Lesson Scripts

Victor Wildner

4/1/2020

R Tips

Number as Integer:

```
1L ## L Sufix
```

```
## [1] 1
```

Access Atributes:

```
x <- "hello"
attributes(x)
```

```
## NULL
```

Set vector:

Concatenate:

```
a <- c("a", "b", "c")
a
```

```
## [1] "a" "b" "c"
```

Convert to something:

```
x <- 0:6
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6
```

```
as.logical(x)
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6"
```

Lists:

```
x <- list(1, "a", TRUE, 1 + 4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

Matrices:

```
m <- matrix(1:6, nrow = 2, ncol = 3)
m
```

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

```
dim(m)
```

```
## [1] 2 3
```

Matrices:

```
m <- 1:10
dim(m) <-c(2, 5)
m
```

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

cbiding and rbiding:

```
x <- 1:3
y <- 10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

Factors:

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

```
table(x)
```

```
## x
##  no yes
##   2   3
```

```
unclass(x)
```

```
## [1] 2 2 1 2 1
## attr(,"levels")
## [1] "no"  "yes"
```

```r
x <- factor(c("yes", "yes", "no", "yes", "no"),
 levels = c("yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: yes no
```

Missing Values:

```r
x <- c(1, 2, NA, 10, 3)
is.na(x)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

```r
is.nan(x)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```r
y <- c(1, 2, NaN, NA, 4)
is.na(y)
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE
```

```r
is.nan(y)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

Data Frames:

```r
x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
x
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```r
nrow(x)
```

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

```r
ncol(x)
```

```
## [1] 2
```

Data Types - Name Attributes:

```r
x <- 1:3
names(x)
```

```
## NULL
```

```r
names(x) <- c("foo", "bar", "norf")
x
```

```
##  foo  bar norf
##    1    2    3
```

```r
y <- list(a = 1, b = 2, c = 3)
y
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] 3
```

```
m <- matrix(1:4, nrow = 2, ncol = 2)
dimnames(m) <- list(c("a", "b"), c("c", "d"))
m
```

```
##   c d
## a 1 3
## b 2 4
```

Reading Data Functions:

- **read.table**, **read.csv**, for reading tabular data
- **readLines**, for reading lines of a text file
- **source**, for reading in R code files (inverse of dump)
- **dget**, for reading in R code files (inverse of dput)
- **load**, for reading in saved workspaces
- **unserialize**, for reading single R objects in binary form

Writing Data:

- **write.table**
- **writeLines**
- **dump**
- **dput**
- **save**
- **serialize**

Reading Data read.table:

- **file**, the name of a file, or a connection
- **header**, logical indicating if the file has a header line
- **sep**, a string indicating how the columns are separated
- **colClasses**, a character vector indicating the class of each column in the dataset
- **nrows**, the number of rows in the dataset
- **comment.char**, a character string indicating the comment character
- **skip**, the number of lines to skip from the beginning
- **stringsAsFactors**, should character variables be coded as factors? (default=TRUE)

read.table

```
## data <- read.table("foo.txt")
```

read.csv

```
## data <- read.csv("foo.csv")
```

Read Large Datasets

```
## comment.char = ""
## initial <- read.table("datatable.txt", nrows = 100)
## classes <- sapply(initial, class)
## tabAll <- read.table("datatable.txt", colClasses = Classes)
```

Estimate Memory Requirements

Supose you have 1.500.000 rows and 120 cols: 1.500.00 X 120 X 8 bytes/numeric = 1440000000 bytes 1440000000 bytes/2^(20) bytes/MB = 1.373.29 MB ~= 1.34 GB

Deconstruct R Objects

dput-ting R Objects dput-ting R Objects (one object at a time)

```
y <- data.frame(a = 1, b = "a")
dput(y)
```

```
## structure(list(a = 1, b = structure(1L, .Label = "a", class = "factor")), .Names = c("a",
## "b"), row.names = c(NA, -1L), class = "data.frame")
```

```
# Put into a file
dput(y, file="y.R")
new.y <- dget("y.R")
new.y
```

```
##   a b
## 1 1 a
```

dumping R Objects (various objects)

```
x <- "foo"
y <- data.frame(a = 1, b = "a")
dump(c("x", "y"), file = "data.R")
rm(x, y)
source("data.R")
y
```

```
##   a b
## 1 1 a
```

```
x
```

```
## [1] "foo"
```

Setting a Connection

```
con <- file("y.R")
```

Removing NA Values

```
x <- c(1, 2, NA, 4, NA, 5)
bad <- is.na(x) # searches for missing elements
bad
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE FALSE
```

```
x[!bad]
```

```
## [1] 1 2 4 5
```

```
y <- c("a", "b", NA, "d", NA, "f")
good <- complete.cases(x, y) # searches for non missing vectors
good
```

```
## [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE
```

```
x[good]
```

```
## [1] 1 2 4 5
```

```
y[good]
```

```
## [1] "a" "b" "d" "f"
```