Presentation 1: R

Murray Logan

15 Jul 2017

Section 1

Preparation

Course

www.flutterbys.com.au/stats

/downloads/slides/

http://r4ds.had.co.nz/

AND NOW FOR R

Scripts and commands

- enter a command (expression) at the prompt (>)
- store commands in a script

Basic syntax

The R Environment and command line

```
> 5+1
```

```
[1] 6
```

Basic syntax

The R Environment and command line

```
> 5+1
```

```
[1] 6
```

```
> VAR1 <- 2 + 3
> VAR1
```

```
[1] 5
```

Basic syntax

The R Environment and command line

```
> 5+1
```

```
[1] 6
```

```
> VAR1 <- 2 + 3
> VAR1
```

```
[1] 5
```

```
> VAR2 <-
+ 2+3
> VAR2
```

```
[1] 5
```

Basic syntax

The R Environment and command line

```
> 5+1
```

```
[1] 6
```

```
> VAR1 <- 2 + 3
> VAR1
```

```
[1] 5
```

```
> VAR2 <-
+ 2+3
> VAR2
```

```
[1] 5
```

```
> VAR2- 1
```

```
[1] 4
```

```
> ANS1 <- VAR1 * VAR2
> ANS1
```

Your turn

- using an inbuilt constant for π (pi), calculate the area of a circle of radius 10cm

$$A = \pi r^2$$

```
> pi
```

```
[1] 3.141593
```

Your turn

- using an inbuilt constant for π (pi), calculate the area of a circle of radius 10cm

$$A = \pi r^2$$

```
> RADIUS <- 10
> AREA <- pi*RADIUS^2
> AREA
```

```
[1] 314.1593
```

```
> #OR
> pi*10^2
```

```
[1] 314.1593
```

Basic syntax

OBJECT NAMES

- must start with a letter
- cannot include a space or - + * / # % &
  [ ] { } ( ) ~

Basic syntax

OBJECT NAMES

Good practice

- avoid lower case names (conflict)
- should reflect contents
- use underscore and periods to separate words

```
> MEAN.HEAD.LENGTH <- 1
```

Basic syntax

CONCATENATING OBJECTS

```
> c(1,2,6)
```

```
[1] 1 2 6
```

```
> c(VAR1,VAR2)
```

```
[1] 5 5
```

Basic syntax

CONCATENATING OBJECTS

```
> c(1,2,6)
```

```
[1] 1 2 6
```

```
> c(VAR1,VAR2)
```

```
[1] 5 5
```

c() is a function

Basic syntax

CONCATENATING OBJECTS

```
> c(1,2,6)
```

```
[1] 1 2 6
```

```
> c(VAR1,VAR2)
```

```
[1] 5 5
```

c() is a function

```
> RADIUS <- c(10,20,30)
> AREA <- pi*RADIUS^2
> AREA
```

```
[1]  314.1593 1256.6371 2827.4334
```

Comments

```
> c(1,2,6) # ignore all after the # sign
```

```
[1] 1 2 6
```

```
> # allow you to annotate your code
```

Scripting Advice #1

1. Make your code as readable as possible
    - use meaningful names
    - make use of indentation and spaces

Scripting Advice #1

1. Make your code as readable as possible
    - use meaningful names
    - make use of indentation and spaces
2. Use comments to document what and why

Basic usage

COMMAND HISTORY

- up and down arrows

CODE COMPLETION

- hit the TAB key

Basic usage

THE WORKSPACE

```
> # list all objects created by the user in the session
> ls()
```

```
[1] "ANS1"          "AREA"          "fn"            "MEAN.HEAD.LENGTH" "RADIUS"
[6] "tab.count"         "VAR1"              "VAR2"
```

Basic usage

THE WORKSPACE

```
> # list all objects created by the user in the session
> ls()
```

```
[1] "ANS1"          "AREA"          "fn"            "MEAN.HEAD.LENGTH" "RADIUS"
[6] "tab.count"         "VAR1"              "VAR2"
```

```
> # list all objects created by the user in the session that match a speci
> ls(pat="VAR")
```

```
[1] "VAR1" "VAR2"
```

```
> ls(pat="A*1")
```

```
[1] "ANS1" "VAR1"
```

Basic usage

THE WORKSPACE

```
> rm(VAR1,VAR2)   #remove the VAR1 and VAR2 objects
> rm(list=ls())   #remove all user defined objects
```

Basic usage

THE WORKING DIRECTORY

```
> setwd("/home/murray/tmp/")  #change the current working directory path
> getwd()  #review the current working directory
```

```
[1] "/home/murray/tmp"
```

Basic usage

THE WORKING DIRECTORY

```
> setwd("/home/murray/tmp/")  #change the current working directory path
> getwd()  #review the current working directory
```

```
[1] "/home/murray/tmp"
```

Note that R uses Unix directory slashes (/) NOT Windows directory slashes (\)

```
> list.files(path=getwd()) #list all files (and directories) in the curren
```

```
 [1] "beamerHeader.sty"  "end1.matter"     "figure"          "head.template"
 [5] "images"            "junk2.dzslides"  "Makefile"        "NotebookPaper.jpg"
 [9] "pres.0.aux"        "pres.0.html"     "pres.0.log"      "pres.0.nav"
[13] "pres.0_notes.aux"  "pres.0_notes.log" "pres.0_notes.out" "pres.0_notes.
[17] "pres.0_notes.tex"  "pres.0_notes.toc" "pres.0.out"      "pres.0.pdf"
[21] "pres.0.R"          "pres.0.Rmd"      "pres.0.snm"      "pres.0.tex"
[25] "pres.0_tmp.md"     "pres.0.toc"      "pres.1.aux"      "pres.1.html"
```

Basic usage

QUITTING

```
> q()
```

NOTE - do not put such a command in a script!

Basic syntax

FUNCTIONS

- collections of commands that expand the syntax of R
- perform a single action
- parameters that alter behaviour

Basic syntax

FUNCTIONS

seq() - a function that generates sequences

```
> seq(from=2,to=10)
```

```
[1]  2  3  4  5  6  7  8  9 10
```

Basic syntax

FUNCTIONS

seq() - a function that generates sequences

```
> seq(from=2,to=10)
```

```
[1]  2  3  4  5  6  7  8  9 10
```

```
function (from = 1, to = 1, by = ((to - from)/(length.out - 1)), length.out = NULL
    ...)
```

- type seq(and then hit the TAB key

Basic syntax

FUNCTIONS

seq()

```
> seq(from=2,to=10)
```

```
[1]  2  3  4  5  6  7  8  9 10
```

```
> seq(from=2,to=10,by=2)
```

```
[1]  2  4  6  8 10
```

Basic syntax

FUNCTIONS

seq()

```
> seq(from=2,to=10)
```

```
[1]  2  3  4  5  6  7  8  9 10
```

```
> seq(from=2,to=10,by=2)
```

```
[1]  2  4  6  8 10
```

```
> seq(from=2,to=10,length.out=3)
```

```
[1]  2  6 10
```

Basic syntax

FUNCTIONS

seq()

```
> seq(from=2,to=10)
```

```
[1]  2  3  4  5  6  7  8  9 10
```

```
> seq(from=2,to=10,by=2)
```

```
[1]  2  4  6  8 10
```

```
> seq(from=2,to=10,length.out=3)
```

```
[1]  2  6 10
```

Your turn

- generate a sequence of 10 numbers that increments by 2 and starting at 8

Your turn

- generate a sequence of 10 numbers that increments by 2 and starting at 8

```
> seq(from=8, len=10, by=2)
```

```
[1]  8 10 12 14 16 18 20 22 24 26
```

Basic syntax

OVERLOADED FUNCTIONS

seq()

```
> apropos("^seq\\.")
```

```
[1] "seq.Date"    "seq.default" "seq.int"     "seq.POSIXt"
```

```
> str(seq.default)
```

```
function (from = 1, to = 1, by = ((to - from)/(length.out - 1)), length.out = NULL
    ...)
```

```
> str(seq.Date)
```

```
function (from, to, by, length.out = NULL, along.with = NULL, ...)
```

Basic syntax

OVERLOADED FUNCTIONS

seq()

```
> # create a sequence of dates spaced 7 days apart between 29th Feb 2000 a
> sampleDates <- seq(from=as.Date("2000-02-29"),
+                    to=as.Date("2000-04-30"), by="7 days")
> # print (view) these dates
> sampleDates
```

```
[1] "2000-02-29" "2000-03-07"
[3] "2000-03-14" "2000-03-21"
[5] "2000-03-28" "2000-04-04"
[7] "2000-04-11" "2000-04-18"
[9] "2000-04-25"
```

Basic syntax

OVERLOADED FUNCTIONS

```
> mean(c(1,2,3,4))
```

```
[1] 2.5
```

```
> mean(sampleDates)
```

```
[1] "2000-03-28"
```

Getting help

```
> help(mean)
```

Getting help

```
> help(mean)
```

```
> ?mean
```

Getting help

```
> help(mean)
```

```
> ?mean
```

```
> example(mean)
```

Getting help

DEMONSTRATIONS

```
> demo(graphics)   #run the graphics demo
> demo()           #list all demos available on your system
```

Getting help

SEARCHING

```
> apropos('mea')
```

Getting help

SEARCHING

```
> apropos('mea')
```

```
> help.search('mean')   #search the local R manuals
> help.start()          #search the local HTML R manuals
```

Getting help

SEARCHING

```
> apropos('mea')
```

```
> help.search('mean')   #search the local R manuals
> help.start()          #search the local HTML R manuals
```

Getting help

FUNCTION ARGUMENTS

```
> args(mean)   #the arguments that apply to the mean function
```

```
function (x, ...)
NULL
```

```
> args(mean.default)
```

```
function (x, trim = 0, na.rm = FALSE, ...)
NULL
```

```
> args(list.files)    #the arguments that apply to the list.files function
```

```
function (path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE,
    recursive = FALSE, ignore.case = FALSE, include.dirs = FALSE,
    no.. = FALSE)
NULL
```

Package management

PACKAGES

Loaded packages

```
> search()
```

```
 [1] ".GlobalEnv"        "package:knitr"     "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

Package management

PACKAGES

Loading packages

```
> library(MASS)
> search()
```

```
 [1] ".GlobalEnv"        "package:MASS"      "package:knitr"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "package:methods"
[10] "Autoloads"         "package:base"
```

Package management

PACKAGES

Loading packages

```
> library(MASS)
> search()
```

Unloading packages

```
> detach("package:MASS")
> search()
```

```
 [1] ".GlobalEnv"        "package:knitr"     "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

Package management

LISTING INSTALLED PACKAGES

```
> installed.packages()
```

```
> installed.packages(fields =
+   c("Package", "LibPath", "Version", "Depends", "Built"))
```

Package management

AVAILABLE PACKAGES

```
> available.packages()
```

LIBRARY PATH

```
> .libPaths()
```

```
[1] "/usr/local/lib/R/site-library" "/usr/lib/R/site-library"
[3] "/usr/lib/R/library"
```

```
> .libPaths()[2]
```

```
[1] "/usr/lib/R/site-library"
```

Package management

INSTALLING PACKAGES

```
> install.packages()
```

Package management

INSTALLING PACKAGES

```
> install.packages()
```

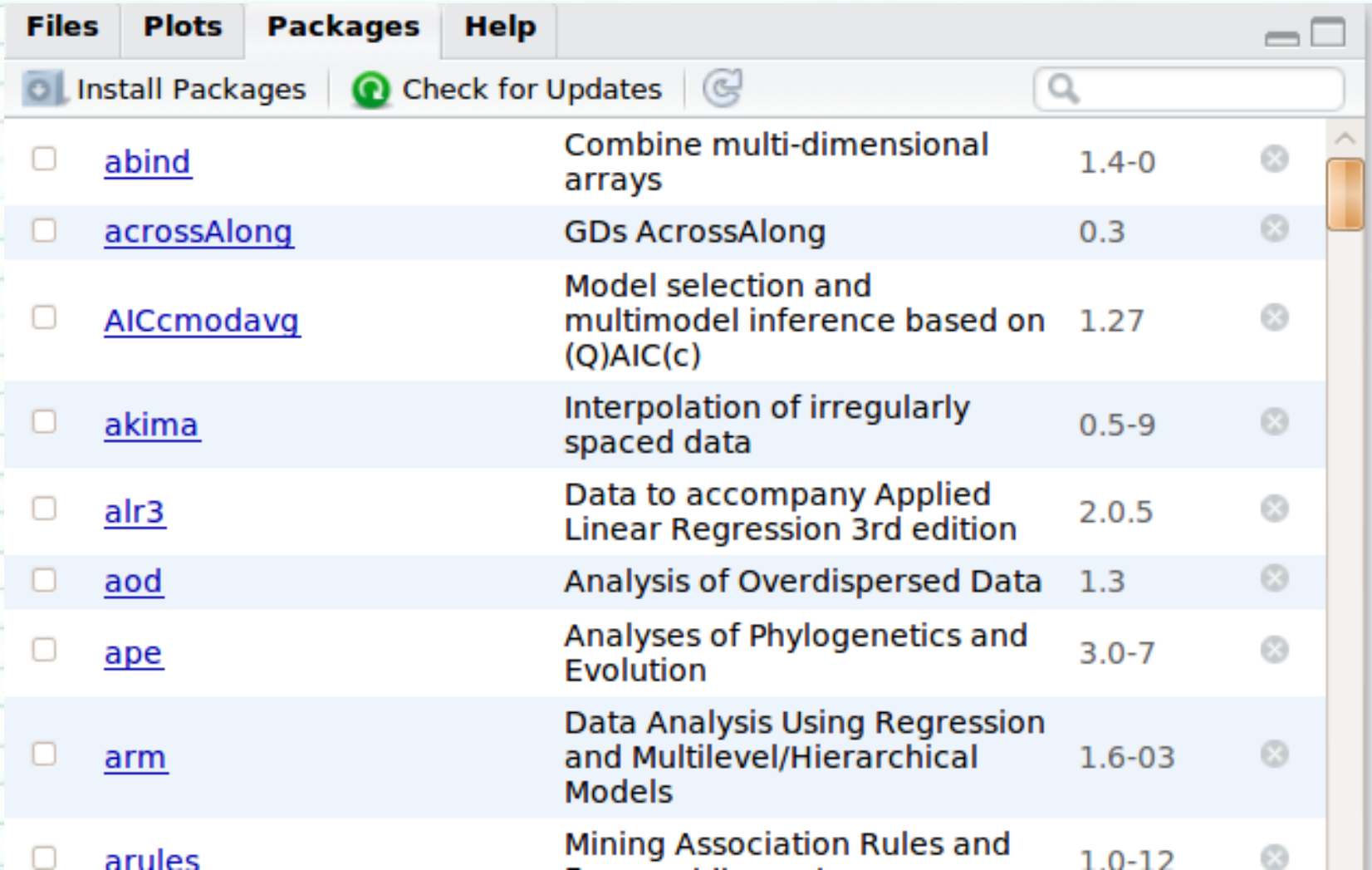
- For example, to install the car package

```
> install.packages('car')
> install.packages('car', repos='http://cran.csiro.au', lib=.libPaths()[2]
```

- Best to install packages as administrator

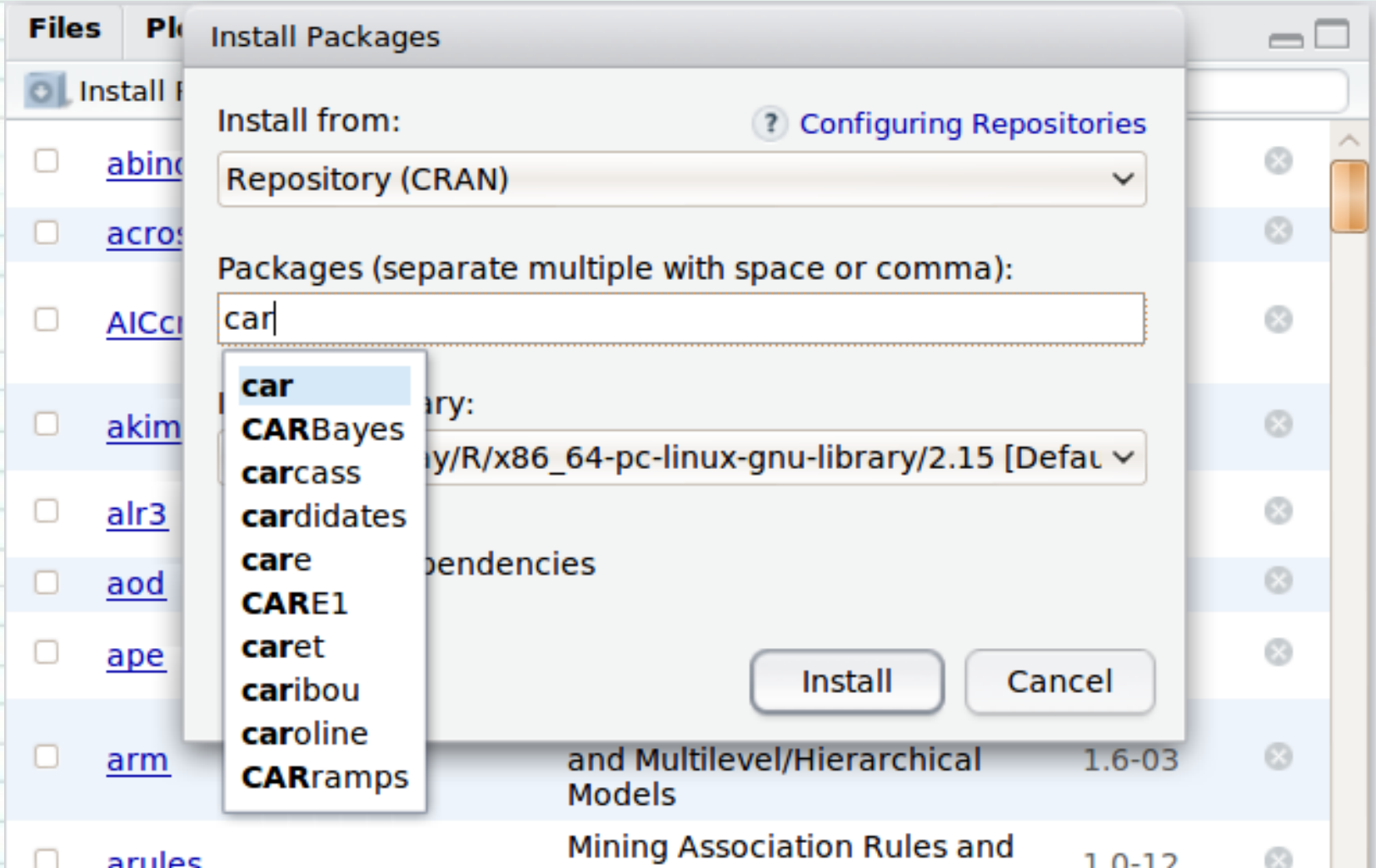
Package management (Rstudio)

AVAILABLE PACKAGES



Package management (Rstudio)

INSTALLING PACKAGES



Data types (atomic modes)

PRIMARY DATA TYPES

Type	Description
integer	whole numbers
numeric	numbers (floating points)
character	words
logical	TRUE/FALSE (0/1)

Data types

DERIVED DATA TYPES

Type	Description
factor	categorical variable
date	(Date) number of days since 1970-01-01
POSIX	(Time and Date) number of seconds since 1900-01-01 00:00:00

Dates

```
> Date <- c('2000-02-29','2002-08-20','2004-02-21')
> (Dates<-as.Date(Date))
```

```
[1] "2000-02-29" "2002-08-20" "2004-02-21"
```

```
> mean(Dates)
```

```
[1] "2002-04-24"
```

- But what about other formats!

Dates

```
> as.Date('29/02/2000', format='%d/%m/%Y')
```

```
[1] "2000-02-29"
```

```
> as.Date('29th Feb 2000', format='%dth %b %Y')
```

```
[1] "2000-02-29"
```

check out the help for `strptime`

Dates and times

```
> as.POSIXct('29/02/2000 07:22:30', format='%d/%m/%Y %H:%M:%S')
```

```
[1] "2000-02-29 07:22:30 AEST"
```

Formatting dates and times

```
> Dates
```

```
[1] "2000-02-29" "2002-08-20" "2004-02-21"
```

```
> format(Dates,format='%d/%m/%Y')
```

```
[1] "29/02/2000" "20/08/2002" "21/02/2004"
```

```
> format(Dates,format='%b %Y')
```

```
[1] "Feb 2000" "Aug 2002" "Feb 2004"
```

```
> format(Dates,format='%Y')
```

```
[1] "2000" "2002" "2004"
```

Dates and times

packages: lubridate

```
> Date
```

```
[1] "2000-02-29" "2002-08-20" "2004-02-21"
```

```
> library(lubridate)
> ymd(Date)
```

```
[1] "2000-02-29" "2002-08-20" "2004-02-21"
```

```
> dmy('29/02/2000')
```

```
[1] "2000-02-29"
```

Dates and times

packages: lubridate

```
> Dates
```

```
[1] "2000-02-29" "2002-08-20" "2004-02-21"
```

```
> library(lubridate)
> year(Dates)
```

```
[1] 2000 2002 2004
```

```
> class(Date)
```

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

```
> decimal_date(Dates)
```

```
[1] 2000.161 2002.633 2004.139
```

```
> week(Dates)
```

```
[1]  9 34  8
```

Dates and times

packages: lubridate

```
> ddays(100)
```

```
[1] "8640000s (~14.29 weeks)"
```

```
> Dates+100
```

```
[1] "2000-06-08" "2002-11-28" "2004-05-31"
```

```
> Dates+ddays(100)
```

```
[1] "2000-06-08" "2002-11-28" "2004-05-31"
```

```
> Dates+dweeks(5)
```

```
[1] "2000-04-04" "2002-09-24" "2004-03-27"
```

Dates and times

packages: lubridate

```
> dmy_hms('29/02/2000 07:22:30')
```

```
[1] "2000-02-29 07:22:30 UTC"
```

```
> dmy_hms('29/02/2000 07:22:30')+100
```

```
[1] "2000-02-29 07:24:10 UTC"
```

```
> dmy_hms('29/02/2000 07:22:30')+ddays(100)
```

```
[1] "2000-06-08 07:22:30 UTC"
```

```
> quarter(Dates)
```

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

Storage types

Type	Description
vector	1-d array (same type)
matrix	2-d array (same type and length)
list	collection of vectors
data frame	2-d array (any type, same length)

Data types

VECTORS

```
> # a numeric vector
> TEMPERATURE <- c(36.1, 30.6, 31, 36.3, 39.9, 6.5, 11.2, 12.8, 9.7, 15.9)
> TEMPERATURE
```

```
 [1] 36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7
[10] 15.9
```

Data types

VECTORS

```
> # a numeric vector
> TEMPERATURE <- c(36.1, 30.6, 31, 36.3, 39.9, 6.5, 11.2, 12.8, 9.7, 15.9)
> TEMPERATURE
```

```
 [1] 36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7
[10] 15.9
```

```
> # a character vector
> WORDS<-c('Fish', 'Rock', 'Tree', "Git")
> WORDS
```

```
[1] "Fish" "Rock" "Tree" "Git"
```

Data types

VECTORS

```
> # a numeric vector
> TEMPERATURE <- c(36.1, 30.6, 31, 36.3, 39.9, 6.5, 11.2, 12.8, 9.7, 15.9)
> TEMPERATURE
```

```
 [1] 36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7
[10] 15.9
```

```
> # a character vector
> WORDS<-c('Fish', 'Rock', 'Tree', "Git")
> WORDS
```

```
[1] "Fish" "Rock" "Tree" "Git"
```

```
> # a boolean vector
> BOOL<-c(TRUE, TRUE, FALSE, TRUE)
> BOOL
```

Data types

VECTORS

Regular sequences

```
> rep(4,5)
```

```
[1] 4 4 4 4 4
```

```
> rep('Fish',5)
```

```
[1] "Fish" "Fish" "Fish" "Fish" "Fish"
```

Data types

VECTORS

paste()

```
> QUADRATS <- c("Q1","Q2","Q3","Q4","Q5","Q6","Q7","Q8","Q9","Q10")
> QUADRATS
```

```
[1] "Q1"  "Q2"  "Q3"  "Q4"  "Q5"  "Q6"  "Q7"
[8] "Q8"  "Q9"  "Q10"
```

Data types

VECTORS

paste()

```
> QUADRATS <- c("Q1","Q2","Q3","Q4","Q5","Q6","Q7","Q8","Q9","Q10")
> QUADRATS
```

```
 [1] "Q1"  "Q2"  "Q3"  "Q4"  "Q5"  "Q6"  "Q7"
 [8] "Q8"  "Q9"  "Q10"
```

```
> QUADRATS <- paste("Q",1:10,sep="")
> QUADRATS
```

```
 [1] "Q1"  "Q2"  "Q3"  "Q4"  "Q5"  "Q6"  "Q7"
 [8] "Q8"  "Q9"  "Q10"
```

Data types

VECTORS

attributes

```
> TEMPERATURE
```

```
 [1] 36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7
[10] 15.9
```

```
> names(TEMPERATURE)
```

```
NULL
```

Data types

VECTORS

attributes

```
> TEMPERATURE
```

```
 [1] 36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7
[10] 15.9
```

```
> names(TEMPERATURE)
```

```
NULL
```

```
> names(TEMPERATURE) <- QUADRATS
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

Data types

attributes

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> scale(TEMPERATURE)
```

```
           [,1]
Q1    1.0151616
Q2    0.5889487
Q3    0.6199460
Q4    1.0306603
Q5    1.3096360
Q6   -1.2786387
Q7   -0.9144204
Q8   -0.7904312
Q9   -1.0306603
Q10  -0.5502021
attr(,"scaled:center")
[1] 23
attr(,"scaled:scale")
[1] 12.90435
```

Data types

VECTORS

```
> SITE <- paste(rep(LETTERS[1:5], each = 2), 1:2, sep = "")
> SITE
```

```
 [1] "A1" "A2" "B1" "B2" "C1" "C2" "D1" "D2" "E1"
[10] "E2"
```

Your turn

Calculate the area of circles with the following radii:

- circle A=10cm
- circle B=12.3cm
- circle C=25.6cm and store the results in a vector with item names reflecting the circle names

Your turn

```
> RADIUS <- c(10,12.3,25.6)
> names(RADIUS) <- c('circle A', 'circle B', 'circle C')
> #OR
> names(RADIUS) <- paste('circle',LETTERS[1:3])
> #OR even better
> names(RADIUS) <- paste('circle',LETTERS[1:length(RADIUS)])
> AREA <- pi*RADIUS^2
> AREA
```

```
circle A  circle B  circle C
314.1593  475.2916 2058.8742
```

Data types

VECTORS

Factors

```
> SHADE <- rep(c("no","full"),each=5)
> SHADE
```

```
[1] "no"   "no"   "no"   "no"   "no"   "full"
[7] "full" "full" "full" "full"
```

Data types

VECTORS

Factors

```
> SHADE <- rep(c("no","full"),each=5)
> SHADE
```

```
 [1] "no"   "no"   "no"   "no"   "no"   "full"
 [7] "full" "full" "full" "full"
```

```
> SHADE <- factor(SHADE)
> SHADE
```

```
 [1] no   no   no   no   no   full full full full
[10] full
Levels: full no
```

```
> str(SHADE)
```

Data types

VECTORS

Factors

```
> SHADE <- rep(c("no","full"),each=5)
> SHADE
```

```
 [1] "no"   "no"   "no"   "no"   "no"   "full"
 [7] "full" "full" "full" "full"
```

```
> SHADE <- factor(SHADE)
> SHADE
```

```
 [1] no   no   no   no   no   full full full full
[10] full
Levels: full no
```

```
> SHADE <- factor(SHADE, levels=c("no","full"))
> SHADE
```

Data types

VECTORS

Factors

```
> SHADE <- gl(2,5,10,c("no","full"))
> SHADE
```

```
 [1] no   no   no   no   no   full full full full
[10] full
Levels: no full
```

```
> str(SHADE)
```

```
 Factor w/ 2 levels "no","full": 1 1 1 1 1 2 2 2 2 2
```

Your turn

Create a categorical vector with: - three levels (A, B and C) - four replicates of each level

Your turn

```
> gl(3,4,12,lab=LETTERS[1:3])
```

```
 [1] A A A A B B B B C C C C
Levels: A B C
```

But what if you needed to arrange such that there was only two replicates in a row??

Your turn

```
> gl(3,2,12,lab=LETTERS[1:3])
```

```
 [1] A A B B C C A A B B C C
Levels: A B C
```

Data types

MATRICES

```
> matrix(SHADE,nrow=5)
```

```
     [,1] [,2]
[1,] "no" "full"
[2,] "no" "full"
[3,] "no" "full"
[4,] "no" "full"
[5,] "no" "full"
```

Data types

MATRICES

```
> X <- c(16.92,24.03,7.61,15.49,11.77)
> Y<- c(8.37,12.93,16.65,12.2,13.12)
> XY <- cbind(X,Y)
> XY
```

```
          X     Y
[1,] 16.92  8.37
[2,] 24.03 12.93
[3,]  7.61 16.65
[4,] 15.49 12.20
[5,] 11.77 13.12
```

Data types

MATRICES

```
> X <- c(16.92,24.03,7.61,15.49,11.77)
> Y<- c(8.37,12.93,16.65,12.2,13.12)
> XY <- cbind(X,Y)
> XY
```

```
          X     Y
[1,] 16.92  8.37
[2,] 24.03 12.93
[3,]  7.61 16.65
[4,] 15.49 12.20
[5,] 11.77 13.12
```

```
> rbind(X,Y)
```

```
   [,1]  [,2]  [,3]  [,4]  [,5]
X 16.92 24.03  7.61 15.49 11.77
Y  8.37 12.93 16.65 12.20 13.12
```

Data types

MATRICES

Beware of mixing data types

```
> cbind(TEMPERATURE,SITE)
```

```
    TEMPERATURE SITE
Q1  "36.1"      "A1"
Q2  "30.6"      "A2"
Q3  "31"        "B1"
Q4  "36.3"      "B2"
Q5  "39.9"      "C1"
Q6  "6.5"       "C2"
Q7  "11.2"      "D1"
Q8  "12.8"      "D2"
Q9  "9.7"       "E1"
Q10 "15.9"      "E2"
```

Data types

MATRICES

Beware of mixing data types

```
> cbind(TEMPERATURE,SHADE)
```

```
    TEMPERATURE SHADE
Q1         36.1     1
Q2         30.6     1
Q3         31.0     1
Q4         36.3     1
Q5         39.9     1
Q6          6.5     2
Q7         11.2     2
Q8         12.8     2
Q9          9.7     2
Q10        15.9     2
```

Data types

LISTS

```
> EXPERIMENT <- list(SITE=SITE,QUADRATS = QUADRATS,
+  COORDINATES = XY, SHADE = SHADE,
+  TEMPERATURE = TEMPERATURE)
> EXPERIMENT
```

```
$SITE
 [1] "A1" "A2" "B1" "B2" "C1" "C2" "D1" "D2" "E1"
[10] "E2"

$QUADRATS
 [1] "Q1"  "Q2"  "Q3"  "Q4"  "Q5"  "Q6"  "Q7"
 [8] "Q8"  "Q9"  "Q10"

$COORDINATES
          X     Y
[1,] 16.92  8.37
[2,] 24.03 12.93
[3,]  7.61 16.65
[4,] 15.49 12.20
[5,] 11.77 13.12
```

Data types

LISTS

Elements in lists (NOTE THE $)

```
> EXPERIMENT$TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

Data types

DATA FRAMES

Special type of list

```
> DATA <- data.frame(QUADRATS = QUADRATS,
+   SHADE = SHADE, TEMPERATURE = TEMPERATURE)
> DATA
```

```
    QUADRATS SHADE TEMPERATURE
Q1         Q1    no        36.1
Q2         Q2    no        30.6
Q3         Q3    no        31.0
Q4         Q4    no        36.3
Q5         Q5    no        39.9
Q6         Q6  full         6.5
Q7         Q7  full        11.2
Q8         Q8  full        12.8
Q9         Q9  full         9.7
Q10       Q10  full        15.9
```

Object manipulation

OBJECT INFORMATION

```
> class(TEMPERATURE)
```

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

```
> mode(TEMPERATURE)
```

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

```
> class(SHADE)
```

```
[1] "factor"
```

```
> mode(SHADE)
```

Object manipulation

OBJECT INFORMATION

```
> class(DATA)
```

```
[1] "data.frame"
```

```
> mode(DATA)
```

```
[1] "list"
```

```
> class(mean)
```

```
[1] "function"
```

```
> mode(mean)
```

Object manipulation

OBJECT INFORMATION

```
> class(TEMPERATURE)
```

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

```
> class(SHADE)
```

```
[1] "factor"
```

```
> class(DATA)
```

```
[1] "data.frame"
```

```
> class(mean)
```

Object manipulation

OBJECT CONVERSION

```
> as.character(TEMPERATURE)
```

```
[1] "36.1" "30.6" "31"   "36.3" "39.9" "6.5"
[7] "11.2" "12.8" "9.7"  "15.9"
```

```
> as.matrix(DATA)
```

```
    QUADRATS SHADE  TEMPERATURE
Q1  "Q1"     "no"   "36.1"
Q2  "Q2"     "no"   "30.6"
Q3  "Q3"     "no"   "31.0"
Q4  "Q4"     "no"   "36.3"
Q5  "Q5"     "no"   "39.9"
Q6  "Q6"     "full" " 6.5"
Q7  "Q7"     "full" "11.2"
Q8  "Q8"     "full" "12.8"
Q9  "Q9"     "full" " 9.7"
Q10 "Q10"    "full" "15.9"
```

Object manipulation

OBJECT ATTRIBUTES

```
> attributes(XY)
```

```
$dim
[1] 5 2

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "X" "Y"
```

Object manipulation

OBJECT ATTRIBUTES

```
> attributes(XY)
```

```
$dim
[1] 5 2

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "X" "Y"
```

```
> dim(XY)
```

```
[1] 5 2
```

Object manipulation

OBJECT ATTRIBUTES

```
> attr(XY, "dim")
```

```
[1] 5 2
```

```
> attr(XY, "description") <- "coordinates of quadrats"
> XY
```

```
         X     Y
[1,] 16.92  8.37
[2,] 24.03 12.93
[3,]  7.61 16.65
[4,] 15.49 12.20
[5,] 11.77 13.12
attr(,"description")
[1] "coordinates of quadrats"
```

Object indexing

SUBSET A VECTOR

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

1. Vector of positive numbers

```
> TEMPERATURE[2]
```

```
  Q2
30.6
```

```
> TEMPERATURE[2:5]
```

Object indexing

SUBSET A VECTOR

```
> TEMPERATURE
```

```
 Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

2. Vector of negative numbers

```
> TEMPERATURE[-2]
```

```
 Q1   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> TEMPERATURE[-2:-5]
```

Object indexing

SUBSET A VECTOR

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

3. Vector of character strings

```
> TEMPERATURE["Q1"]
```

```
  Q1
36.1
```

```
> TEMPERATURE[c("Q1","Q4")]
```

Object indexing

SUBSET A VECTOR

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

4. Vector of logical values

```
> TEMPERATURE[TEMPERATURE < 15]
```

```
  Q6   Q7   Q8   Q9
 6.5 11.2 12.8  9.7
```

```
> TEMPERATURE[SHADE == "no"]
```

Your turn

Subset the SITES vector:

- sites that have temperatures between 14 and 31
- sites that have temperature < 10 or no shade
- full shade sites with temperatures > 14 or no shade sites with temperatures greater than 38

Your turn

```
    SITE TEMPERATURE SHADE
Q1    A1        36.1    no
Q2    A2        30.6    no
Q3    B1        31.0    no
Q4    B2        36.3    no
Q5    C1        39.9    no
Q6    C2         6.5  full
Q7    D1        11.2  full
Q8    D2        12.8  full
Q9    E1         9.7  full
Q10   E2        15.9  full
```

```
> SITE[TEMPERATURE >= 14 & TEMPERATURE <=31]
```

```
[1] "A2" "B1" "E2"
```

```
> SITE[TEMPERATURE < 10 | SHADE=='no']
```

```
[1] "A1" "A2" "B1" "B2" "C1" "C2" "E1"
```

```
> SITE[TEMPERATURE > 14 & SHADE=='full' | (TEMPERATURE > 38 & SHADE=='no')]
```

```
[1] "C1" "E2"
```

Your turn

Subset the SITES vector:

	SITE	TEMPERATURE	SHADE
Q1	A1	36.1	no
Q2	A2	30.6	no
Q3	B1	31.0	no
Q4	B2	36.3	no
Q5	C1	39.9	no
Q6	C2	6.5	full
Q7	D1	11.2	full
Q8	D2	12.8	full
Q9	E1	9.7	full
Q10	E2	15.9	full

- sites with temperature > 35 or full shade and temperature < 39

Your turn

```
    SITE TEMPERATURE SHADE
Q1    A1        36.1    no
Q2    A2        30.6    no
Q3    B1        31.0    no
Q4    B2        36.3    no
Q5    C1        39.9    no
Q6    C2         6.5  full
Q7    D1        11.2  full
Q8    D2        12.8  full
Q9    E1         9.7  full
Q10   E2        15.9  full
```

```
> SITE[TEMPERATURE > 35 | SHADE=='full' & TEMPERATURE < 39]
```

```
[1] "A1" "B2" "C1" "C2" "D1" "D2" "E1" "E2"
```

```
> SITE[(TEMPERATURE > 35 | SHADE=='full') & TEMPERATURE < 39]
```

```
[1] "A1" "B2" "C2" "D1" "D2" "E1" "E2"
```

Basic data formatting

ROUNDING

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> #round up to nearest integer
> ceiling(TEMPERATURE)
```

```
Q1  Q2  Q3  Q4  Q5  Q6  Q7  Q8  Q9 Q10
37  31  31  37  40   7  12  13  10  16
```

```
> #round down to nearest integer
> floor(TEMPERATURE)
```

```
Q1  Q2  Q3  Q4  Q5  Q6  Q7  Q8  Q9 Q10
36  30  31  36  39   6  11  12   9  15
```

Basic data formatting

ROUNDING

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> #round to specified decimal places
> round(TEMPERATURE)
```

```
Q1  Q2  Q3  Q4  Q5  Q6  Q7  Q8  Q9 Q10
36  31  31  36  40   6  11  13  10  16
```

```
> round(TEMPERATURE/2.2,digits=2)
```

```
   Q1    Q2    Q3    Q4    Q5    Q6    Q7    Q8
16.41 13.91 14.09 16.50 18.14  2.95  5.09  5.82
   Q9   Q10
```

Formatting strings

Function	Description
`paste()`	Concatenate vectors after converting into characters
`format()`	Adjust decimal places, justification, padding and width of string and whether to use scientific notation
`formatC()`	A version of `format()` that is compliant with C style formatting
`sprintf()`	A wrapper for the C style formatting function of the same name, provides even greater flexibility (and complexity)

Formatting strings

PASTE

```
> paste("Quadrat", 1:3, sep=":")
```

```
[1] "Quadrat:1" "Quadrat:2" "Quadrat:3"
```

```
> #create a joint label for Site and Quadrat combinations
> paste(SITE,QUADRATS,sep=":")
```

```
 [1] "A1:Q1"  "A2:Q2"  "B1:Q3"  "B2:Q4"  "C1:Q5"
 [6] "C2:Q6"  "D1:Q7"  "D2:Q8"  "E1:Q9"  "E2:Q10"
```

```
> #create a string of comma separated items
> paste('Quad',1:4,sep='',collapse=', ')
```

```
[1] "Quad1, Quad2, Quad3, Quad4"
```

Formatting strings

FORMAT

```
> TEMPERATURE
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> #create equal width strings (add padding on the left)
> format(TEMPERATURE)
```

```
    Q1     Q2     Q3     Q4     Q5     Q6     Q7
"36.1" "30.6" "31.0" "36.3" "39.9" " 6.5" "11.2"
    Q8     Q9    Q10
"12.8" " 9.7" "15.9"
```

Formatting strings

FORMAT

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> #create equal width strings with a minimum of 2 decimal places
> format(TEMPERATURE, nsmall=2)
```

```
     Q1       Q2       Q3       Q4       Q5       Q6
"36.10" "30.60" "31.00" "36.30" "39.90" " 6.50"
     Q7       Q8       Q9      Q10
"11.20" "12.80" " 9.70" "15.90"
```

```
> #contrast to the following
> #truncate to 2 decimal placed before rounding
> format(TEMPERATURE, digits=2)
```

```
   Q1     Q2     Q3     Q4     Q5     Q6     Q7
```

Formatting strings

FORMAT

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
36.1 30.6 31.0 36.3 39.9  6.5 11.2 12.8  9.7 15.9
```

```
> #truncate to 1 decimal place before rounding
> format(TEMPERATURE, digits=1)
```

```
  Q1   Q2   Q3   Q4   Q5   Q6   Q7   Q8   Q9  Q10
"36" "31" "31" "36" "40" " 6" "11" "13" "10" "16"
```

```
> #scientific notation
> format(TEMPERATURE,scientific=TRUE)
```

```
        Q1         Q2         Q3         Q4
"3.61e+01" "3.06e+01" "3.10e+01" "3.63e+01"
        Q5         Q6         Q7         Q8
"3.99e+01" "6.50e+00" "1.12e+01" "1.28e+01"
```

Formatting strings

FORMAT

Works for data frames as well

```
> # character and factor columns left justified
> # numerical columns minimum of two decimal places
> format(DATA, justify="left", nsmall=2)
```

```
    QUADRATS SHADE TEMPERATURE
Q1        Q1   no         36.10
Q2        Q2   no         30.60
Q3        Q3   no         31.00
Q4        Q4   no         36.30
Q5        Q5   no         39.90
Q6        Q6   full        6.50
Q7        Q7   full       11.20
Q8        Q8   full       12.80
Q9        Q9   full        9.70
Q10      Q10   full       15.90
```

Formatting strings

FORMATC

```
> #generate a sequence of numbers
> (NUM <- exp(seq(0,10,length=5)))
```

```
[1]     1.00000    12.18249   148.41316
[4]  1808.04241 22026.46579
```

```
> format(NUM,big.mark = ",",digits=2)
```

```
[1] "     1" "    12" "   148" " 1,808" "22,026"
```

Formatting strings

FORMATC

```
> #generate a sequence of numbers
> (NUM <- exp(seq(-1,10,length=5)))
```

```
[1] 3.678794e-01 5.754603e+00 9.001713e+01
[4] 1.408105e+03 2.202647e+04
```

```
> format(NUM,big.mark = ",",digits=2)
```

```
[1] "3.7e-01" "5.8e+00" "9.0e+01" "1.4e+03"
[5] "2.2e+04"
```

```
> formatC(NUM, format='f',big.mark = ",",digits=2)
```

```
[1] "0.37"      "5.75"      "90.02"
[4] "1,408.10"  "22,026.47"
```

Formatting strings

FORMATC

- 'd' for integers
- 'f' for reals in the standard xxx.xxx format
- 'e', 'E' for reals in the scientific (n.ddde+nn) format
- 'g', 'G' for reals in the scientific (n.ddde+nn) format when it saves space to do so

Formatting strings

FORMATC

- 's' for strings
- 'd' for integers