Introduction to the R programming language

Dan Sewell
Department of Biostatistics, University of Iowa

Introduction to R

What is R?

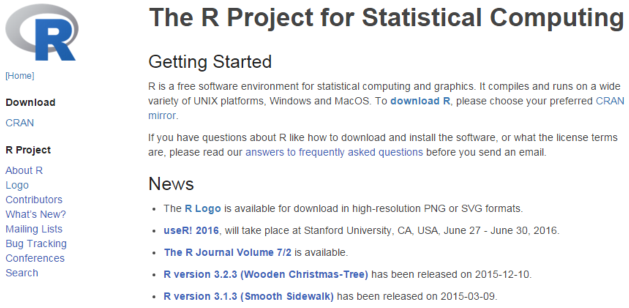
From the R Core Team:
“R is a system for statistical computation and graphics.”

- high level programming language
- run-time environment with graphics
- debugger
- data management tools
- sophisticated analytic tools
- thousands of packages are available
- open source- you can contribute!

Getting Started

Download R from Comprehensive R Archive Network (CRAN)

`www.r-project.com`



The R Project for Statistical Computing

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.

News

- The **R Logo** is available for download in high-resolution PNG or SVG formats.
- **useR! 2016**, will take place at Stanford University, CA, USA, June 27 - June 30, 2016.
- **The R Journal Volume 7/2** is available.
- **R version 3.2.3 (Wooden Christmas-Tree)** has been released on 2015-12-10.
- **R version 3.1.3 (Smooth Sidewalk)** has been released on 2015-03-09.

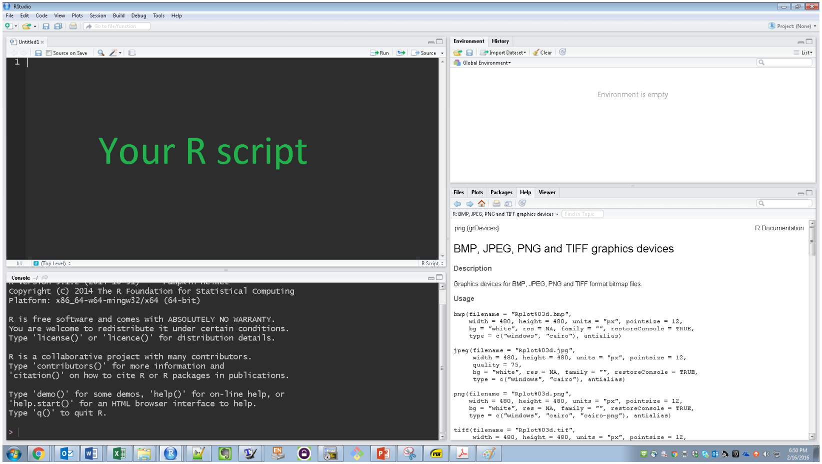
RStudio

RStudio IDE (Integrated Development Environment)

`www.rstudio.com`

- Nicer GUI
- Lots of features
- Integrated version control
- integration with Sweave/knitr/markdown/Shiny

RStudio



```
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

>
```

png {grDevices} R Documentation

BMP, JPEG, PNG and TIFF graphics devices

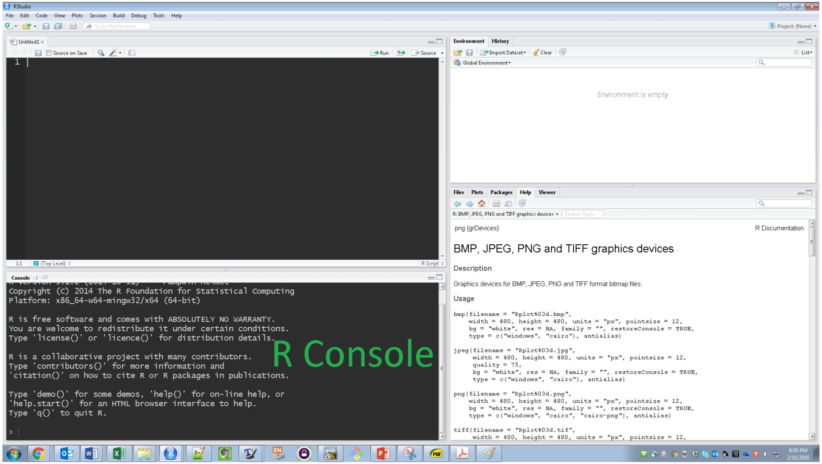
Description

Graphics devices for BMP, JPEG, PNG and TIFF format bitmap files.

Usage

```
bmp(filename = "Rplot%03d.bmp",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo"), antialias)

jpeg(filename = "Rplot%03d.jpg",
     width = 480, height = 480, units = "px", pointsize = 12,
     quality = 75,
     bg = "white", res = NA, family = "", restoreConsole = TRUE,
     type = c("windows", "cairo"), antialias)

png(filename = "Rplot%03d.png",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo", "cairo-png"), antialias)

tiff(filename = "Rplot%03d.tif",
     width = 480, height = 480, units = "px", pointsize = 12,
```

RStudio



```
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

>
```

png {grDevices} R Documentation

BMP, JPEG, PNG and TIFF graphics devices

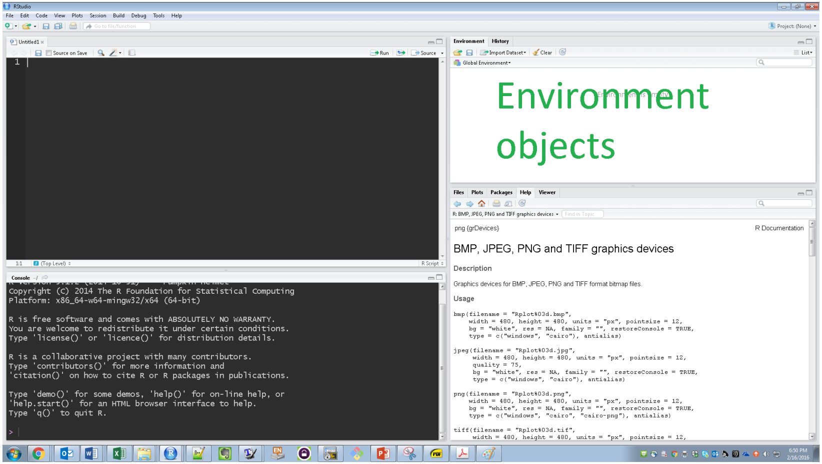
Description

Graphics devices for BMP, JPEG, PNG and TIFF format bitmap files.

Usage

```
bmp(filename = "Rplot%03d.bmp",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo"), antialias)

jpeg(filename = "Rplot%03d.jpg",
     width = 480, height = 480, units = "px", pointsize = 12,
     quality = 75,
     bg = "white", res = NA, family = "", restoreConsole = TRUE,
     type = c("windows", "cairo"), antialias)

png(filename = "Rplot%03d.png",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo", "cairo-png"), antialias)

tiff(filename = "Rplot%03d.tif",
     width = 480, height = 480, units = "px", pointsize = 12,
```

RStudio



```
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

>
```

png {grDevices} R Documentation

BMP, JPEG, PNG and TIFF graphics devices

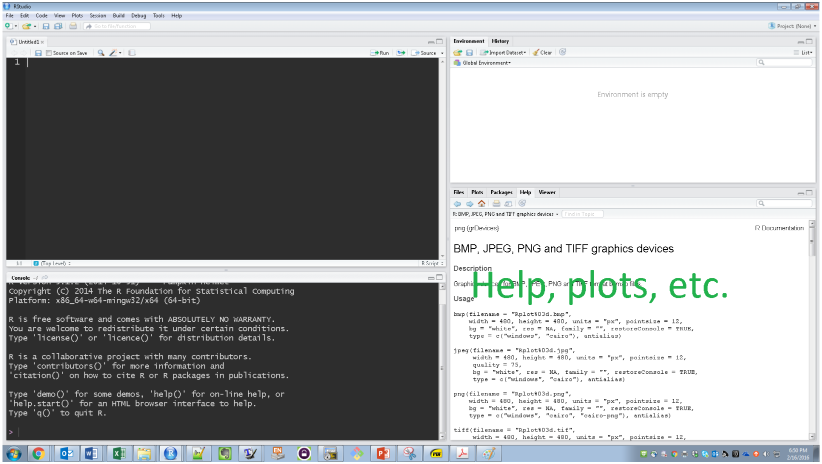
Description

Graphics devices for BMP, JPEG, PNG and TIFF format bitmap files.

Usage

```
bmp(filename = "Rplot%03d.bmp",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo"), antialias)

jpeg(filename = "Rplot%03d.jpg",
     width = 480, height = 480, units = "px", pointsize = 12,
     quality = 75,
     bg = "white", res = NA, family = "", restoreConsole = TRUE,
     type = c("windows", "cairo"), antialias)

png(filename = "Rplot%03d.png",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo", "cairo-png"), antialias)

tiff(filename = "Rplot%03d.tif",
     width = 480, height = 480, units = "px", pointsize = 12,
```

RStudio



```
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

>
```

png {grDevices} R Documentation

BMP, JPEG, PNG and TIFF graphics devices

Description

Graphics devices for BMP, JPEG, PNG and TIFF format bitmap files.

Usage

```
bmp(filename = "Rplot%03d.bmp",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo"), antialias)

jpeg(filename = "Rplot%03d.jpg",
     width = 480, height = 480, units = "px", pointsize = 12,
     quality = 75,
     bg = "white", res = NA, family = "", restoreConsole = TRUE,
     type = c("windows", "cairo"), antialias)

png(filename = "Rplot%03d.png",
    width = 480, height = 480, units = "px", pointsize = 12,
    bg = "white", res = NA, family = "", restoreConsole = TRUE,
    type = c("windows", "cairo", "cairo-png"), antialias)

tiff(filename = "Rplot%03d.tif",
     width = 480, height = 480, units = "px", pointsize = 12,
```

Scripts

R has a run-time environment, but we can still save code in scripts (.R extension), or source them:

```
> setwd("~/testDirectory/")
> #use getwd() to see working directory
> source("testScript.R")
```

Basics

- ; or a newline separate commands

Basics

- ; or a newline separate commands
- R code is *case sensitive*

Basics

- ; or a newline separate commands
- R code is *case sensitive*
- # comments all characters until the next newline

Basics

- ; or a newline separate commands
- R code is *case sensitive*
- # comments all characters until the next newline
- To get help, use ?keyword, or ??searchword, or F1, or TAB

Workspaces

The R workspace is the collection of all R objects. To view. . .

```
> ls()
```

```
[1] "DataSet" "X"         "Y"
```

To save an object:

```
> save(X,file="test.RData")
```

To save your entire workspace:

```
> save.image("test.RData")
```

To load an R object or R workspace:

```
> load("test.RData")
```

Where are you saving to?

```
> getwd()
> setwd("~/newdir/")
```

Packages

The biggest strength of R: PACKAGES
First you must install them (only run once):

```
> #install.packages("lme4")
```

Each R session, load the necessary packages:

```
> library("lme4")
> #require("lme4")
```

Packages

The biggest difference between `library` and `require` is which throws an error:

```
> library("asdf");print("script is killed")
Error in library("asdf") : there is no pack-
age called 'asdf'
```

> require("asdf");print("code is still running!")

```
[1] "code is still running!"
```

R script

The accompanying R script for this seminar may be found at
`http://myweb.uiowa.edu/dksewell/RSeminarScript.R`

Data Structures

Classes

- integer

```
> 5L
[1] 5
> as.integer(5.0)
[1] 5
```

Classes

- integer

```
> 5L
[1] 5
> as.integer(5.0)
[1] 5
```

- numeric

```
> 5.25
[1] 5.25
> pi
[1] 3.141593
> as.numeric(1L)
[1] 1
```

Classes

- character

```
> "hello world"
[1] "hello world"
> as.character(1.0)
[1] "1"
```

Classes

- character

```
> "hello world"
[1] "hello world"
> as.character(1.0)
[1] "1"
```

- logical/boolean

```
> TRUE
[1] TRUE
> 1 > 2
[1] FALSE
> as.logical(1L)
[1] TRUE
```

Classes

- Missing value

```
> NA
[1] NA
> is.na(NA)
[1] TRUE
```

Classes

- Missing value

```
> NA
[1] NA
> is.na(NA)
[1] TRUE
```

- NULL

```
> NULL
NULL
> is.null(NULL)
[1] TRUE
```

Scalars

One can create a scalar object:

```
> x1= 1/3
> x2 <- 1/6
> 1/2 -> x3
> class(x1)

[1] "numeric"
```

Scalars

One can create a scalar object:

```
> x1= 1/3
> x2 <- 1/6
> 1/2 -> x3
> class(x1)

[1] "numeric"
```

The R console can also act as a calculator:

```
> (x1+x2-2*x3)*0.1

[1] -0.05
```

Vectors

One can create vectors:

```
> v1=c(1,2,4,6)
> v2=1:4
> ( v3=0:3*5+1 )

[1]  1  6 11 16
```

Vectors

One can create vectors:

```
> v1=c(1,2,4,6)
> v2=1:4
> ( v3=0:3*5+1 )

[1]  1  6 11 16
```

These may also be named

```
> ( v1 = c(x1=1,x2=2,x3=4,x4=6) )

x1 x2 x3 x4
 1  2  4  6

> names(v2) = c("x1","x2","x3","x4"); print(v2)

x1 x2 x3 x4
 1  2  3  4

> names(v3) = paste("x",1:4,sep=""); print(v3)

x1 x2 x3 x4
 1  6 11 16
```

Vectors

You can retrieve elements or subsets:

```
> v1[4]

x4
 6

> v1["x4"]

x4
 6

> v1[3:4]

x3 x4
 4  6

> v1[c(1,2,4)]

x1 x2 x4
 1  2  6

> v1[-c(1,2,4)]

x3
 4
```

Vectors

Vectors also may be of varying classes:

```
> ( v4 = c("Hello","World") )
[1] "Hello" "World"
> class(v4)
[1] "character"
> ( v5 = rep(TRUE,4) )
[1] TRUE TRUE TRUE TRUE
> class(v5)
[1] "logical"
```

Factors

One may create nominal variables via factors:

```
> ( fac1= factor(rep(1:3,3),labels=paste("Drug",1:3,sep="")
```

```
[1] Drug1 Drug2 Drug3 Drug1 Drug2 Drug3 Drug1 Drug2 Drug3
Levels: Drug1 Drug2 Drug3
```

One may rename the levels as well:

```
> levels(fac1) = paste("Drug",3:1,sep="")
> fac1
```

```
[1] Drug3 Drug2 Drug1 Drug3 Drug2 Drug1 Drug3 Drug2 Drug1
Levels: Drug3 Drug2 Drug1
```

Matrices

One may create matrices (column major by default):

```
> ( mat1 = matrix(0L,2,3) )

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0

> ( mat2 = matrix(1:6,2,3) )

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> ( mat3 = matrix(1:6,2,3,byrow=TRUE) )

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Matrices

And name the rows and columns:

```
> rownames(mat2)=paste("row",1:2)
> colnames(mat2)=paste("column",1:3)
> mat2
```

```
      column 1 column 2 column 3
row 1        1        3        5
row 2        2        4        6
```

Matrices

Again we can extract elements or subsets:

```
> mat2[2,3]

[1] 6

> mat2["row 2","column 3"]

[1] 6

> mat2[6]

[1] 6

> mat2[,c(1,3)]

      column 1 column 3
row 1        1        5
row 2        2        6
```

Matrices

There are lots of types of matrices and packages that deal with them, e.g., “Matrix”

```
> require("Matrix")
> mat4 = matrix(0,1000,50)
> mat5 = Matrix(0,1000,50,sparse=TRUE)
> class(mat4)

[1] "matrix"

> object.size(mat4)

400200 bytes

> class(mat5)

[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"

> object.size(mat5)

1624 bytes
```

Matrices

Other special packages for sparse matrices:

- Matrix
- glmnet
- SparseM
- slam
- spam
- igraph/network
- irlba (Fast and memory efficient methods for truncated singular and eigenvalue decompositions and principal component analysis of large sparse or dense matrices.)

Matrices

Special packages for huge general matrices:

- bigmemory (Manage Massive Matrices with Shared Memory and Memory-Mapped Files)
- bigalgebra (BLAS routines for native R matrices and big.matrix objects)
- bigtabulate (Table, Apply, and Split Functionality for Matrix and 'big.matrix' Objects)
- biganalytics (Utilities for 'big.matrix' Objects from Package 'bigmemory')
- bigpca (PCA, Transpose and Multicore Functionality for 'big.matrix' Objects)

Matrices

Using bigmemory:

```
> require("bigmatrix");require("bigalgebra")
> mat6 = matrix(1.0,5000,5000)
> mat7 = as.big.matrix(mat6)
> object.size(mat6);object.size(mat7)
> #200MB vs. 0.6KB
> system.time(mat6%*%mat6)
> #76.44sec
> system.time(mat7%*%mat7)
> #82.03sec
```

Arrays/Tensors

One may create multidimensional arrays:

```
> ( arr1 = array(1:12,c(2,3,2)) )

, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

Arrays/Tensors

And extract in exactly the same way as before:

```
> arr1[2,1,1]
[1] 2
> arr1[1,,2]
[1]  7  9 11
```

Data frames

Data are most often in data frames:

```
> ( df1 = data.frame(id=1:6,
+             gender=factor(rep(c("M","F"),3)),
+             treatment1=factor(rep(LETTERS[1:3],2)),
+             treatment2=factor(rep(LETTERS[1:2],each=3)),
+             response=rnorm(6)) )
  id gender treatment1 treatment2   response
1  1      M          A          A -0.6264538
2  2      F          B          A  0.1836433
3  3      M          C          A -0.8356286
4  4      F          A          B  1.5952808
5  5      M          B          B  0.3295078
6  6      F          C          B -0.8204684
```

Note: The columns may be of varying formats!

Data frames

And extraction:

```
> df1$treatment1

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

> df1[,"treatment1"]

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

> df1[,3]

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

Lists

One may create lists of *anything*:

```
> ls1 = list()
> ls1[[1]] = "Hello World"
> ls1[[2]] = pi
> ls1[[3]] = matrix(1:4,2,2)
> print(ls1)

[[1]]
[1] "Hello World"

[[2]]
[1] 3.141593

[[3]]
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Functions

Example: Dose-response Exponential model

```
> DRfun = function(dose){
+   return( 1-exp(-2.18E-04*dose) )
+ }
> DRfun(3.18E+03)

[1] 0.5000464

> DRfun(1+03)

[1] 0.0008716199
```

Functions

Example: Dose-response Exponential model

```
> DRfun = function(dose,K){
+   return( 1-exp(-K*dose) )
+ }
> DRfun(1e3,2e-4)

[1] 0.1812692
```

Special extraction commands

Perhaps the most useful is `which()`

```
> df1[which(df1$response<0),]
```

```
  id gender treatment1 treatment2   response
1  1      M          A          A -0.6264538
3  3      M          C          A -0.8356286
6  6      F          C          B -0.8204684
```

Special extraction commands

Use `with()` to run commands using a particular data set:

```
> with(df1,response[which(treatment1=="C")])
[1] -0.8356286 -0.8204684
```

Special extraction commands

Use `with()` to run commands using a particular data set:

```
> with(df1,response[which(treatment1=="C")])

[1] -0.8356286 -0.8204684
```

Use `by()` to perform functions according to a grouping variable:

```
> with(df1,by(response,gender,mean))

gender: F
[1] 0.3194852
------------------------------------------------------------

gender: M
[1] -0.3775249
```

Programming Structures

for loops

Keywords: `for` and `in`

```
> for(i in 1:4){
+   print(i^2)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
```

for loops

More generally:

```
> x = c(1,2,4,8,16)
> for(i in x){
+   print(log2(i))
+ }
```

```
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
```

while loops

Keywords: `while`

```
> count=0
> while(count<5){
+   print(count)
+   count = count + 1
+ }
```

```
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
```

while loops

```
> set.seed(1)
> x=0
> while(x<0.5){
+    x = rnorm(1)
+    print(x)
+ }

[1] -0.6264538
[1] 0.1836433
[1] -0.8356286
[1] 1.595281
```

if/else statements

Keywords: `if`, `else`, `ifelse`

```
> set.seed(1)
> uu = runif(1)
> if(uu < 0.5){
+   print("heads")
+ }else{
+   print("tails")
+ }
[1] "heads"
```

if/else statements

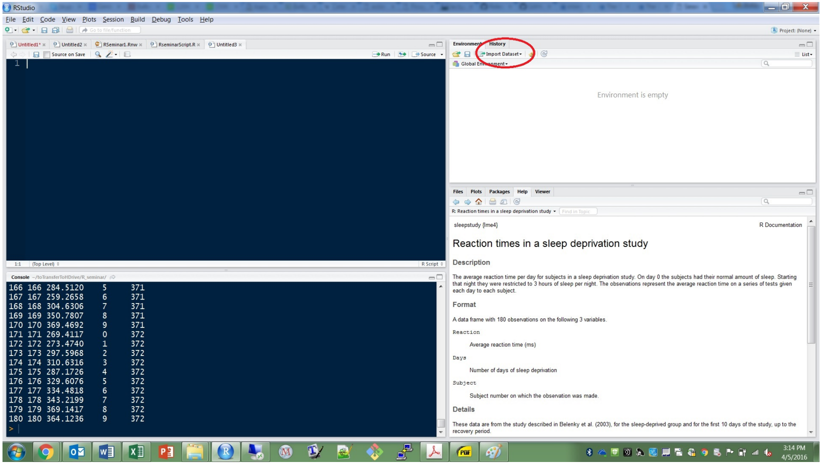
Keywords: `if`, `else`, `ifelse`

```
> set.seed(1)
> uu = runif(1)
> result = ifelse(uu < 0.5, "heads", "tails")
> print(result)

[1] "heads"
```

Reading/Writing Data

Reading in Data

To read data into R:


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
Untitled1* Untitled2 RSeminar1.Rnw RseminarScript.R Untitled3
Source on Save
Run
Source
1:1 (Top Level)
R Script
Console ~/toTransferToHDrive/R_seminar/
Environment History
Import Dataset
Global Environment
List
Environment is empty
Files Plots Packages Help Viewer
R: Reaction times in a sleep deprivation study
Find in Topic
3:14 PM
4/5/2016

```
166 166 284.5120    5     371
167 167 259.2658    6     371
168 168 304.6306    7     371
169 169 350.7807    8     371
170 170 369.4692    9     371
171 171 269.4117    0     372
172 172 273.4740    1     372
173 173 297.5968    2     372
174 174 310.6316    3     372
175 175 287.1726    4     372
176 176 329.6076    5     372
177 177 334.4818    6     372
178 178 343.2199    7     372
179 179 369.1417    8     372
180 180 364.1236    9     372
>
```

sleepstudy {lme4} R Documentation

Reaction times in a sleep deprivation study

Description

The average reaction time per day for subjects in a sleep deprivation study. On day 0 the subjects had their normal amount of sleep. Starting that night they were restricted to 3 hours of sleep per night. The observations represent the average reaction time on a series of tests given each day to each subject.

Format

A data frame with 180 observations on the following 3 variables.

`Reaction`
Average reaction time (ms)

`Days`
Number of days of sleep deprivation

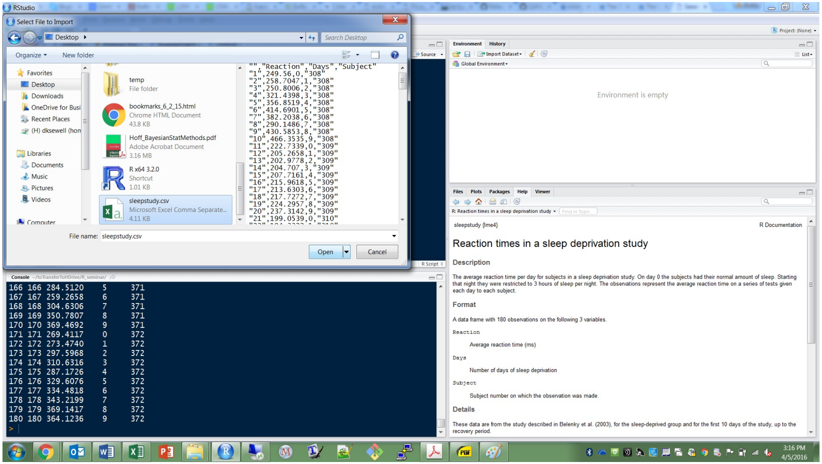
`Subject`
Subject number on which the observation was made.

Details

These data are from the study described in Belenky et al. (2003), for the sleep-deprived group and for the first 10 days of the study, up to the recovery period.

Reading in Data

To read data into R:



```
"","Reaction","Days","Subject"
"1",249.56,0,"308"
"2",258.7047,1,"308"
"3",250.8006,2,"308"
"4",321.4398,3,"308"
"5",356.8519,4,"308"
"6",414.6901,5,"308"
"7",382.2038,6,"308"
"8",290.1486,7,"308"
"9",430.5853,8,"308"
"10",466.3535,9,"308"
"11",222.7339,0,"309"
"12",205.2658,1,"309"
"13",202.9778,2,"309"
"14",204.707,3,"309"
"15",207.7161,4,"309"
"16",215.9618,5,"309"
"17",213.6303,6,"309"
"18",217.7272,7,"309"
"19",224.2957,8,"309"
"20",237.3142,9,"309"
"21",199.0539,0,"310"
```

```
166 166 284.5120     5     371
167 167 259.2658     6     371
168 168 304.6306     7     371
169 169 350.7807     8     371
170 170 369.4692     9     371
171 171 269.4117     0     372
172 172 273.4740     1     372
173 173 297.5968     2     372
174 174 310.6316     3     372
175 175 287.1726     4     372
176 176 329.6076     5     372
177 177 334.4818     6     372
178 178 343.2199     7     372
179 179 369.1417     8     372
180 180 364.1236     9     372
>
```

sleepstudy {lme4} R Documentation

Reaction times in a sleep deprivation study

Description

The average reaction time per day for subjects in a sleep deprivation study. On day 0 the subjects had their normal amount of sleep. Starting that night they were restricted to 3 hours of sleep per night. The observations represent the average reaction time on a series of tests given each day to each subject.

Format

A data frame with 180 observations on the following 3 variables.

`Reaction`

Average reaction time (ms)

`Days`

Number of days of sleep deprivation

`Subject`

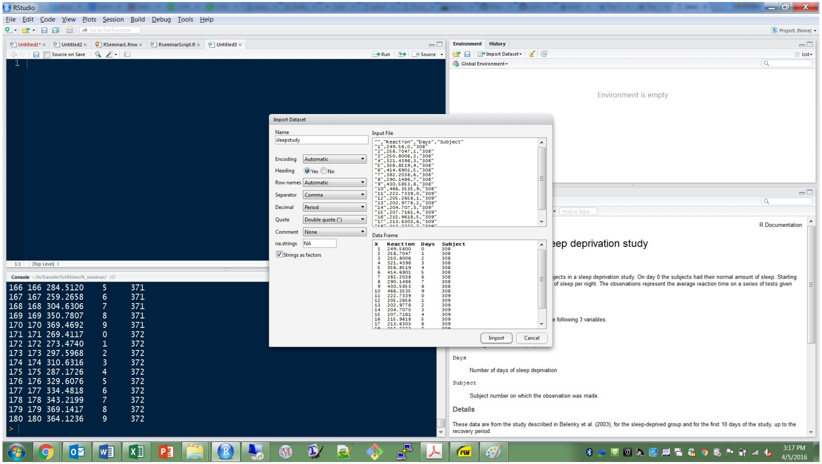
Subject number on which the observation was made.

Details

These data are from the study described in Belenky et al. (2003), for the sleep-deprived group and for the first 10 days of the study, up to the recovery period.

Reading in Data

To read data into R:



Import Dataset

Name: sleepstudy

Encoding: Automatic
Heading: [✓ Yes] No
Row names: Automatic
Separator: Comma
Decimal: Period
Quote: Double quote (")
Comment: None
na.strings: NA
[✓] Strings as factors

Input File

```
"","Reaction","Days","Subject"
"1",249.56,0,"308"
"2",258.7047,1,"308"
"3",250.8006,2,"308"
"4",321.4398,3,"308"
"5",356.8519,4,"308"
"6",414.6901,5,"308"
"7",382.2038,6,"308"
"8",290.1486,7,"308"
"9",430.5853,8,"308"
"10",466.3535,9,"308"
"11",222.7339,0,"309"
"12",205.2658,1,"309"
"13",202.9778,2,"309"
"14",204.707,3,"309"
"15",207.7161,4,"309"
"16",215.9618,5,"309"
"17",213.6303,6,"309"
```

Data Frame

```
X   Reaction  Days  Subject
 1  249.5600  0     308
 2  258.7047  1     308
 3  250.8006  2     308
 4  321.4398  3     308
 5  356.8519  4     308
 6  414.6901  5     308
 7  382.2038  6     308
 8  290.1486  7     308
 9  430.5853  8     308
10  466.3535  9     308
11  222.7339  0     309
12  205.2658  1     309
13  202.9778  2     309
14  204.7070  3     309
15  207.7161  4     309
16  215.9618  5     309
17  213.6303  6     309
```

Console ~/toTransferToHDrive/R_seminar/

```
166 166 284.5120    5    371
167 167 259.2658    6    371
168 168 304.6306    7    371
169 169 350.7807    8    371
170 170 369.4692    9    371
171 171 269.4117    0    372
172 172 273.4740    1    372
173 173 297.5968    2    372
174 174 310.6316    3    372
175 175 287.1726    4    372
176 176 329.6076    5    372
177 177 334.4818    6    372
178 178 343.2199    7    372
179 179 369.1417    8    372
180 180 364.1236    9    372
>
```

R Documentation

eep deprivation study

jects in a sleep deprivation study. On day 0 the subjects had their normal amount of sleep. Starting of sleep per night. The observations represent the average reaction time on a series of tests given

e following 3 variables.

Days

Number of days of sleep deprivation

Subject

Subject number on which the observation was made.

Details

These data are from the study described in Belenky et al. (2003), for the sleep-deprived group and for the first 10 days of the study, up to the recovery period.

Reading in Data

```
> sleepstudy = read.csv("http://myweb.uiowa.edu/dksewell/sl
> test = read.table("http://myweb.uiowa.edu/dksewell/sleeps
+                         sep=",",header=TRUE)
> all.equal(sleepstudy,test)
```

Reading in Data

- `require("foreign")` can import Minitab, SAS, Stata, SPSS, etc.
- `require("xlsx")` can import excel files

Reading in Data

Common ways to view data:

```
> head(sleepstudy)
  Reaction Days Subject
1 249.5600    0     308
2 258.7047    1     308
3 250.8006    2     308
4 321.4398    3     308
5 356.8519    4     308
6 414.6901    5     308
```

Reading in Data

Common ways to view data:

```
> tail(sleepstudy)

    Reaction Days Subject
175 287.1726    4     372
176 329.6076    5     372
177 334.4818    6     372
178 343.2199    7     372
179 369.1417    8     372
180 364.1236    9     372
```

Reading in Data

Common ways to view data:

```
> View(sleepstudy)  #In RStudio only
```

Reading in Data

Common ways to view data:

```
> print(sleepstudy)
```

Reading in Data

Common ways to view data:

```
> colnames(sleepstudy)
[1] "Reaction" "Days"     "Subject"
```

Reading in Data

Common ways to view data:

```
> str(sleepstudy)
```

```
'data.frame':       180 obs. of  3 variables:
 $ Reaction: num  250 259 251 321 357 ...
 $ Days    : num  0 1 2 3 4 5 6 7 8 9 ...
 $ Subject : Factor w/ 18 levels "308","309","310",..: 1 1
```

Reading in Data

Common ways to view data:

```
> dim(sleepstudy)
[1] 180   3
```

Reading in Data

Try it:

```
> blowdown = alr3::blowdown
```

```
> ?blowdown
```

Reading in Data

```
> summary(blowdown)
       D                S                  y             SP
 Min.   : 5.00   Min.   :0.02175   Min.   :0.0000   BS
 1st Qu.: 9.00   1st Qu.:0.21856   1st Qu.:0.0000   BF
 Median :14.00   Median :0.39326   Median :0.0000   JP
 Mean   :15.91   Mean   :0.41156   Mean   :0.4594   PB
 3rd Qu.:21.00   3rd Qu.:0.58940   3rd Qu.:1.0000   A
 Max.   :85.00   Max.   :0.98327   Max.   :1.0000   C
                                                    (Other
```

Reading in Data

```
> summary(blowdown$D)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    9.00   14.00   15.91   21.00   85.00
```

Reading in Data

```
> with(blowdown,by(D,y,summary))
y: 0
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    7.00   11.00   13.09   16.00   85.00
------------------------------------------------------------
y: 1
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00   13.00   17.50   19.22   25.00   58.00
```

Reading in Data

```
> colMeans(blowdown[,-4])
         D          S          y 
15.9050736  0.4115597  0.4593562 
> #rowMeans()
```

Reading in Data

```
> apply(blowdown[,-4],2,mean)
         D          S          y
15.9050736  0.4115597  0.4593562
```

Reading in Data

```
> apply(blowdown[,-4],2,function(x) mean(abs(x-mean(x))))
        D         S         y
6.7392200 0.1965914 0.4966962
```

Reading in Data

```
> blowdownScaled = scale(blowdown[,-4])
> apply(blowdownScaled,2,
+       function(x) round(c(mean=mean(x),sd=sd(x)),10))
     D S y
mean 0 0 0
sd   1 1 1
```

Reading in Data

Reshaping the data:

```
> require("reshape2")
> ###Long form:
> head(sleepstudy)
```

```
  Reaction Days Subject
1 249.5600    0     308
2 258.7047    1     308
3 250.8006    2     308
4 321.4398    3     308
5 356.8519    4     308
6 414.6901    5     308
```

Reading in Data
Reshaping the data:

```
> ###Wide form:
> sleepWide = acast(sleepstudy,Subject~factor(Days),
+                     value.var="Reaction")
> dim(sleepWide)

[1] 18 10

> head(round(sleepWide,1))

        0     1     2     3     4     5     6     7     8     9
308 249.6 258.7 250.8 321.4 356.9 414.7 382.2 290.1 430.6 466.4
309 222.7 205.3 203.0 204.7 207.7 216.0 213.6 217.7 224.3 237.3
310 199.1 194.3 234.3 232.8 229.3 220.5 235.4 255.8 261.0 247.5
330 321.5 300.4 283.9 285.1 285.8 297.6 280.2 318.3 305.3 354.0
331 287.6 285.0 301.8 320.1 316.3 293.3 290.1 334.8 293.7 371.6
332 234.9 242.8 273.0 309.8 317.5 310.0 454.2 346.8 330.3 253.9
```

Reading in Data
Reshaping the data:

```
> ###Back to tall form:
> sleepTall = melt(sleepWide,value.name="Reaction")
> dim(sleepTall)

[1] 180   3

> head(round(sleepTall,1))

  Var1 Var2 Reaction
1  308    0    249.6
2  309    0    222.7
3  310    0    199.1
4  330    0    321.5
5  331    0    287.6
6  332    0    234.9
```

Writing data

```
> write.csv(sleepWide,file="sleepWide.csv")
> write.table(sleepWide,file="sleepWide.txt",sep="\t")
```

Linear Algebra

Linear Algebra

Let's use the design matrix for sleepstudy:

```
> X = model.matrix(~Days,data=sleepstudy)
> class(X)

[1] "matrix"

> dim(X)

[1] 180   2
```

Linear Algebra

Is the matrix of full rank?

```
> qr(X)$rank
[1] 2
```

Linear Algebra

Matrix multiplication (and transposition)

```
> XtX = t(X)%*%X
> all.equal(XtX,crossprod(X,X))  #Uses less memory
[1] TRUE
```

Linear Algebra

Matrix Inversion

```
> XtXInv = solve(XtX)
> # XtXgenInv = MASS::ginv(Xtx)
> # XtXInv = chol2inv(chol(XtX))
```

Linear Algebra

OLS estimates:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

```
> XtXInv%*%t(X)%*%sleepstudy$Reaction
```

```
                  [,1]
(Intercept) 251.40510
Days         10.46729
```

```
> lm(Reaction~Days,data=sleepstudy)$coef
```

```
(Intercept)        Days
  251.40510    10.46729
```

Linear Algebra

US Judge ratings

```
A data frame containing 43 observations on 12 numeric variables.

[,1]   CONT  Number of contacts of lawyer with judge.
[,2]   INTG  Judicial integrity.
[,3]   DMNR  Demeanor.
[,4]   DILG  Diligence.
[,5]   CFMG  Case flow managing.
[,6]   DECI  Prompt decisions.
[,7]   PREP  Preparation for trial.
[,8]   FAMI  Familiarity with law.
[,9]   ORAL  Sound oral rulings.
[,10]  WRIT  Sound written rulings.
[,11]  PHYS  Physical ability.
[,12]  RTEN  Worthy of retention.
```

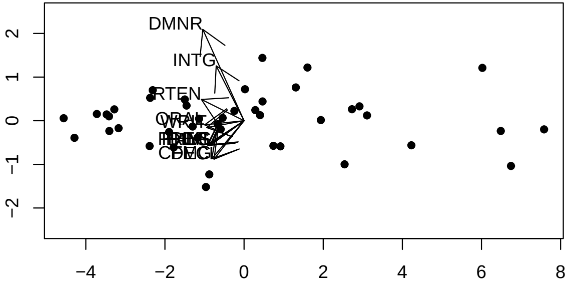
```
> ?USJudgeRatings
```

Linear Algebra

Principle Components (and (sloppy) biplot)

```
> judgesCent = scale(as.matrix(USJudgeRatings[,-1]),scale=F
> Sigma = cov(judgesCent)
> eigs = eigen(Sigma)
> Scores = judgesCent %*% eigs$vectors[,1:2]
> plot(Scores,pch=16,ylim=c(-1,1)*2.5,xlab="",ylab="")
> arrows(0,0,eigs$vec[,1]*3,eigs$vec[,2]*3)
> text(eigs$vec[,1:2]*3,labels=colnames(judgesCent),
+      adj=c(1,0))
```

Linear Algebra



Linear Algebra

Miscellaneous:

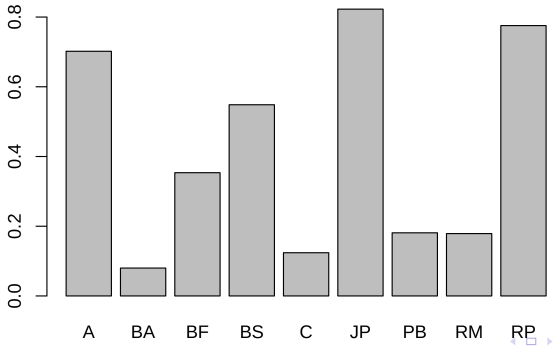
- kronecker product: `kronecker()` or %x%
- Hadamard Product: `A*B`
- Singular Value Decomposition: `svd()`
- Determinant or log determinant: `det()`, `determinant()`
- Diagonal, upper (or lower) triangle: `diag()`, `X[upper.tri(X)]`

Data Visualization

Univariate

Data visualization: univariate

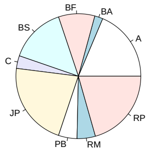
```
> yMeans = as.numeric(with(blowdown,by(y,SPP,mean)))
> barplot(yMeans,names.arg=levels(blowdown$SPP))
```

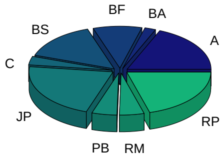


Data visualization: univariate

```
> par(mfrow=c(1,2))
> pie(yMeans,labels=levels(blowdown$SPP),
+     main="Survival rates")
> plotrix::pie3D(yMeans,labels=levels(blowdown$SPP),
+                explode=0.1,
+                col=rgb(20/256,c(1:9*20)/256,120/256))
> par(mfrow=c(1,1))
```

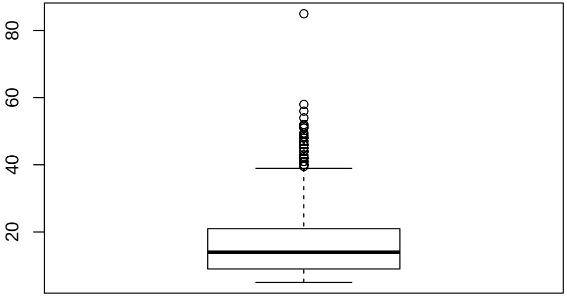
Data visualization: univariate





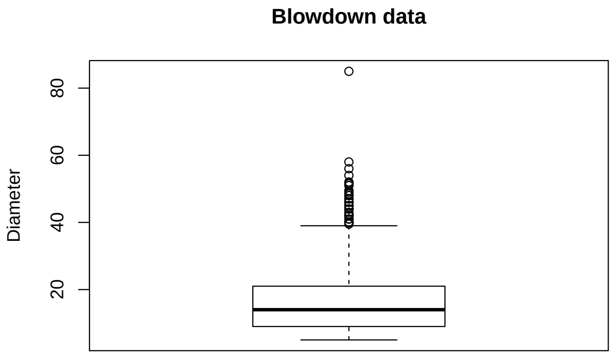
Data visualization: univariate

```
> boxplot(blowdown$D)
```



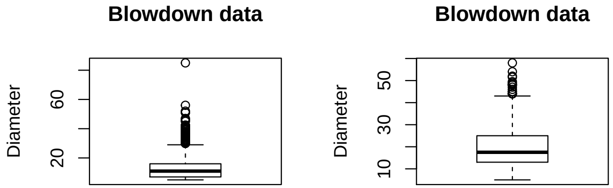
Data visualization: univariate

```
> boxplot(blowdown$D,main="Blowdown data",ylab="Diameter")
```



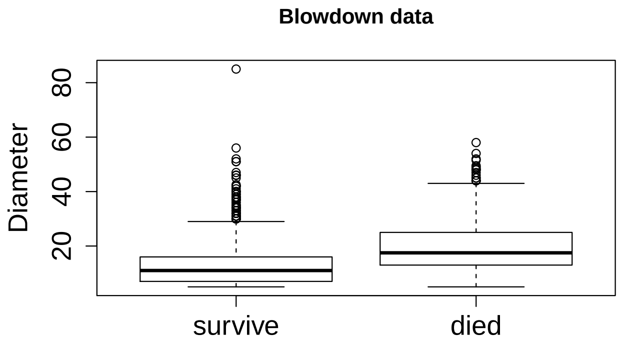
Data visualization: univariate

```
> par(mfrow=c(1,2))
> with(blowdown,by(D,y,boxplot,main="Blowdown data",
+                  ylab="Diameter"))
> par(mfrow=c(1,1))
```



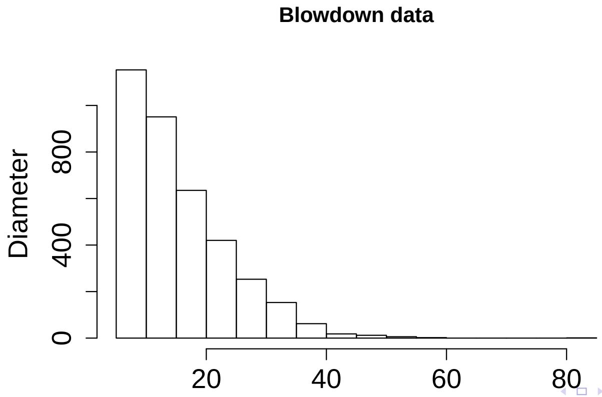
Data visualization: univariate

```
> boxplot(blowdown$D~blowdown$y,main="Blowdown data",
+         ylab="Diameter",names=c("survive","died"),
+         cex.lab=1.5,cex.axis=1.5)
```



Data visualization: univariate

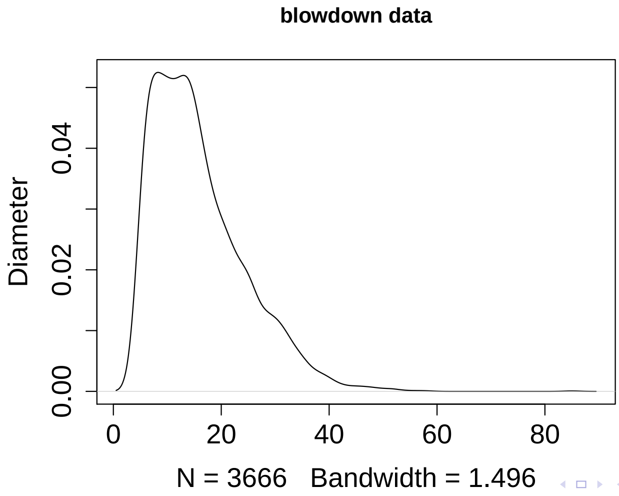
```
> hist(blowdown$D,main="Blowdown data",xlab="",
+          ylab="Diameter",cex.lab=1.5,cex.axis=1.5)
```



Data visualization: univariate

```
> plot(density(blowdown$D),ylab="Diameter",cex.lab=1.5,
+       cex.axis=1.5,main="blowdown data")
```

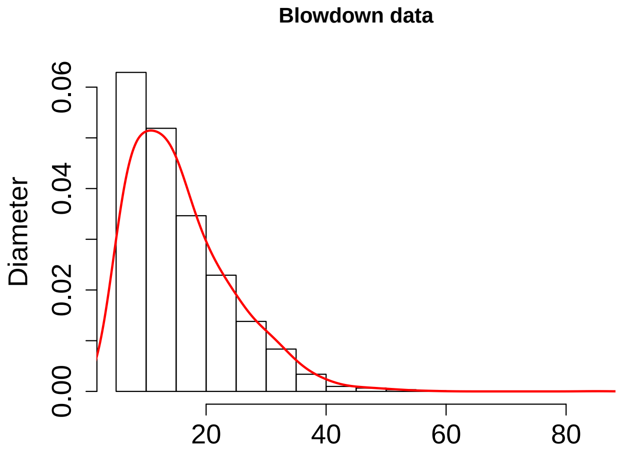
Data visualization: univariate



Data visualization: univariate

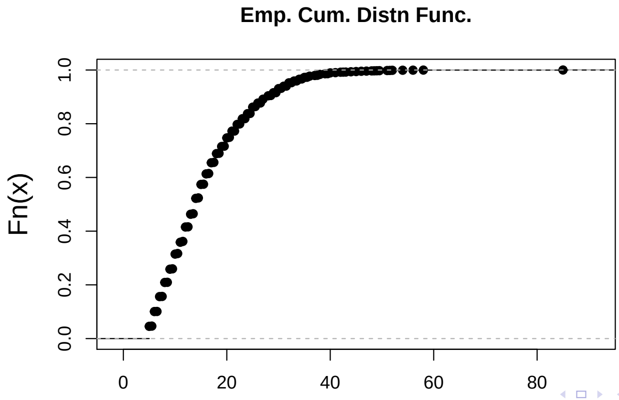
```
> hist(blowdown$D,main="Blowdown data",xlab="",freq=FALSE,
+          ylab="Diameter",cex.lab=1.5,cex.axis=1.5)
> lines(density(blowdown$D,bw=2.5),col="red",lwd=2)
```

Data visualization: univariate



Data visualization: univariate

```
> plot(ecdf(blowdown$D),main="Emp. Cum. Distn Func.",
+       xlab="",cex.lab=1.5)
```



Data visualization: univariate

Plotting a custom function: Example: Dose-response Exponential model

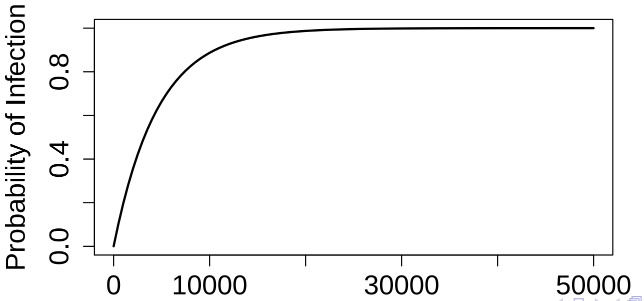
$$Prob(Infection) = 1 - e^{-rd}$$

```
> DRfun = function(dose){
+   return( 1-exp(-2.18E-04*dose) )
+ }
```

Data visualization: univariate

Plotting a custom function: Example: Dose-response Exponential model

$$Prob(Infection) = 1 - e^{-rd}$$

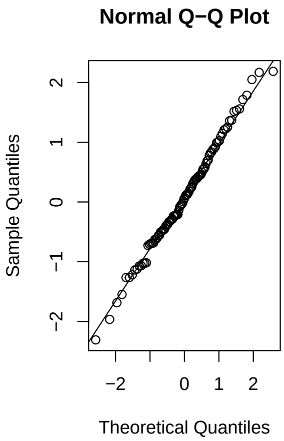


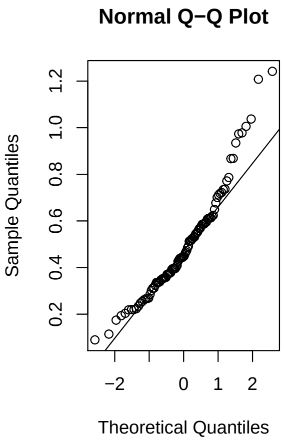
Data visualization: univariate

qq-plots

```
> set.seed(1)
> x1 = rnorm(100)
> x2 = rgamma(100,5,10)
> par(mfrow=c(1,2))
> qqnorm(x1);qqline(x1)
> qqnorm(x2);qqline(x2)
```

Data visualization: univariate



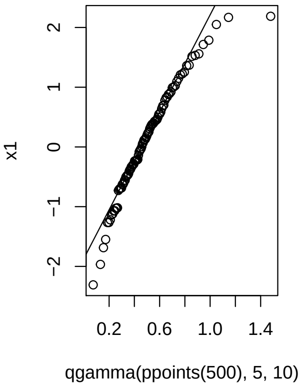


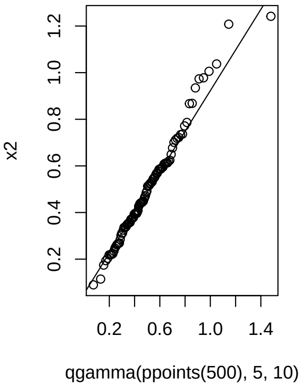
Data visualization: univariate

qq-plots

```
> par(mfrow=c(1,2))
> qqplot(qgamma(ppoints(500),5,10),x1)
> qqline(x1,distribution=function(p)qgamma(p,5,10))
> qqplot(qgamma(ppoints(500),5,10),x2)
> qqline(x2,distribution=function(p)qgamma(p,5,10))
```

Data visualization: univariate





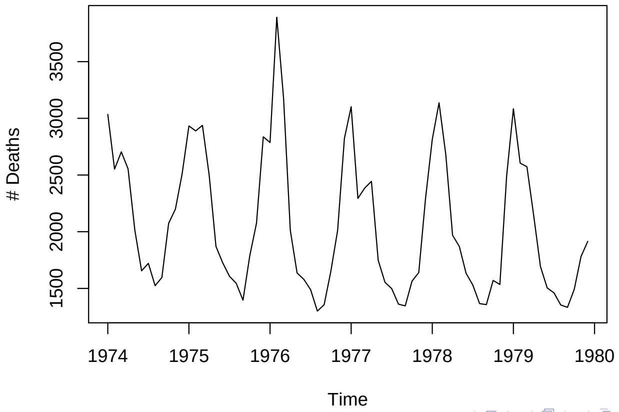
Data visualization: univariate

Time series:

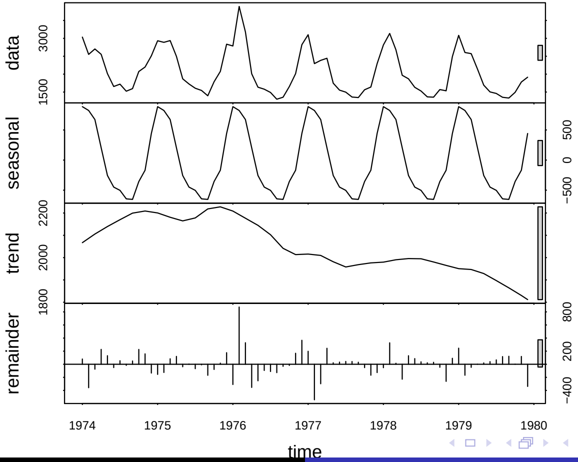
Example: Average monthly deaths from lung diseases in the UK from 1974-1979

```
> class(ldeaths)
> plot(ldeaths,ylab="# Deaths")
> plot(stl(ldeaths,s.window="periodic"))
```

Data visualization: univariate



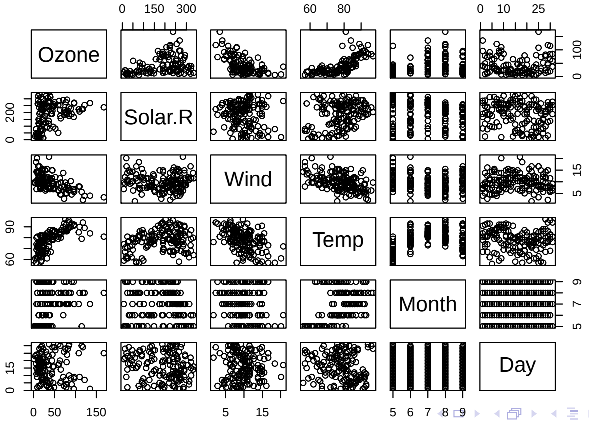
Data visualization: univariate



Multivariate

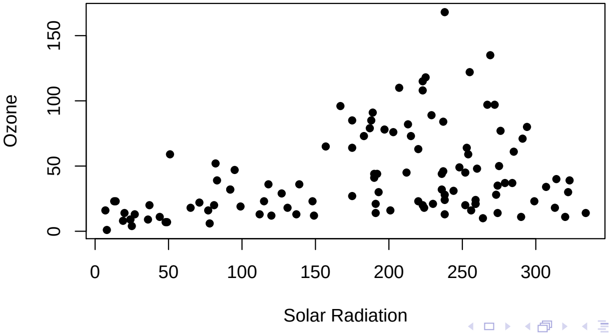
Data visualization: multivariate

```
> pairs(airquality)
```



Data visualization: multivariate

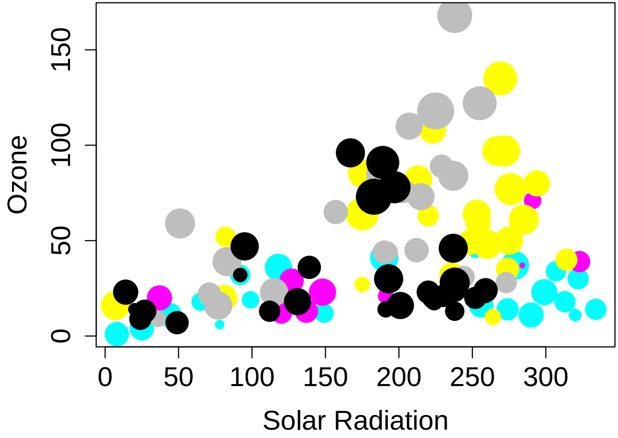
```
> plot(Ozone~Solar.R,data=airquality,xlab="Solar Radiation"
+       ylab="Ozone",pch=16)
```



Data visualization: multivariate

```
> CEXs = seq(0.5,5,length.out=500)
> CEXs = with(airquality,
+             CEXs[length(CEXs)*(max(Wind)+1-Wind)/
+                    (max(Wind)+1)])
> with(airquality,
+      plot(Ozone~Solar.R,xlab="Solar Radiation",
+      ylab="Ozone",pch=16,cex=CEXs,col=Month,
+      cex.lab=1.5,cex.axis=1.5))
```

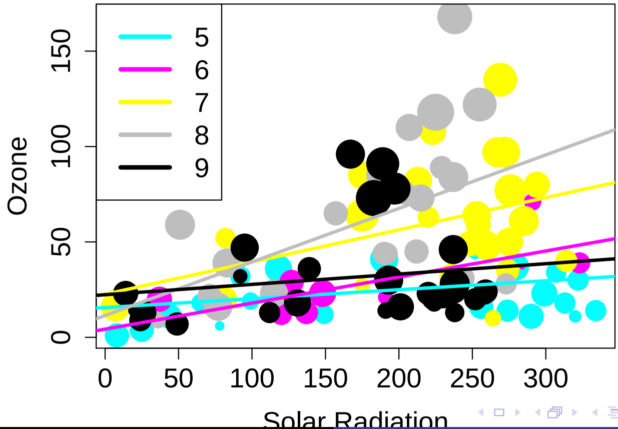
Data visualization: multivariate



Data visualization: multivariate

```
> with(airquality,sapply(unique(Month),
+        function(x){ind=which(Month==x);
+                        abline(lm(Ozone[ind]~Solar.R[ind]),
+                               col=x,lwd=3)}))
> with(airquality,legend("topleft",lwd=rep(4,5),cex=1.5,
+         col=unique(Month),legend=unique(Month)))
```

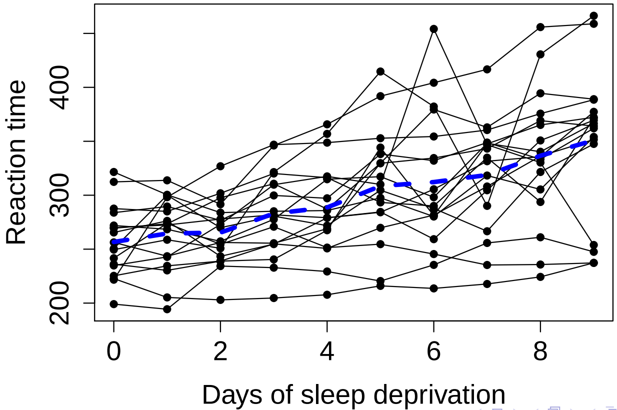
Data visualization: multivariate



Data visualization: multivariate

```
> plot(sleepTall$Reaction~sleepTall$Var2,pch=16,
+      xlab="Days of sleep deprivation",
+      ylab="Reaction time",cex.lab=1.5,cex.axis=1.5)
> for(i in 1:nrow(sleepWide)){
+   lines(sleepWide[i,]~c(0:9))
+ }
> lines(colMeans(sleepWide)~c(0:9),col="blue",lwd=4,lty=2)
```

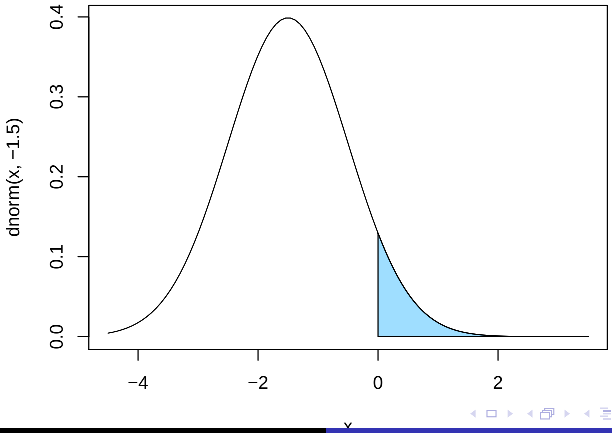
Data visualization: multivariate



Data visualization

```
> curve(dnorm(x,-1.5),-4.5,3.5)
> xseq = seq(0,3.5,length.out=500)
> yseq = dnorm(xseq,-1.5)
> polygon(x=c(xseq,xseq[length(xseq):1],0),
+         y=c(yseq,rep(0,length(xseq)+1)),
+         col=rgb(0.25,0.75,1,alpha=0.5))
```

Data visualization



Data visualization

Other miscellaneous functions:

```
> text()
> segments()
> arrows()
> symbols()
```

etc. . . .

Data visualization

You can save to disk your images:

```
> jpeg()
> pdf()
> png()
> bmp()
> tiff()
```

Data visualization

You can save to disk your images:

```
> jpeg()
> pdf()
> png()
> bmp()
> tiff()
```

Syntax is something like:

```
> jpeg("foo.jpg",height=800,width=800)#in pixels
> plot( ... )
> dev.off()
```

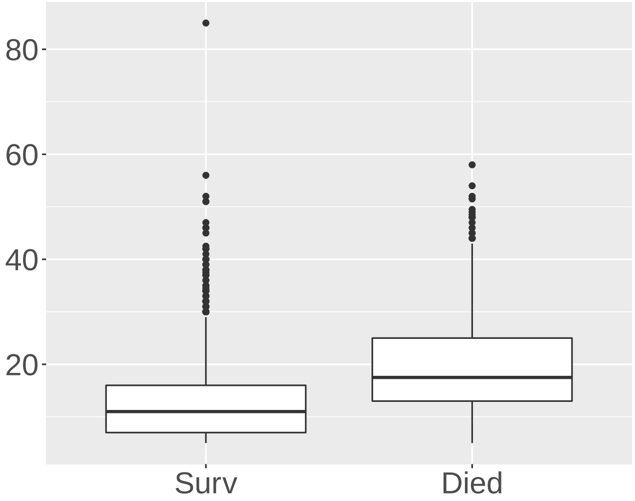
ggplot2

Data visualization: ggplot2

ggplot2: (http://docs.ggplot2.org/current/)
Syntax looks considerably different!

```
> require("ggplot2")
> blowdown$y <-
+   factor(blowdown$y,labels=c("Surv","Died"))
> ggplot(blowdown,aes(factor(y),D))+
+   geom_boxplot()+
+   theme(axis.text=element_text(size=20),
+         axis.title=element_text(size=0,colour="white"))
```

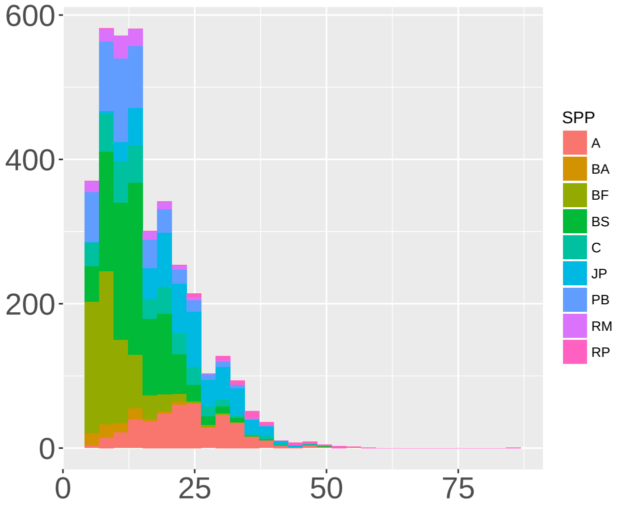
Data visualization: ggplot2



Data visualization: ggplot2

```
> ggplot(blowdown,aes(D,fill=SPP))+
+   geom_histogram()+
+   theme(axis.text=element_text(size=20),
+         axis.title=element_text(size=0,colour="white"))
```

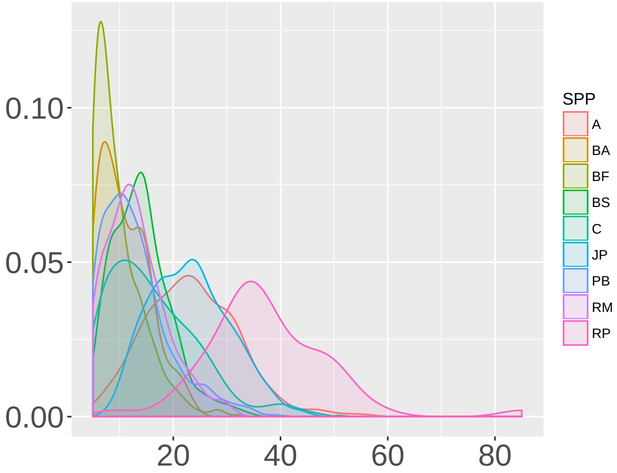
Data visualization: ggplot2



Data visualization: ggplot2

```
> ggplot(blowdown,aes(D,fill=SPP,colour=SPP))+
+   geom_density(alpha=0.1)+
+   theme(axis.text=element_text(size=20),
+         axis.title=element_text(size=0,colour="white"))
```

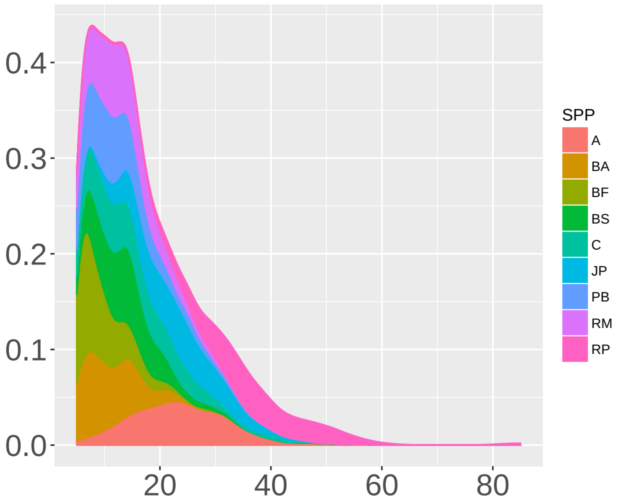
Data visualization: ggplot2



Data visualization: ggplot2

```
> ggplot(blowdown,aes(D,fill=SPP,colour=SPP))+
+   geom_density(position="stack")+
+   theme(axis.text=element_text(size=20),
+         axis.title=element_text(size=0,colour="white"))
```

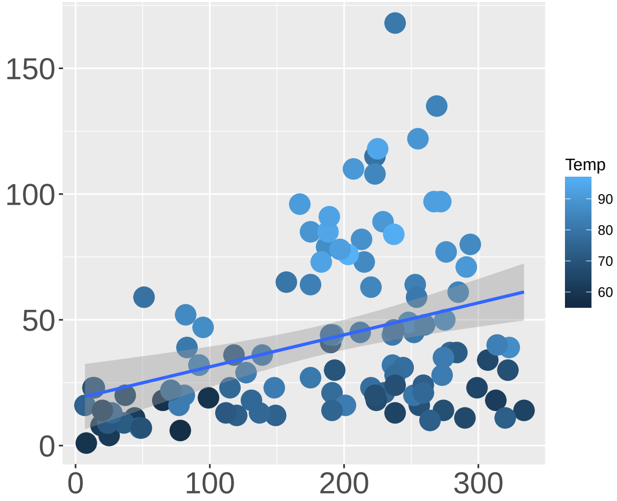
Data visualization: ggplot2



Data visualization: ggplot2

```
> ggplot(airquality,aes(Solar.R,Ozone))+
+   geom_point(size=6,aes(color=Temp))+
+   geom_smooth(method="lm")+
+   theme(axis.text=element_text(size=20),
+         axis.title=element_text(size=0,colour="white"))
```

Data visualization: ggplot2



3D plotting

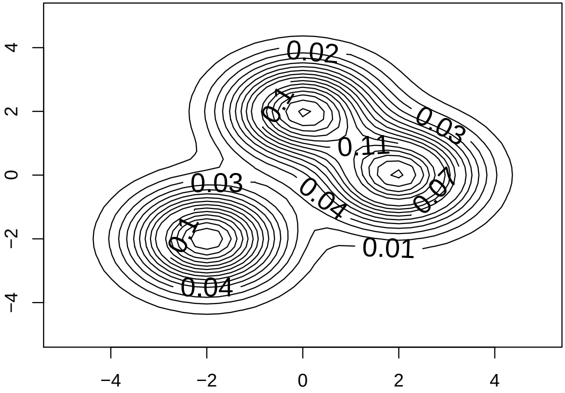
Data visualization: 3D

- scatterplot3d
- scatter3d
- plot3D / plot3Drgl
- emdbook

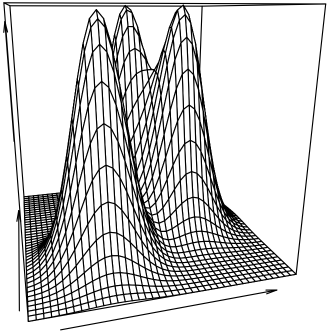
Data visualization: 3D

```
> normMix = function(x,y){
+   mixProbs = c(1/3,1/2,1/6)
+   ret = dnorm(x,-2)*dnorm(y,-2) +
+     dnorm(x,0)*dnorm(y,2) +
+     dnorm(x,2)*dnorm(y,0)
+   return(ret)
+ }
> curve3d(normMix(x,y),from=c(-5,-5),to=c(5,5),sys3d="contour",
+         xlab="",ylab="",labcex=1.5,nlevels=20)
> curve3d(normMix(x,y),from=c(-5,-5),to=c(5,5),sys3d="persp",theta=-15,
+         xlab="",ylab="",zlab="")
> curve3d(normMix(x,y),from=c(-5,-5),to=c(5,5),sys3d="rgl",
+         xlab="",ylab="",zlab="",
+         col = rgb(20/256,60/256,120/256,0.5))
```

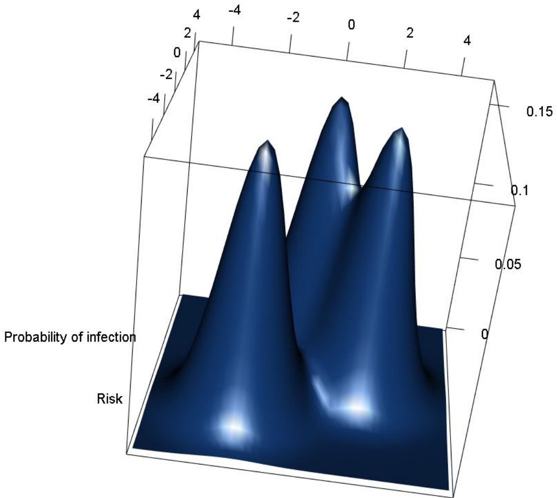
Data visualization: 3D



Data visualization: 3D



Data visualization: 3D

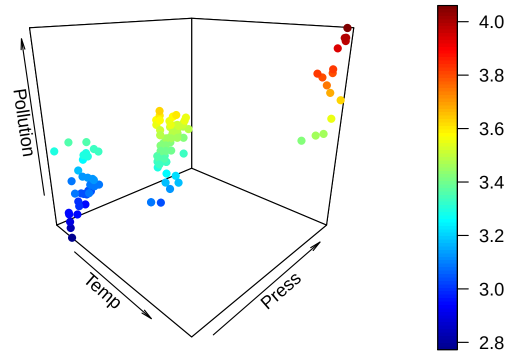


Data visualization: 3D

Hydrocarbon vapor pollution (g) vs. tank temperature (F) and and tank pressure (psi)

```
> require("plot3D");require("alr3")
> pollution = sniffer[,c("TankTemp","TankPres")]
> pollution = data.frame(pollution,logy=log(sniffer$Y))
> with(pollution,scatter3D(TankTemp,TankPres,logy,theta=45,
+                          phi=20,xlab="Temp",ylab="Press",
+                          zlab="Pollution",pch=16))
```

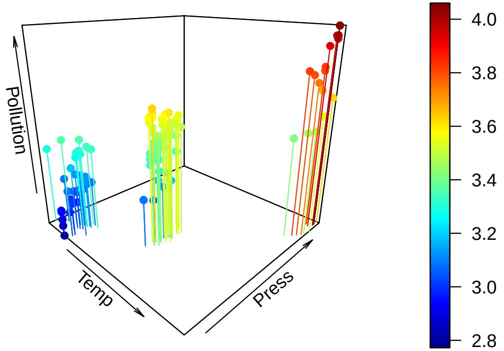
Data visualization: 3D



Data visualization: 3D

```
> with(pollution,scatter3D(TankTemp,TankPres,logy,theta=45,
+                          phi=20,xlab="Temp",ylab="Press",
+                          zlab="Pollution",pch=16,
+                          type="h"))
```

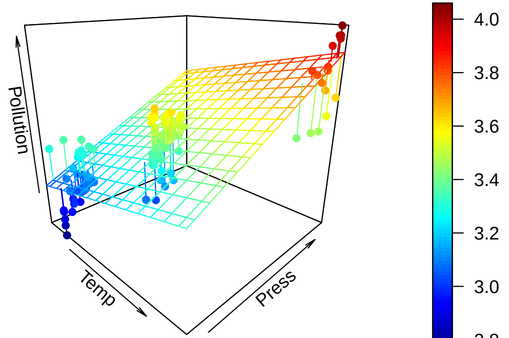
Data visualization: 3D



Data visualization: 3D

```
> fit = lm(logy~TankTemp+TankPres,data=pollution)
> xgrid=with(pollution,
+          seq(min(TankTemp),max(TankTemp),length.out=15))
> ygrid=with(pollution,
+          seq(min(TankPres),max(TankPres),length.out=15))
> xygrid = expand.grid(TankTemp=xgrid,TankPres=ygrid)
> logyPred = matrix(predict(fit,xygrid),15,15)
> fitPts = predict(fit)
> with(pollution,
+        scatter3D(TankTemp,TankPres,logy,theta=45,
+                  phi=20,xlab="Temp",ylab="Press",
+                  zlab="Pollution",pch=16,
+                  surf=list(x=xgrid,y=ygrid,z=logyPred,
+                  fit=fitPts,facets=NA)))
```

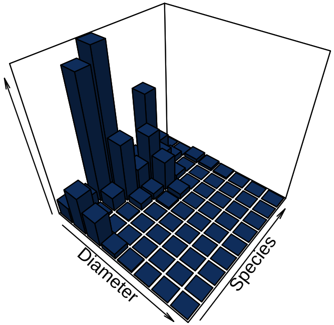
Data visualization: 3D



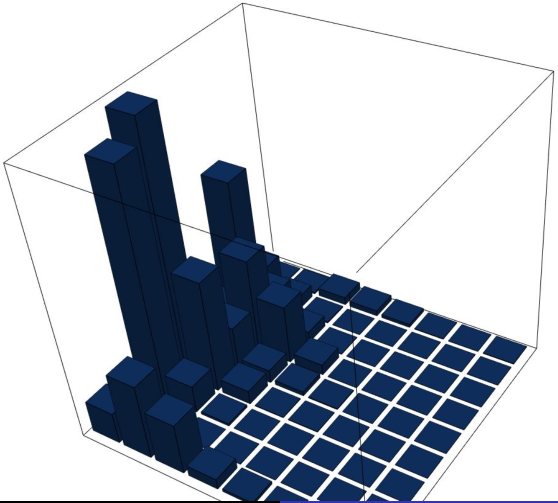
Data visualization: 3D

```
> BDTab = table(cut(blowdown$D,seq(5,85,by=10),
+                        include.lowest=TRUE),
+                   blowdown$SPP)
> hist3D(z=BDTab,col = rgb(20/256,60/256,120/256),
+        border = "black",shade = 0.4,space = 0.15,
+        xlab="Diameter",ylab="Species",zlab="")
> plotrgl()
```

Data visualization: 3D



Data visualization: 3D



Data visualization

Lots of other fun toys:

- Spatial visualization: Rgooglemaps, rworldmap, tmap, etc.
- Network visualization: igraph, networkDynamic, etc.
- animation
- Many others!!!

(Legal) Performance Enhancers

Rcpp

We'll focus on `compiler`, `foreach`, and `Rcpp` (with `inline`).
Our (trivial) example function will be

$$f(n) = \sum_{i=1}^{n} i \qquad \left(= \frac{n(n+1)}{2}\right)$$

```
> RFun = function(n,start=1){
+   ret=0
+   for(i in start:n){
+     ret = ret + i
+   }
+   return(ret)
+ }
> RFun(100) == 100*101/2

[1] TRUE
```

compiler

Thanks to our own Luke Tierney!

```
> require("compiler")
> RFunComp = cmpfun(RFun)
> system.time(RFun(1e7))

   user  system elapsed
   3.47    0.00    3.48

> system.time(RFunComp(1e7))

   user  system elapsed
   0.35    0.00    0.35
```

foreach (parallelizing)

If we have an embarassingly parallel task, use the `foreach` function.

```
> require("foreach")
> require("doParallel")
> registerDoParallel(cl=2)
> system.time({
+   temp = foreach(i=c(5e6,1e7)) %dopar% {
+     if(i==5e6){
+       RFun(i)
+     }else{
+       RFun(i,5e6+1)
+     }
+   }
+   print(sum(unlist(temp)) == 1e7*(1e7+1)/2)
+ })

[1] TRUE
   user  system elapsed
   0.02    0.00    2.81

> stopImplicitCluster()
```

Rcpp

Now in c++:

```
> require("Rcpp");require("inline")
> cppFun = cxxfunction(signature(nFromR="integer"),
+                      body='
+ int n = Rcpp::as<int>(nFromR);
+ int ret = 0;
+ for(int i=1;i<n+1;i++){
+   ret += i;
+ }
+ return wrap(ret);
+ ',plugin="Rcpp")
> cppFun(100)

[1] 5050
```

Rcpp

```
> system.time(RFun(1e7))
   user  system elapsed
   3.58    0.02    3.62
> system.time(cppFun(1e7))
   user  system elapsed
   0.02    0.00    0.01
```

Rcpp

Lots of additional Rcpp packages, such as `RcppSugar`, `RcppArmadillo`, etc.

Extended Examples

Blowdown data

Extended example on the blowdown data

Data from the Boundary Waters Canoe Area Wilderness Blowdown. The data frame blowdown includes nine species of trees.

```
> attach(blowdown)
> dim(blowdown)
```

```
[1] 3666    4
```

```
> head(blowdown)
```

```
   D         S    y SPP
1  9 0.0217509 Surv  BA
2 14 0.0217509 Surv  BA
3 18 0.0217509 Surv  BA
4 23 0.0217509 Surv  BA
5  9 0.0217509 Surv  BA
6 16 0.0217509 Surv  BA
```

Extended example on the blowdown data

Does species influence survival?

```
> y = factor(y,labels=c("Survived","Died"))
> ( tab = table(SPP,y) )
```

```
    y
SPP  Survived Died
  A       130  306
  BA       69    6
  BF      426  233
  BS      438  532
  C       311   44
  JP       89  413
  PB      407   90
  RM      101   22
  RP       11   38
```

Extended example on the blowdown data

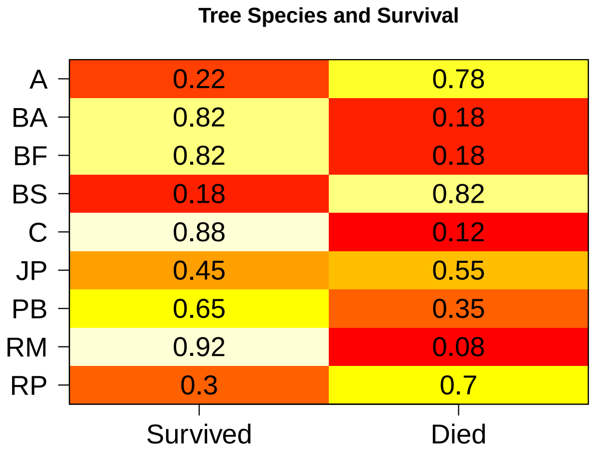
Does species influence survival?

```
> chisq.test(tab)

        Pearson's Chi-squared test


data:  tab
X-squared = 848.72, df = 8, p-value < 2.2e-16
> prop.table(tab,1)
    y
SPP   Survived      Died
  A  0.2981651 0.7018349
  BA 0.9200000 0.0800000
  BF 0.6464340 0.3535660
  BS 0.4515464 0.5484536
  C  0.8760563 0.1239437
```

Extended example on the blowdown data

```
> image(t(prop.table(tab,1)),xaxt="n",yaxt="n",
+       main="Tree Species and Survival")
> box()
> axis(1,at=0:1,labels=levels(y),cex.axis=1.5)
> axis(2,at=seq(1,0,length.out=length(levels(SPP))),
+      labels=levels(SPP),las=1,cex.axis=1.5)
> text(cbind(rep(0:1,each=length(levels(SPP))),
+            rep(seq(0,1,length.out=length(levels(SPP))),2)
+      labels=round(c(prop.table(tab,1)),2),cex=1.5)
```

Extended example on the blowdown data



Extended example on the blowdown data

How about the diameter?

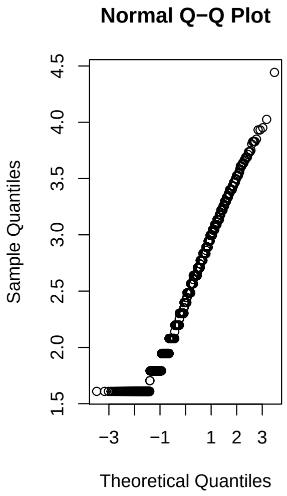
```
> logD = log(D)
> t.test(logD~y,var.equal=FALSE)

        Welch Two Sample t-test

data:  logD by y
t = -27.062, df = 3663.9, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equ
95 percent confidence interval:
 -0.4645252 -0.4017640
sample estimates:
mean in group Survived     mean in group Died
              2.427742               2.860887
```

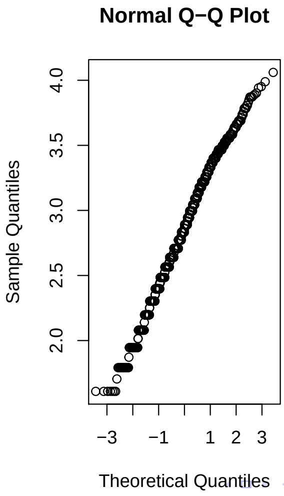
Extended example on the blowdown data

Are the results valid? I.e., are they normal? Probably not. . .

```
> par(mfrow=c(1,2))
> by(logD,y,qqnorm)
> par(mfrow=c(1,1))
```

Extended example on the blowdown data





Extended example on the blowdown data

Let's do a non-parametric test to verify the association:

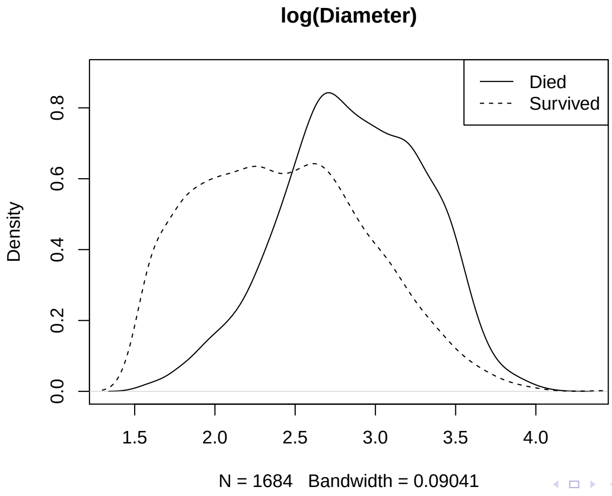
```
> kruskal.test(logD,y)

        Kruskal-Wallis rank sum test

data:  logD and y
Kruskal-Wallis chi-squared = 598.53, df = 1, p-value < 2.2e-16
```

Extended example on the blowdown data

Density plots by survival:

```
> ind=which(y=="Died")
> plot(density(x=logD[ind]),ylim=c(0,0.9),
+      main="log(Diameter)")
> lines(density(x=logD[-ind]),lty=2)
> legend("topright",legend=c("Died","Survived"),lty=1:2)
```

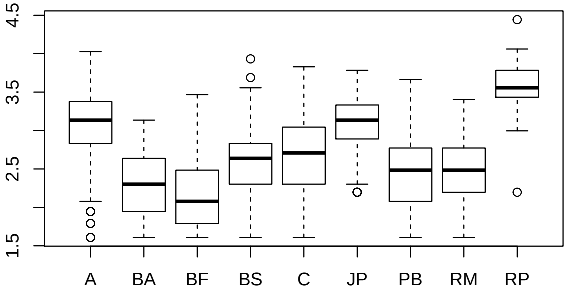
Extended example on the blowdown data



Extended example on the blowdown data

So there likely exists this confounding factor in that species have differing avg. diameters.

```
> boxplot(logD~SPP)
```



Extended example on the blowdown data

Is species related to diameter?

```
> lmMod= lm(logD~SPP)
> anova(lmMod)
```

```
Analysis of Variance Table

Response: logD
            Df Sum Sq Mean Sq F value    Pr(>F)
SPP          8 405.36  50.670  287.95 < 2.2e-16 ***
Residuals 3657 643.52   0.176
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

Extended example on the blowdown data

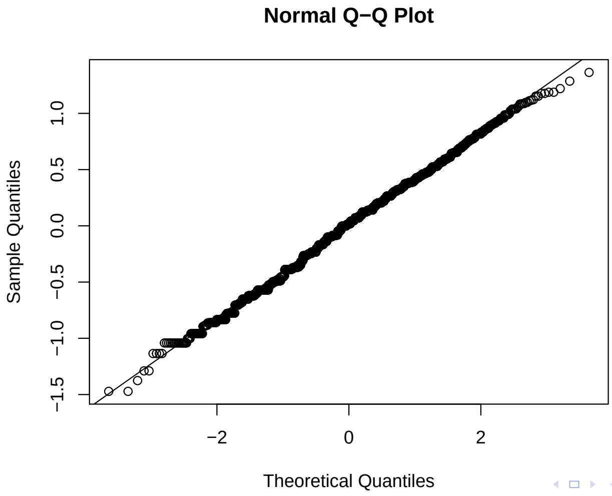
```
> summary(lmMod)

Call:
lm(formula = logD ~ SPP)

Residuals:
     Min       1Q   Median       3Q      Max
-1.47181 -0.26516  0.01793  0.29341  1.36408

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.08125    0.02009 153.374  < 2e-16 ***
SPPBA       -0.77845    0.05244 -14.845  < 2e-16 ***
SPPBF       -0.90196    0.02590 -34.829  < 2e-16 ***
SPPBS       -0.51351    0.02419 -21.231  < 2e-16 ***
SPPC        -0.43022    0.02999 -14.346  < 2e-16 ***
SPPJP        0.01014    0.02746   0.369    0.712
SPPPB       -0.63780    0.02753 -23.171  < 2e-16 ***
SPPRM       -0.60928    0.04283 -14.226  < 2e-16 ***
SPPRP        0.49170    0.06320   7.780  9.4e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4195 on 3657 degrees of freedom
Multiple R-squared:  0.3865,     Adjusted R-squared:  0.3851
F-statistic: 287.9 on 8 and 3657 DF,  p-value: < 2.2e-16
```

Extended example on the blowdown data

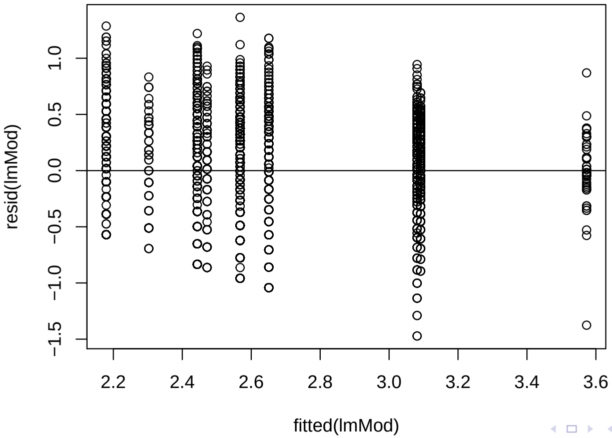
Let's check the model assumptions:

```
> qqnorm(resid(lmMod));qqline(resid(lmMod))
> plot(resid(lmMod)~fitted(lmMod));abline(h=0)
```

Extended example on the blowdown data



Extended example on the blowdown data

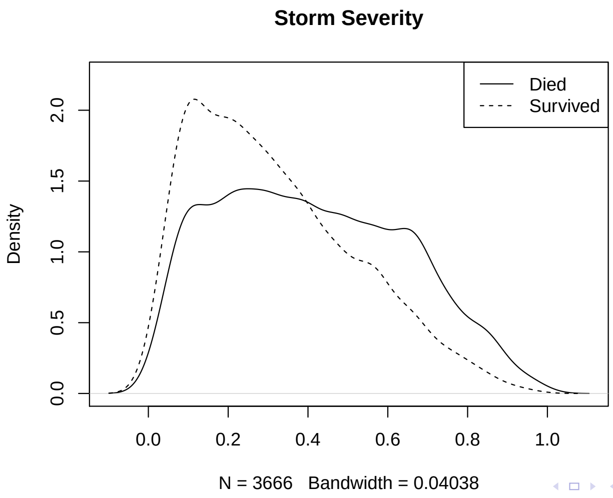


Extended example on the blowdown data

Another factor to consider is the severity of the storm

```
> plot(density(x=S),ylim=c(0,2.25),main="Storm Severity")
> lines(density(x=S[-ind]),lty=2)
> legend("topright",legend=c("Died","Survived"),lty=1:2)
```

Extended example on the blowdown data



Extended example on the blowdown data

A more appropriate model would be logistic regression

```
> ( logRegMod = glm(y~logD + SPP + S, family=binomial) )

Call:  glm(formula = y ~ logD + SPP + S, family = binomial)

Coefficients:
(Intercept)         logD        SPPBA        SPPBF        SPPBS
 -5.9971951    1.5813423   -2.2427869    0.0002284    0.1672262
      SPPJP        SPPPB        SPPRM        SPPRP            S
  1.0399651   -1.7235679   -1.7956738    0.0031381    4.6288861

Degrees of Freedom: 3665 Total (i.e. Null);  3655 Residual
Null Deviance:          5058
Residual Deviance: 3259         AIC: 3281
```

Extended example on the blowdown data

```
> round(summary(logRegMod)$coef,4)

             Estimate Std. Error  z value Pr(>|z|)
(Intercept)   -5.9972     0.3748 -15.9993   0.0000
logD           1.5813     0.1115  14.1876   0.0000
SPPBA         -2.2428     0.4936  -4.5439   0.0000
SPPBF          0.0002     0.1789   0.0013   0.9990
SPPBS          0.1672     0.1518   1.1020   0.2705
SPPC          -2.0765     0.2162  -9.6031   0.0000
SPPJP          1.0400     0.1788   5.8176   0.0000
SPPPB         -1.7236     0.1865  -9.2435   0.0000
SPPRM         -1.7957     0.3019  -5.9472   0.0000
SPPRP          0.0031     0.4132   0.0076   0.9939
S              4.6289     0.2128  21.7477   0.0000
```

Extended example on the blowdown data

We could also have done a probit regression model

```
> ( probRegMod = glm(y~logD + SPP + S,
+                     family=binomial(link="probit")) )

Call:  glm(formula = y ~ logD + SPP + S, family = binomial(link

Coefficients:
(Intercept)         logD        SPPBA        SPPBF        SPPBS
  -3.350235     0.875785    -1.186295    -0.031899     0.091554
      SPPJP        SPPPB        SPPRM        SPPRP            S
   0.600456    -0.971438    -0.966891    -0.005013     2.652496

Degrees of Freedom: 3665 Total (i.e. Null);  3655 Residual
Null Deviance:            5058
Residual Deviance: 3281         AIC: 3303
```

Extended example on the blowdown data

We could also have done a probit regression model

```
> summary(probRegMod)$coef

              Estimate Std. Error     z value      Pr(>|z|)
(Intercept) -3.35023538 0.20911836 -16.02076157  9.152243e-58
logD         0.87578494 0.06240703  14.03343517  9.732093e-45
SPPBA       -1.18629531 0.26124554  -4.54092074  5.600909e-06
SPPBF       -0.03189922 0.10327689  -0.30887084  7.574198e-01
SPPBS        0.09155382 0.08782421   1.04246677  2.971953e-01
SPPC        -1.17118254 0.11825869  -9.90356469  4.016963e-23
SPPJP        0.60045604 0.10104811   5.94227897  2.810865e-09
SPPPB       -0.97143796 0.10571710  -9.18903323  3.963877e-20
SPPRM       -0.96689104 0.16695204  -5.79142987  6.978970e-09
SPPRP       -0.00501280 0.23671805  -0.02117625  9.831051e-01
S            2.65249622 0.11714384  22.64307109 1.632635e-113
```

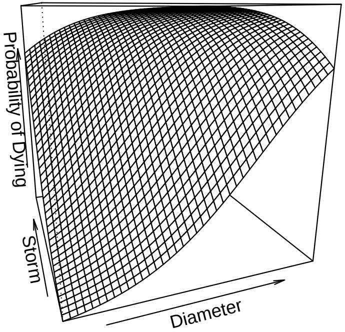
Extended example on the blowdown data

We can also plot the probability surface for a particular species, say, black spruce.

```
> logitFun = function(x,y){
+   eta = logRegMod$coef["(Intercept)"] +
+     logRegMod$coef["SPPBS"]+logRegMod$coef["logD"]*x+
+     logRegMod$coef["S"]*y
+   return(1/(1+exp(-eta)))
+ }
> with(blowdown,
+      curve3d(logitFun,from=c(min(logD),min(S)),
+              to=c(max(logD),max(S)),sys3d="persp",
+              theta=-20,xlab="Diameter",ylab="Storm",
+              zlab="Probability of Dying"))
```

Extended example on the blowdown data

We can also plot the probability surface for a particular species, say, black spruce.



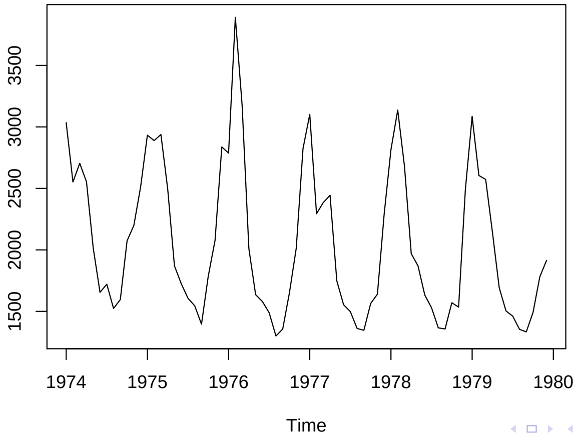
UK monthly deaths from lung diseases

Extended example on the lung diseases data

Consider monthly deaths from bronchitis, emphysema and asthma in the UK, 1974-1979.

```
> yr <- floor(tt <- time(mdeaths))
> plot(ldeaths,ylab="",
+       xy.labels=paste(month.abb[12*(tt-yr)],
+                        yr-1900,sep="'"))
```

Extended example on the lung diseases data



Extended example on the lung diseases data

Specifically we want to get predicted values for future numbers of lung disease fatalities.
The statistical model is

$$y_t = \phi y_{t-1} + \epsilon_t$$

where $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$. We'll construct a Gibbs sampler (a MCMC algorithm). This can be done by iteratively drawing from

$$\pi(\sigma^2 | data)$$
$$\pi(\psi | \sigma^2, data)$$
$$\pi(y_{new} | \psi, \sigma^2, data)$$

Extended example on the lung diseases data

These conditional distributions (with uninformative improper priors) are given as follows. First, let

$$\begin{aligned} B &= \sum_t y_{t-1}^2 \\ b &= \sum_t y_t y_{t-1}/B \\ Q_b &= \sum_t (y_t - b y_{t-1})^2. \end{aligned}$$

Then we have

$$\begin{aligned} \pi(\sigma^2|\phi, data) &\sim \Gamma^{-1}(0.5(T-2), 0.5Q_b) \\ \pi(\phi|\sigma^2, data) &\sim N(b, \sigma^2/B) \\ \pi(y_{new}|\phi, \sigma^2, data) &\sim N(\phi y_T, \sigma^2). \end{aligned}$$

Extended example on the lung diseases data

Now to do this in R.

```
> require("MCMCpack")
> fun1= function(y,nsims=1000,stepsAhead=100){
+   TT = length(y)
+   BB = sum(y[-TT]^2)
+   bb = sum(y[-1]*y[-TT])/BB
+   Qb = sum((y[-1]-bb*y[-TT])^2)
+
+   s2 = rinvgamma(nsims,shape=0.5*(TT-3),scale=0.5*Qb)
+   phi = rnorm(nsims,mean=bb,sd=sqrt(s2/BB))
+   ypred = matrix(0.0,nsims,stepsAhead)
+   ypred[,1]=rnorm(nsims,phi*y[TT],sd=sqrt(s2))
+   for(tt in 2:stepsAhead){
+     ypred[,tt] = rnorm(nsims,phi*ypred[,tt-1],sd=sqrt(s2))
+   }
+   return(list(s2=s2,phi=phi,ypred=ypred))
+ }
```

Extended example on the lung diseases data

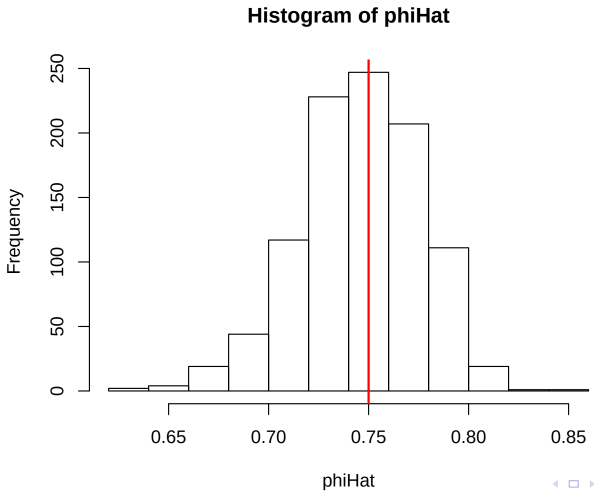
Let's test the Bayesian estimators out via simulation:

```
> set.seed(1)
> M=1000
> phiTrue = 0.75
> s2True = 0.5
> TT=500
> Y = matrix(NA,M,TT+1)
> Y[,1]=rnorm(M,sd=sqrt(s2True/(1-phiTrue^2)))
> phiHat = s2Hat = numeric(M)
> for(tt in 1+1:TT){
+   Y[,tt] = 0.75*Y[,tt-1] + rnorm(M,sd=sqrt(s2True))
+ }
> for(iter in 1:M){
+   temp = fun1(Y[iter,],stepsAhead=2)
+   s2Hat[iter] = mean(temp$s2)
+   phiHat[iter] = mean(temp$phi)
+ }
```

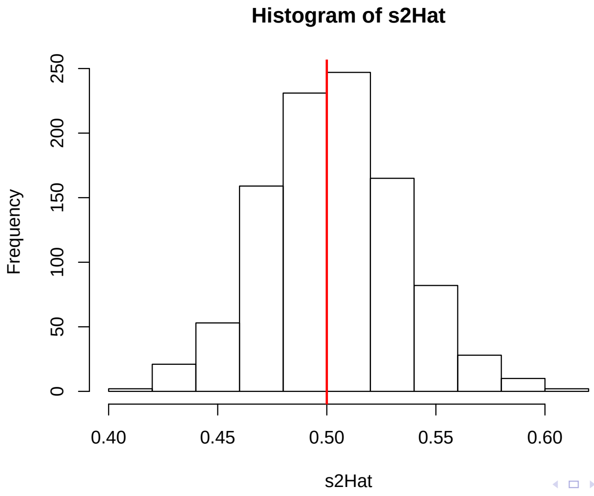
Extended example on the lung diseases data

```
> hist(phiHat)
> abline(v=phiTrue,col="red",lwd=2)
> hist(s2Hat)
> abline(v=s2True,col="red",lwd=2)
```

Extended example on the lung diseases data



Extended example on the lung diseases data



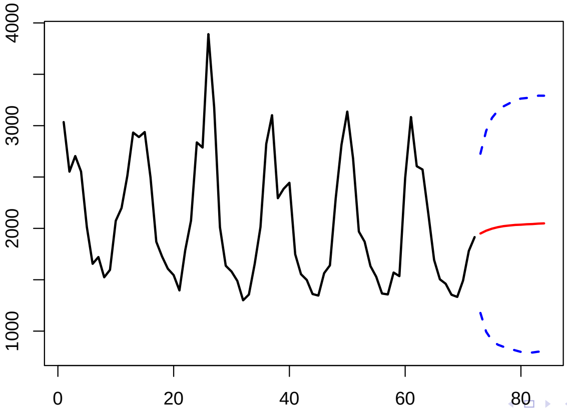
Extended example on the lung diseases data

Now let's apply this to our lung disease mortality data:

```
> ldeaths1 = ldeaths-mean(ldeaths)
> fit1 = fun1(ldeaths1,nsims=1e5,stepsAhead=12)
> ypreds = colMeans(fit1$ypred)+mean(ldeaths)
> yBounds = t(apply(fit1$ypred,2,quantile,
+                 probs=c(0.025,0.975)))+mean(ldeaths)
> YL = range(c(ldeaths,yBounds))
> plot(c(ldeaths),ylab="",xlab="",type="l",
+      xlim=c(1,(length(ldeaths)+12)),ylim=YL,lwd=2)
> lines(ypreds~c(length(ldeaths)+1:12),col="red",lwd=2)
> lines(yBounds[,1]~c(length(ldeaths)+1:12),col="blue",
+       lwd=2,lty=2)
> lines(yBounds[,2]~c(length(ldeaths)+1:12),col="blue",
+       lwd=2,lty=2)
```

Extended example on the lung diseases data

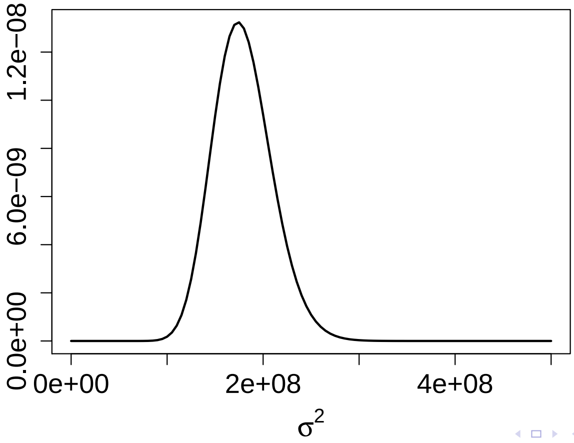
Now let's apply this to our lung disease mortality data:



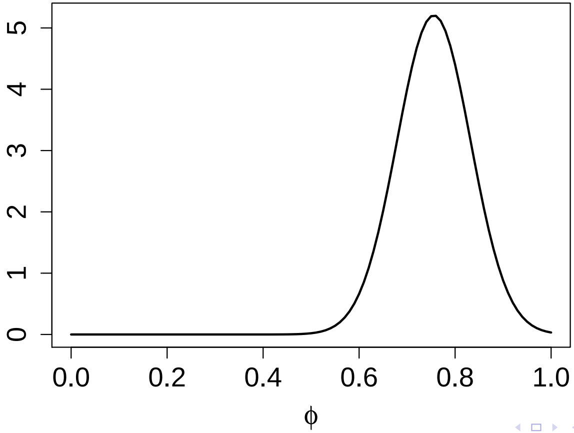
Extended example on the lung diseases data

```
> TT = length(ldeaths1);  BB = sum(ldeaths1[-TT]^2);
> bb = sum(ldeaths1[-1]*ldeaths1[-TT])/BB;
> Qb = sum((ldeaths1[-1]-bb*ldeaths1[-TT])^2)
> curve(dgamma(x,shape=0.5*(TT-3),scale=0.5*Qb),
+       lwd=2,xlab=expression(sigma^2),from=0,to=5e8,
+       ylab="",cex.axis=1.5,cex.lab=1.5)
> curve(dnorm(x,bb,sd=sqrt(mean(fit1$s2)/BB)),
+       lwd=2,xlab=expression(phi),from=0,to=1,
+       ylab="",cex.axis=1.5,cex.lab=1.5)
```

Extended example on the lung diseases data



Extended example on the lung diseases data



Extended example on the lung diseases data

Can we make it faster?

```
> require("compiler")
> fun1Comp = cmpfun(fun1)

funcpp = cxxfunction(
  signature(YY="numeric",NSIMS="integer",STEPSAHEAD="intege
  body='

//Environment stats("package:stats");
//Function rnorm = stats["rnorm"];
Environment MCMCpack("package:MCMCpack");
Function rinvgamma = MCMCpack["rinvgamma"];
```

⋮

Extended example on the lung diseases data

Let's compare:

```
> system.time({for(it in 1:100){
+   Rsims = fun1(Y[1,])
+   }
+ })

   user  system elapsed
   1.34    0.00    1.35

> system.time({for(it in 1:100){
+   RsimsComp = fun1Comp(Y[1,])
+   }
+ })

   user  system elapsed
   1.56    0.00    1.59

> system.time({for(it in 1:100){
+   Rcppsims = funcpp(Y,NSIMS=1000,STEPSAHEAD=100)
+   }
+ })

   user  system elapsed
   0.89    0.08    0.97
```

THE END