The R Book

Chapter 2: Essentials of the R Language
Session 13

Testing for Equality

rounding differences problematic when testing equality of two numbers

```
> x <- sqrt(2)
> x * x == 2
[1] FALSE

> x*x-2
[1] 4.440892e-16
```

Sets: union, intersect and setdiff

```
> setA<-c("a", "b", "c", "d", "e")
> setB<-c("d", "e", "f", "g")
```

three essential functions

Union

```
> union(setA,setB)
[1] "a" "b" "c" "d" "e" "f" "g"
```

Intersection

```
> intersect(setA,setB)
[1] "d" "e"
```

Difference (material in the first, that is not in the second set)

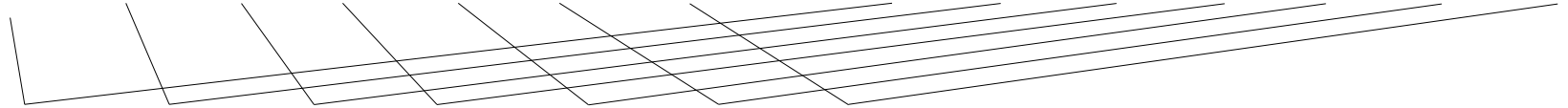
```
> setdiff(setA,setB)
[1] "a" "b" "c"

> setdiff(setB,setA)
[1] "f" "g"
```

Sets: union, intersect and setdiff

```
>all(c(setdiff(setA,setB),intersect(setA,setB),setdiff(setB,setA))==
+ union(setA,setB))

[1] TRUE
```

statements on the left and right side of the equation as a single entry

```
> c(setdiff(setA,setB),intersect(setA,setB),setdiff(setB,setA))==
+ union(setA,setB)

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

> c("a","b","c","d","e","f","g")==c("a","b","c","d","e","f","g")
```



```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Built-in function:

```
> setequal(c(setdiff(setA,setB),intersect(setA,setB),
+ setdiff(setB,setA)),union(setA,setB))

[1] TRUE
```

Sets: union, intersect and setdiff

« %in% » for comparing sets:

```
> setA<-c("a", "b", "c", "d", "e")
> setB<-c("d", "e", "f", "g")

> setA %in% setB
[1] FALSE FALSE FALSE  TRUE  TRUE

setB %in% setA ????

> all(setA[setA %in% setB] == intersect(setA,setB))
[1] TRUE
```

Testing and Coercing in R

Testing / setting object-types

Type	Testing	Coercing
Array	is.array	as.array
Character	is.character	as.character
Complex	is.complex	as.complex
Dataframe	is.data.frame	as.data.frame
Double	is.double	as.double
Factor	is.factor	as.factor
List	is.list	as.list
Logical	is.logical	as.logical
Matrix	is.matrix	as.matrix
Numeric	is.numeric	as.numeric
Raw	is.raw	as.raw
Time series (ts)	is.ts	as.ts
Vector	is.vector	as.vector

Testing and Coercing in R

- Factor levels can be coerced to numbers
- Numbers can be coerced into characters
- non-numeric characters cannot be coerced into numbers

```
> as.numeric(factor(c("a","b","c")))
[1] 1 2 3

> as.numeric(c("a","b","c"))
[1] NA NA NA
Message d'avis :
NAs introduits lors de la conversion automatique

> as.numeric(c("a","4","c"))
[1] NA  4 NA
Message d'avis :
NAs introduits lors de la conversion automatique
```

NOT operator !

in functions to return an error message
Example : function to calculate geometric means :

a) input must be numeric test with !is.numeric

```
> geometric<-function(x){
+ if(!is.numeric(x)) stop ("Input must be numeric")
+ exp(mean(log(x))) }

> geometric(c(3,300,30,300))
[1] 53.34838

> geometric(c(a,b,c))
Erreur dans geometric(c(a, b, c)) : Input must be numeric
```

NOT operator !

in functions to return an error message
Example : function to calculate geometric means :

a) input must be numeric test with !is.numeric

```
> geometric<-function(x){
+ if(!is.numeric(x)) stop ("Input must be numeric")
+ exp(mean(log(x))) }

> geometric(c(3,300,30,300))
[1] 53.34838

> geometric(c(a,b,c))
Erreur dans geometric(c(a, b, c)) : Input must be numeric
```

b) input numeric and > 0

```
> geometric<-function(x){
+ if(!is.numeric(x)) stop ("Input must be numeric")
+ if(min(x)<=0) stop ("Input must be greater than zero")
+ exp(mean(log(x))) }

> geometric(c(2,3,0,4))
Erreur dans geometric(c(2, 3, 0, 4)) : Input must be greater than zero

> geometric(c(10,1000,10,1,1))
[1] 10
```

Dates and Times in R

```
> Sys.time()
[1] "2015-01-13 18:12:08 CET"
```

extract the date using substr

```
> substr(as.character(Sys.time()),1,10)
[1] "2015-01-13"

> substr(as.character(Sys.time()),12,19)
[1] "18:29:27"

> unclass(Sys.time())
[1] 1421170169 number of seconds since 1 January 1970
               ( … when did I prepare this slide????)
```

Dates and Times in R

two basic classes of date/times

POSIXct

- **number of seconds since the beginning of 1970 as a numeric vector**

```
> unclass(as.POSIXct(Sys.time()))
[1] 1421248616
```

POSIXlt

list of vectors (human-readable forms), representing time since 1900

- **Seconds**
- **Minutes**
- **Hours**
- **Days**
- **Months**
- **Years**

```
> date<- as.POSIXlt(Sys.time())
> date$wday
[1] 3

> date$yday
[1] 13
```

Dates and Times in R

two basic classes of date/times

POSIXct

- **number of seconds since the beginning of 1970 as a numeric vector**

```
> unclass(as.POSIXct(Sys.time()))
[1] 1421248616
```

POSIXlt

list of vectors (human-readable forms), representing time since 1900

- **Seconds**
- **Minutes**
- **Hours**
- **Days**
- **Months**
- **Years**

```
> unclass(as.POSIXlt(Sys.time()))
```

- **?????**

```
> unlist(unclass(as.POSIXlt(Sys.time())))
```

sec	min	hour	mday	mon	year	wday	yday	isdst
33.33315	19.00000	16.00000	14.00000	0.00000	115.00000	3.00000	13.00000	0.00000

Calculations with dates and times

Possible calculations with dates and times:

- **time + number**
- **time – number**
- **time1 – time2**
- **time1 ‘logical operation’ time2 (==, !=, <, <=, >, >=)**

convert dates and times into POSIXlt objects before doing calculations

```
> y2<-as.POSIXlt("2003-10-22")
> y1<-as.POSIXlt("2005-10-22")
> y2
[1] "2003-10-22"
> y1
[1] "2005-10-22"
> y1-y2
Time difference of 731 days
```

you cannot add two dates

```
> y1+y2
Erreur dans `+.POSIXt`(y1, y2) :
  l'opérateur binaire '+' n'est pas défini pour les objets "POSIXt"
```

Calculations with dates and times

Possible calculations with dates and times:
- **time + number**
- **time – number**
- **time1 – time2**
- **time1 ‘logical operation’ time2 (==, !=, <, <=, >, >=)**

convert dates and times into POSIXlt objects before doing calculations

```
> y2<-as.POSIXlt("2003-10-22")
> y1<-as.POSIXlt("2005-10-22")
> y2
[1] "2003-10-22"
> y1
[1] "2005-10-22"
> y1-y2
Time difference of 731 days
```

Convention: date separation by « - », time by « : »

```
> y3<-as.POSIXlt("2005-10-22 09:30:59")
> y4<-as.POSIXlt("2005-10-22 12:45:06")
> y4-y3
Time difference of 3.235278 hours
```

The difftime function to calculate time differences

```
> y4<-c("2005-10-22 12:45:06")
> y4
[1] "2005-10-22 12:45:06"

> y3<-c("2005-10-22 09:30:59")
> y3
[1] "2005-10-22 09:30:59"

> y4-y3
Erreur dans y4 - y3 : argument non numérique pour un opérateur binaire

> difftime(y4,y3)
Time difference of 3.235278 hours

> as.numeric(difftime(y4,y3))
[1] 3.235278

> ISOdate(2005,10,22,12,45,06)-ISOdate(2005,10,22,09,30,59)
Time difference of 3.235278 hours

> as.difftime(c("0:3:20", "11:23:15"))
Time differences in mins
[1]   3.333333 683.250000
attr(,"tzone")
[1] ""
```

convert character stings into difftime objects

The strptime function ‘strip a date’ out of a character string

Time specifications :

%a Abbreviated weekday name
%A Full weekday name
%b Abbreviated month name
%B Full month name
%c Date and time, locale-specific
%d Day of the month as decimal number (01–31)
%H Hours as decimal number (00–23) on the 24-hour clock
%I Hours as decimal number (01–12) on the 12-hour clock
%j Day of year as decimal number (001–366)
%m Month as decimal number (01–12)
%M Minute as decimal number (00–59)
%p AM/PM indicator in the locale
%S Second as decimal number (00–61, allowing for two ‘leap seconds’)
%U Week of the year (00–53) using the first Sunday as day 1 of week 1
%w Weekday as decimal number (0–6, Sunday is 0)
%W Week of the year (00–53) using the first Monday as day 1 of week 1
%x Date, locale-specific
%X Time, locale-specific
%Y Year with century
%Z Time zone as a character string (output only)

The strptime function ‘strip a date’ out of a character string

Example :

```
> excel.dates <- c("27/02/2004", "27/02/2005", "14/01/2003",
+ "28/06/2005", "01/01/1999")

> strptime(excel.dates,format="%d/%m/%Y")
[1] "2004-02-27" "2005-02-27" "2003-01-14" "2005-06-28" "1999-01-01"


> other.dates<- c("01janvier99", "02janvier05", "31mars04",
"30juillet05")

> strptime(other.dates,format="%d%b%y")
[1] "1999-01-01" "2005-01-02" "2004-03-31" "2005-07-30"
```

The strptime function time calculations with a dataframe

```
> times<-read.table("/perso/mlang/Documents/BioInfo/data/times.txt",
header=T)
> times
????

> attach(times)
> paste(hrs,min,sec,sep=":")
 [1] "2:23:6"  "3:16:17" "3:2:56"  "2:45:0"  "3:4:42"  "2:56:25" "3:12:28"
 [8] "1:57:12" "2:22:22" "1:42:7"  "2:31:17" "3:15:16" "2:28:4"  "1:55:34"
[15] "2:17:7"  "1:48:48"

> duration<-as.difftime(paste(hrs,min,sec,sep=":"))
> duration
Time differences in hours
 [1] 2.385000 3.271389 3.048889 2.750000 3.078333 2.940278 3.207778 1.953333
 [9] 2.372778 1.701944 2.521389 3.254444 2.467778 1.926111 2.285278 1.813333
attr(,"tzone")
[1] ""

> tapply(duration,experiment, mean)
       A        B
2.829375 2.292882
```

Calculating time differences between the rows of a dataframe

... using subscripts

```
> duration[1:15]-duration[2:16]   between successive rows
Time differences in hours
 [1] -0.8863889  0.2225000  0.2988889 -0.3283333  0.1380556 -0.2675000
 [7]  1.2544444 -0.4194444  0.6708333 -0.8194444 -0.7330556  0.7866667
[13]  0.5416667 -0.3591667  0.4719444

Import differences into the dataframe, beware :

> length(duration[1:15]-duration[2:16]) one entry less than duration
[1] 15
> length(duration)
[1] 16

> diffs<-c(duration[1:15]-duration[2:16],NA)     add one NA
> diffs
 [1] -0.8863889  0.2225000  0.2988889 -0.3283333  0.1380556 -0.2675000
 [7]  1.2544444 -0.4194444  0.6708333 -0.8194444 -0.7330556  0.7866667
[13]  0.5416667 -0.3591667  0.4719444          NA

> times$diffs<-diffs import into 'times' dataframe
> times
????
```