,391500),
+                          endpoint = c(64000,394000),
+                          method="dimension")
>                       # cf. course for function....
> par(mfrow=c(1,2))
> image.plot(x,y,z[,,"1988"],col = c(tim.colors(),"black"))
> segments(x0=depthProfile$start[1],y0=depthProfile$start[2],
+          x1=depthProfile$end[1],
+          y1=depthProfile$end[2],lwd=2,col="white")
> plot(depthProfile$distance,depthProfile$values,
+     xlab="Distance from origine [m]",ylab="Depth [m]")
```



Writing NetCDF files

Suppose: Landsat 7 spectral bands as ArcInfo ASCII raster files

```
> library(adehabitat)
> ETM3 <- import.asc(file="/home/tom/Desktop/ETM3.asc")
> ETM4 <- import.asc(file="/home/tom/Desktop/ETM4.asc")
> NDVI <- (ETM4-ETM3)/(ETM4+ETM3)   # NDVI = (NIR - red) / (NIR + red)
> image.plot(NDVI,col=topo.colors(64),
+           main="Chlorophyll in the Delaware Estuary
+                 intertidal area \n (NDVI)",
+           xlab="Longitude",ylab="Lattitude")
```



Writing NetCDF files

- We want to store this NDVI map in a NetCDF file.
- Steps to follow:
  - Create dimensions (`dim.ncdf class`)
  - Create non-coordinate variables (`var.ncdf class`)
  - Create ncdf file in memory (`create.ncdf()`)
  - Add the variable content to the NetCDF file (`put.var.ncdf()`)
  - Close the file, causing the entire buffer to be written to disk
  - Add attributes (metadata) to the file after re-opening in writeable mode

Writing NetCDF files

- Create dimensions

```
# Define R variables with the dimension values, derived from the ESRI file
x <- seq(attr(NDVI,"xll"),length.out=dim(NDVI)[1],by=attr(NDVI,"cellsize"))
y <- seq(attr(NDVI,"yll"),length.out=dim(NDVI)[2],by=attr(NDVI,"cellsize"))

# Define the dimension objects (dim.ncdf), based on the R variables above
x_dim <- dim.def.ncdf(name="x",units="m", vals=x, unlim=F, create_dimvar=T)
y_dim <- dim.def.ncdf(name="y",units="m", vals=y, unlim=F, create_dimvar=T)
```

Writing NetCDF files

- Create non-coordinate variables

```
# Define a 0 dimensional variable (var.ncdf)
# Only needed to attach metadata, cf. later)
crs_var <- var.def.ncdf.no_dim(name="crs",units="",dim=numeric(0),
                               longname="UTM 18N",prec="integer",
                               missval=NA)
        # This is a derived function from the ncdf package
        # see remark 2 for function
# Define a NetCDF variable
ndvi_var <- var.def.ncdf(name="ndvi",units="",dim=list(x_dim,y_dim),
                         longname="Normalized Difference Vegetation Index",
                         prec="double",missval=-9999)
```

Writing NetCDF files

- Create ncdf file in memory

```
> ndvi.nc <- create.ncdf("/home/tom/Dropbox/NetCDF/CourseMaterial/Data/NDVI.nc",
+                          vars=list(crs_var,ndvi_var),verbose=T)
```

- Add the variable content to the NetCDF file

```
> put.var.ncdf(nc=ndvi.nc, varid="ndvi",vals=NDVI,start=c(1,1),count=c(-1,-1))
```

- Close the file, allowing for the entire buffer to written to disk

```
> close.ncdf(ndvi.nc)
```

Writing NetCDF files

- Re-open the file to add extra attributes

```
> ndvi.nc <- open.ncdf("/home/tom/Dropbox/NetCDF/CourseMaterial/Data/NDVI.nc",
+                      write=T)
```

- Which attributes to add?
- Different conventions and standards

  (CF convention is used frequently)
- Here we will work in three step:
  - Global attributes

    (refer to the dataset as a whole)
  - Non-coordinate variable attributes

    (refer to individual variables)
  - Information on the coordinate system

    (implemented by adding an extra variable)

Writing NetCDF files

- Global attributes accoding to the CF convention
  (title, institution, source, history, references, comment)

```
att.put.ncdf(nc=ndvi.nc,varid=0,attname="title",
             attval="NDVI of the Delaware Estuary Intertidal Area")
att.put.ncdf(nc=ndvi.nc,varid=0,attname="institution",attval="NIOO-CEME")
att.put.ncdf(nc=ndvi.nc,varid=0,attname="source",
             attval="Landsat 7 Enhanced Thematic Mapper")
att.put.ncdf(nc=ndvi.nc,varid=0,attname="history",
             attval="The image cube was downloaded from the NASA website
                     and processed using GRASS 6.3")
att.put.ncdf(nc=ndvi.nc,varid=0,attname="references",attval="http://www.nasa.gov/")
att.put.ncdf(nc=ndvi.nc,varid=0, attname="comment",
             attval='The original Landsat image cube, downloaded as Erdas
                     Imagine format (.img), was imported into GRASS 6.3 using
                     the GDAL libraries for conversion. The bands 3 (red) and
                     4 (NIR; near infra-red) were selected for export as
                     ESRII ArcInfo ASCII files. These were read into the R
                     Statistical Software via the import.asc function of
                     the adehabitat package. The NDVI was calculated from
                     these bands, and exported as this NetCDF file.')
```

Writing NetCDF files

- Information on the coordinate system
- In CF convention implemented as attributes to a non-sense variable
- Problem: ncdf requires non-zero dimensions (bug or programming decision?)
- Flexibility <=> stringency in metadata provision
- Should a general-purpose package enforce extra requirements without taking into account generally applied conventions and standards?
- On the other hand: Should we leave metadata provision up to the discretion of the individual scientist?

"Adding all this extra information is time consuming and not productive" on the short term!

"I know my data ..."

Writing NetCDF files

- Information on the coordinate system
- After some change to the some relevant ncdf functions (cf. course remarks)

```
att.put.ncdf(nc=ndvi.nc,varid="crs",attname="spatial_ref",attval='
"PROJCS["WGS 84 / UTM zone 18N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.01745329251994328,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-75],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    AUTHORITY["EPSG","32618"],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH]]"
')   # nesting of ' and "
```

Writing NetCDF files

- Information on the coordinate system
- After some change to the some relevant ncdf functions (cf. course remarks)

```
    ....
        UNIT["degree",0.01745329251994328,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-75],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    AUTHORITY["EPSG","32618"],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH]]"
')    # nesting of ' and "
att.put.ncdf(nc=ndvi.nc,varid="crs",
             attname="grid_mapping_name",
             attval="transverse_mercator")
att.put.ncdf(nc=ndvi.nc,varid="crs",
             attname="scale_factor_at_central_meridian",
             attval="0.9996")
....
```

Writing NetCDF files

- Information on the coordinate system
- Additions to the coordinate values

```
# x coordinate
att.put.ncdf(ndvi.nc,varid="x",attname="standard_name",
             attval="projection_x_coordinate")
att.put.ncdf(ndvi.nc,varid="x",attname="_FillValue",
             attval="-9999")
# y coordinate
att.put.ncdf(ndvi.nc,varid="y",attname="standard_name",
             attval="projection_y_coordinate")
att.put.ncdf(ndvi.nc,varid="y",attname="_FillValue",
             attval="-9999")
close.ncdf(ndvi.nc)    # clear buffer and write to disk...
```

- Under certain conditions the standard_name attribute should have a predetermined value
- _FillValue is obligatory in the CF conventions, whereas missing_value is abandoned as obligatory

At work with your own data

Thank you for your attention !

Contact:
Tom Van Engeland
NIOZ-CEME
Department of Ecosystem Studies
Korringaweg 7
4401 NT Yerske
The Netherlands
tel: +31 113 577 473
email: t.vanengeland@nioo.knaw.nl
website: `http://www.nioo.knaw.nl/users/tvanengeland`