The R Book

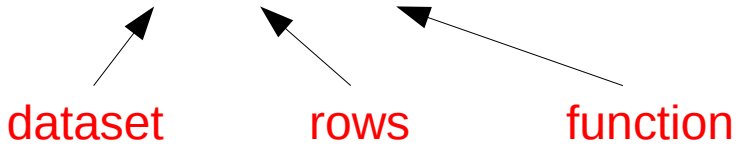
Chapter 2: Essentials of the R Language
Session 12

Evaluating Functions with apply, sapply and lapply

apply
- to apply functions to rows or columns of matrices or dataframes

```
> (X<-matrix(1:24,nrow=4))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24

> apply(X,1,sum)
```




```
[1] 66 72 78 84
```

Evaluating Functions with apply, sapply and lapply

apply
- to apply functions to rows or columns of matrices or dataframes

```
> apply(X,1,function(x) x^2+x)
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    6   12   20
[2,]   30   42   56   72
[3,]   90  110  132  156
[4,]  182  210  240  272
[5,]  306  342  380  420
[6,]  462  506  552  600
```

"anonymous function",
not named

Evaluating Functions with apply, sapply and lapply

sapply
- to apply functions to vectors
- useful with complex iterative calculations

```
> sapply(3:7, seq)
[[1]]
[1] 1 2 3

[[2]]
[1] 1 2 3 4

[[3]]
[1] 1 2 3 4 5

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

[[5]]
[1] 1 2 3 4 5 6 7
```

Evaluating Functions with apply, sapply and lapply

sapply

Example : Decay of radioactive emissions, over a 50-day period
- use non-linear least squares to **estimate the decay constant a** in

$$y = \exp(-ax)$$

a) upload a dataset

```
> sapdecay<-read.table("/home/ .../sapdecay.txt", header=T)
> attach(sapdecay)
> names(sapdecay)
[1] "ex" "wy"
```

b) calculate the sum of the squares wy ~ predicted (yf) values

```
> sumsq <- function(a,xv=ex,yv=wy)
+ { yf <- exp(-a*xv)
+ sum((yv-yf)^2) }
```

Evaluating Functions with apply, sapply and lapply

sapply

- use non-linear least squares to **estimate the decay constant a**

$$y = \exp(-ax)$$

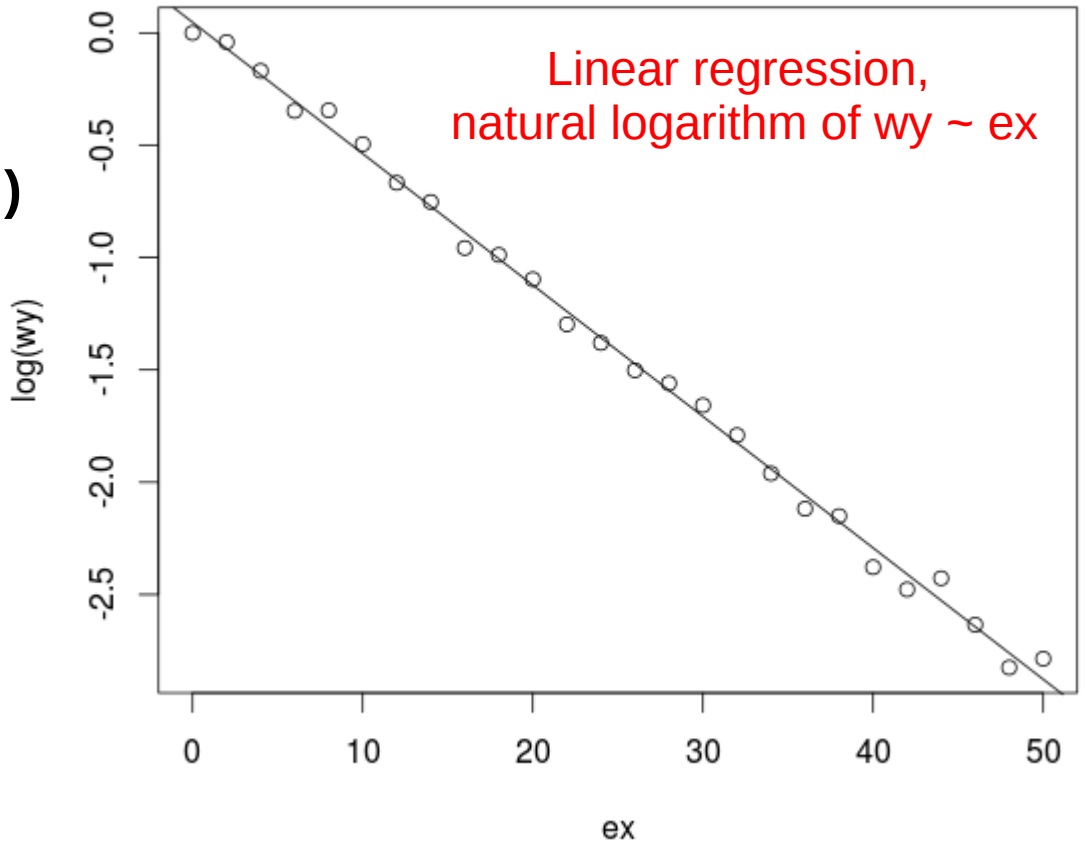
c) get a rough idea of the decay constant a,

```
> lm(log(wy)~ ex)

Call:
lm(formula = log(wy) ~ ex)

Coefficients:
(Intercept)           ex
    0.04688     -0.05849

> plot(log(wy)~ex)
> abline(lm(log(wy)~ex))
```



Evaluating Functions with apply, sapply and lapply

sapply

- use non-linear least squares to **estimate the decay constant a**

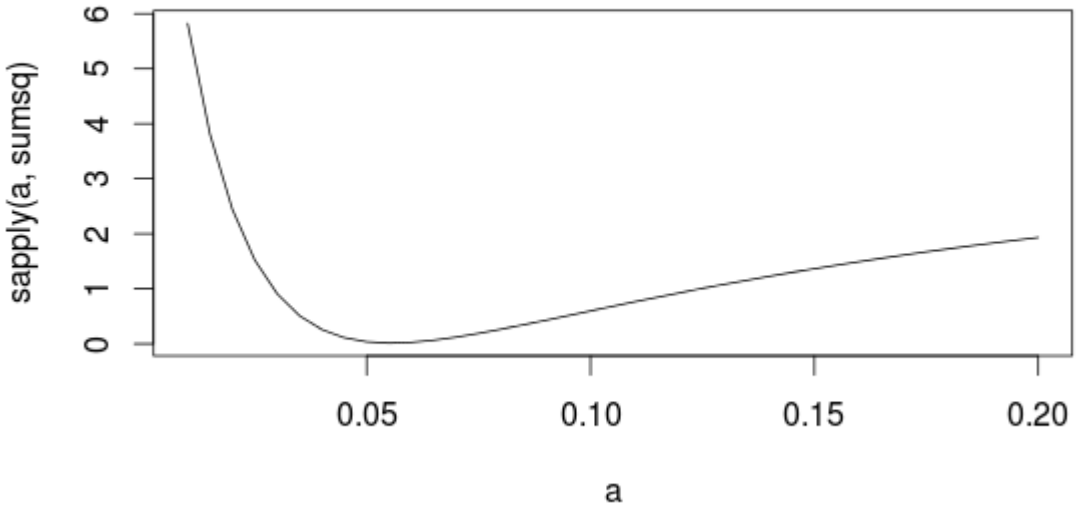
d) generate a range of values for a on either side of 0.058

```
> a<-seq(0.01,0.2,.005)
> a
 [1] 0.010 0.015 0.020 0.025 0.030 0.035 0.040 0.045 0.050 0.055 0.060 0.065 0.070 0.075 0.080
[16] 0.085 0.090 0.095 0.100 0.105 0.110 0.115 0.120 0.125 0.130 0.135 0.140 0.145 0.150 0.155
[31] 0.160 0.165 0.170 0.175 0.180 0.185 0.190 0.195 0.200
```

e) use sapply to apply the sum of squares for each value «a»

```
> plot(a,sapply(a,sumsq),type="l")
```

Type "l" for lines




```
> sumsq<-function(a,xv=ex,yv=wy)
+ { yf <- exp(-a*xv)
+ sum((yv-yf)^2) }
```

Evaluating Functions with apply, sapply and lapply

sapply

- use non-linear least squares to **estimate the decay constant a**

f) determine « a » with the smallest sum of squares

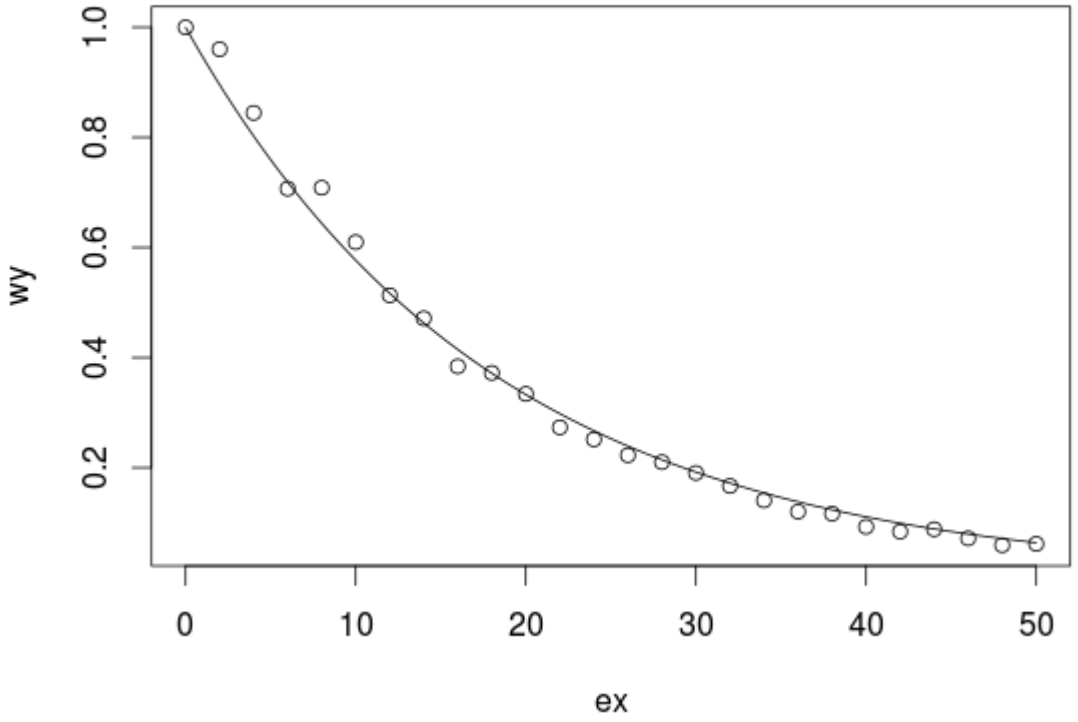
```
> a[min(sapply(a,sumsq))==sapply(a,sumsq)]
[1] 0.055
```

g) plot decay values with regression function and fitted « a »

```
> plot(ex,wy)
> xv<-seq(0,50,0.1)
> lines(xv,exp(-0.055*xv))
```

- use :

```
xv<-seq(0,50,10)
```



Evaluating Functions with apply, sapply and lapply

sapply

- use non-linear least squares to **estimate the decay constant a**

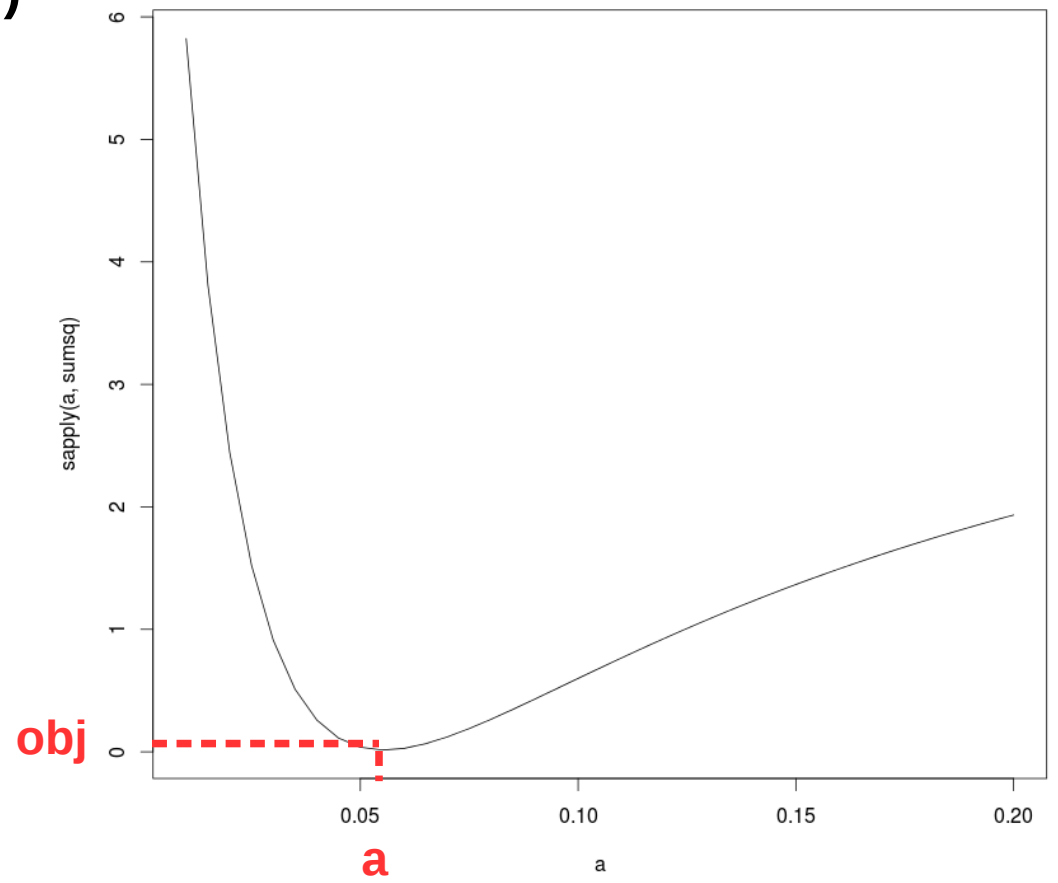
h) streamlined version using the optimize function

```
> fa<-function(a) sum((wy-exp(-a*ex))^2)
> optimize(fa,c(0.01,0.1))

$minimum
[1] 0.05538411

$objective
[1] 0.01473559
```

> sumsq(0.05538411)
[1] 0.01473559



Evaluating Functions with apply, sapply and lapply

Lists and lapply

```
> a<-c("a","b","c","d")
> b<-c(1,2,3,4,4,3,2,1)
> c<-c(T,T,F)
```

character information
numeric information
logical information

```
> list.object<-list(a,b,c)
> class(list.object)
[1] "list"
```

combine into a list

```
> list.object
[[1]]
[1] "a" "b" "c" "d"

[[2]]
[1] 1 2 3 4 4 3 2 1

[[3]]
[1]  TRUE TRUE FALSE
```

```
> lapply(list.object,length)
[[1]]
[1] 4

[[2]]
[1] 8

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

Evaluating Functions with apply, sapply and lapply

Lists and lapply

```
> a<-c("a","b","c","d")
> b<-c(1,2,3,4,4,3,2,1)
> c<-c(T,T,F)
```

character information
numeric information
logical information

```
> list.object<-list(a,b,c)
> class(list.object)
[1] "list"
```

combine into a list

```
> list.object
[[1]]
[1] "a" "b" "c" "d"

[[2]]
[1] 1 2 3 4 4 3 2 1

[[3]]
[1]  TRUE TRUE FALSE
```

```
> lapply(list.object,class)
[[1]]
[1] "character"

[[2]]
[1] "numeric"

[[3]]
[1] "logical"
```

Looking for runs of numbers within vectors

Function rle (‘run length encoding’)

```
> (poisson<-rpois(150,0.7))
  [1] 1 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 2 1 0 0 1 1 0 0 1 2 0 1 2 0 1 1 0
0 0 3 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 1
 [54] 0 0 0 2 1 1 0 0 1 1 0 1 0 0 0 2 1 0 2 2 1 1 0 1 2 1 0 0 0 0 0 1 1 2
0 2 0 0 2 0 0 0 3 0 2 1 0 3 1 0 1 1 1
[107] 0 2 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 2 1 1 2 0 1 0 0 1 0 1 0 0
0 0 1 1 1 0 0 0 1 1

> rle(poisson)
Run Length Encoding
  lengths: int [1:91] 1 4 1 2 2 1 2 3 1 1 ...
  values : num [1:91] 1 0 1 0 1 0 1 0 1 2 ...

> max(rle(poisson)[[1]])
[1] 6

> which(rle(poisson)[[1]]==6)
[1] 78

> rle(poisson)[[2]][78]
[1] 0
```

list [[1]], [[2]]

Max repeat length

Position in [[1]] … and also in [[2]]

Value [78] in [[2]]

Looking for runs of numbers within vectors

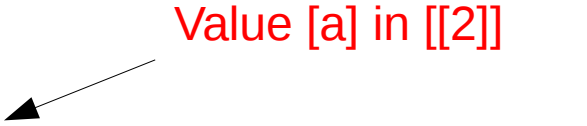
Function rle (‘run length encoding’)

```
> (poisson<-rpois(150,0.7))
  [1] 1 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 2 1 0 0 1 1 0 0 1 2 0 1 2 0 1 1 0
0 0 3 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 1
 [54] 0 0 0 2 1 1 0 0 1 1 0 1 0 0 0 2 1 0 2 2 1 1 0 1 2 1 0 0 0 0 0 1 1 2
0 2 0 0 2 0 0 0 3 0 2 1 0 3 1 0 1 1 1
[107] 0 2 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 2 1 1 2 0 1 0 0 1 0 1 0 0
0 0 1 1 1 0 0 0 1 1
```

In a function:

```
> run.and.value<-function (x) {
+
+     a<- max(rle(poisson)[[1]])
+     b<-rle(poisson)[[2]][which(rle(poisson)[[1]] == a)]
+
+     cat("length = ",a," value = ",b, "\n")}

> poisson<-rpois(150,0.7)
> run.and.value(poisson)
length =  6  value =  0
```



Saving Data Produced within R to Disc

Export a vector into a file (single column)

```
> poisson<-rpois(150,0.7)
> write(poisson,"/home/michael/ownCloud/BioInfo/poisson.txt",1)
```

Saving Data Produced within R to Disc

Export a vector into a file (single column)

```
> poisson<-rpois(150,0.7)
> write(poisson,"/home/michael/ownCloud/BioInfo/poisson.txt",1)
```

Export a table or a matrix of numbers to file:

```
> xmat<-matrix(rpois(100000,0.75),nrow=1000)
> write.table(xmat,"/home/michael/ownCloud/BioInfo/table.txt",
col.names=F,row.names=F)
```

Saving Data Produced within R to Disc

Export a vector into a file (single column)

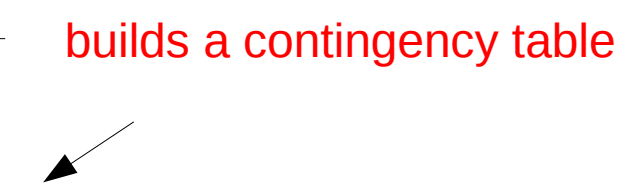
```
> poisson<-rpois(150,0.7)
> write(poisson,"/home/michael/ownCloud/BioInfo/poisson.txt",1)
```

Export a table or a matrix of numbers to file:

```
> xmat<-matrix(rpois(100000,0.75),nrow=1000)
> write.table(xmat,"/home/michael/ownCloud/BioInfo/table.txt",
col.names=F,row.names=F)

> xmat.table<-table(xmat)
> xmat.table
xmat
    0     1     2     3     4     5     6     7
47289 35235 13464  3284   621    95    11     1

> write.table(xmat.table,"/home/michael/ownCloud/BioInfo/
xmattable.txt",col.names=F,row.names=F)
```



Saving Data Produced within R to Disc

Export a vector into a file (single column)

```
> poisson<-rpois(150,0.7)
> write(poisson,"/home/michael/ownCloud/BioInfo/poisson.txt",1)
```

Export a table or a matrix of numbers to file:

```
> xmat<-matrix(rpois(100000,0.75),nrow=1000)
> write.table(xmat,"/home/michael/ownCloud/BioInfo/table.txt",
col.names=F,row.names=F)

> xmat.table<-table(xmat)
> xmat.table
xmat
    0     1     2     3     4     5     6     7
47289 35235 13464  3284   621    95    11     1

> write.table(xmat.table,"/home/michael/ownCloud/BioInfo/
xmattable.txt",col.names=F,row.names=F)

> write.table(unclass(xmat.table),"/home/michael/ownCloud/
BioInfo/xmattableu.txt",col.names=F,row.names=F)
```