SCS 2211 - Laboratory II

Mr. D. T. Bamunuarachchi – dtb@ucsc.cmb.ac.lk
Ms. Piyumi Seneviratne – pws@ucsc.cmb.ac.lk

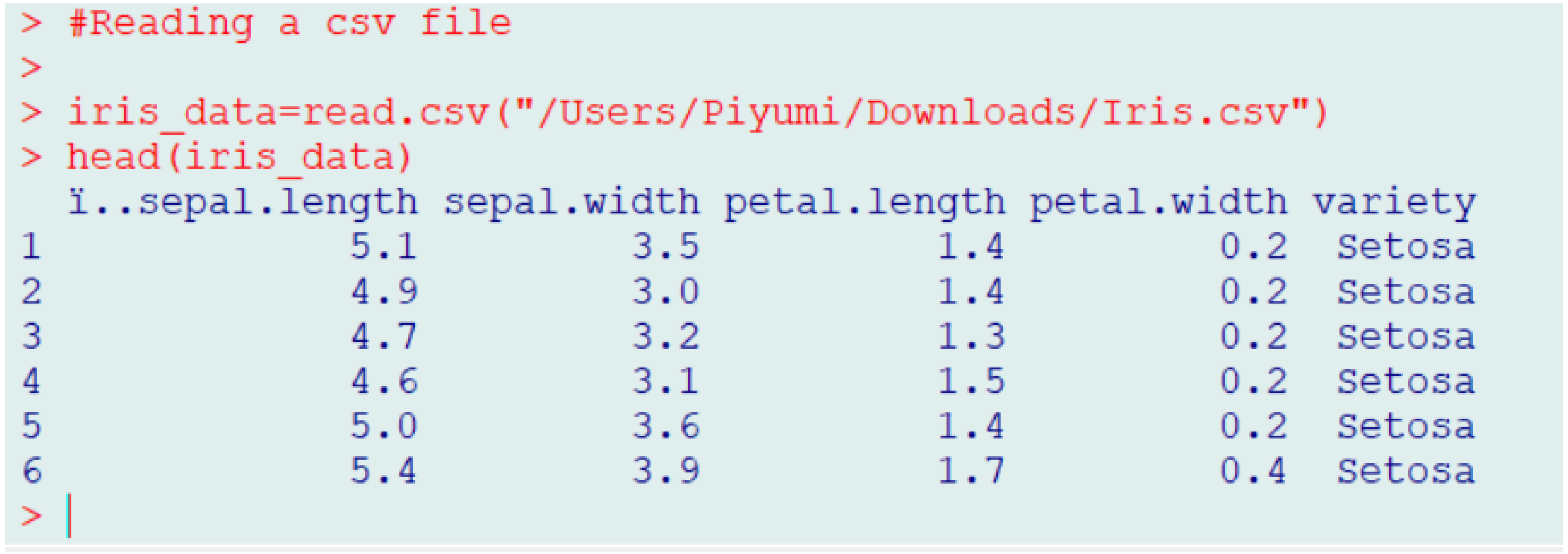
Importing Data

- We can import data into R using several formats
- TXT → name=read.table(“filename.txt”,header=TRUE)
- CSV → name=read.csv(“filename.csv”,header=TRUE)

Reading Data into R

- 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

Importing Data



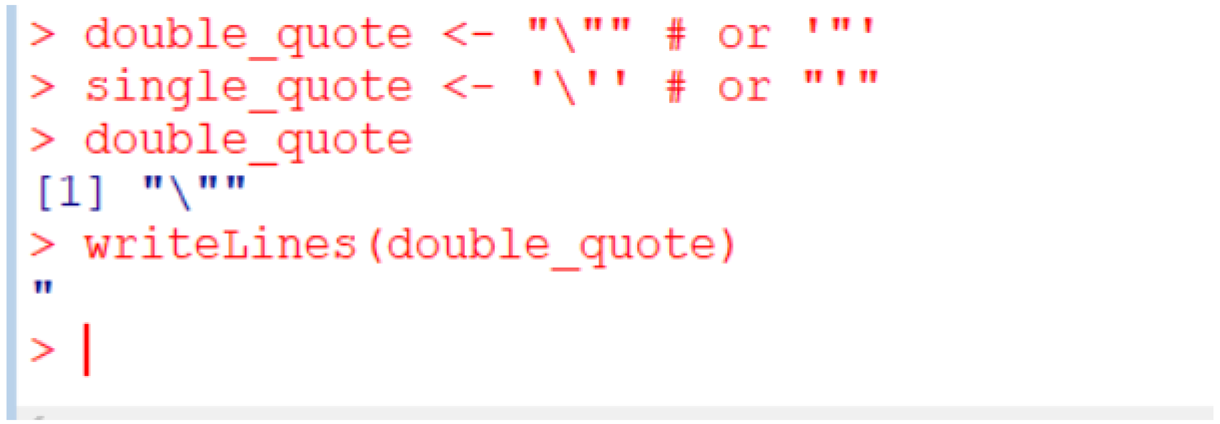
```
> #Reading a csv file
>
> iris_data=read.csv("/Users/Piyumi/Downloads/Iris.csv")
> head(iris_data)
  ï..sepal.length sepal.width petal.length petal.width variety
1             5.1         3.5          1.4         0.2  Setosa
2             4.9         3.0          1.4         0.2  Setosa
3             4.7         3.2          1.3         0.2  Setosa
4             4.6         3.1          1.5         0.2  Setosa
5             5.0         3.6          1.4         0.2  Setosa
6             5.4         3.9          1.7         0.4  Setosa
> |
```

Paths And Directory Names

- In windows system:

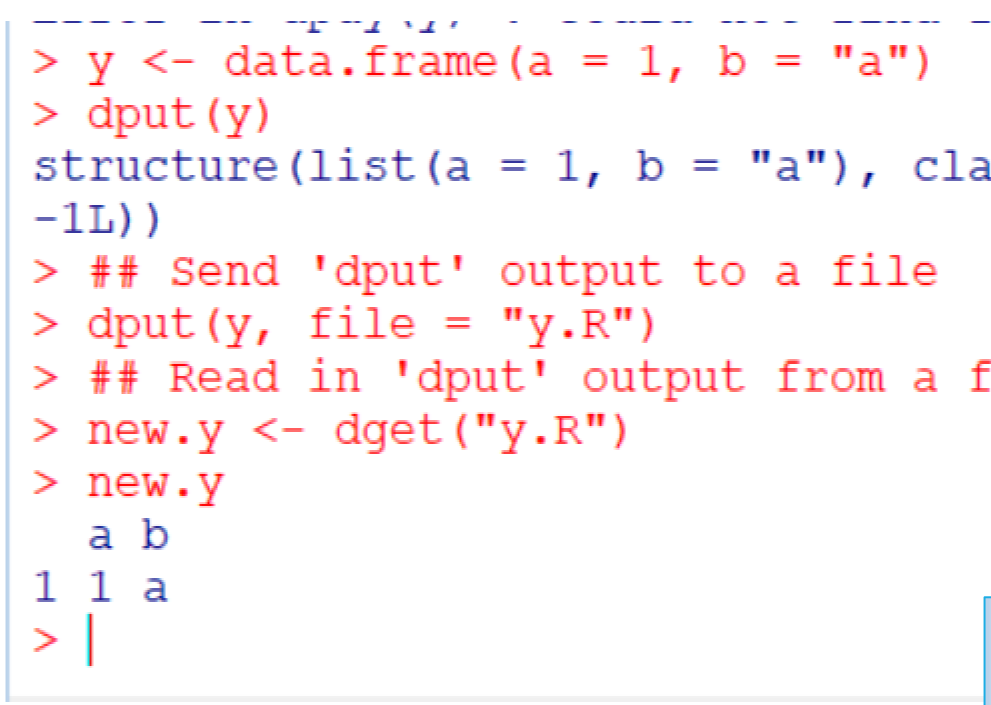
  "\Users\Piyumi\Downloads\Iris.csv“
- But in previous slide (slide No: 04) we have used

  “/Users/Piyumi/Downloads/Iris.csv”
- In R, we use forward slashes for the directories because backslashes are actually used for escape characters.

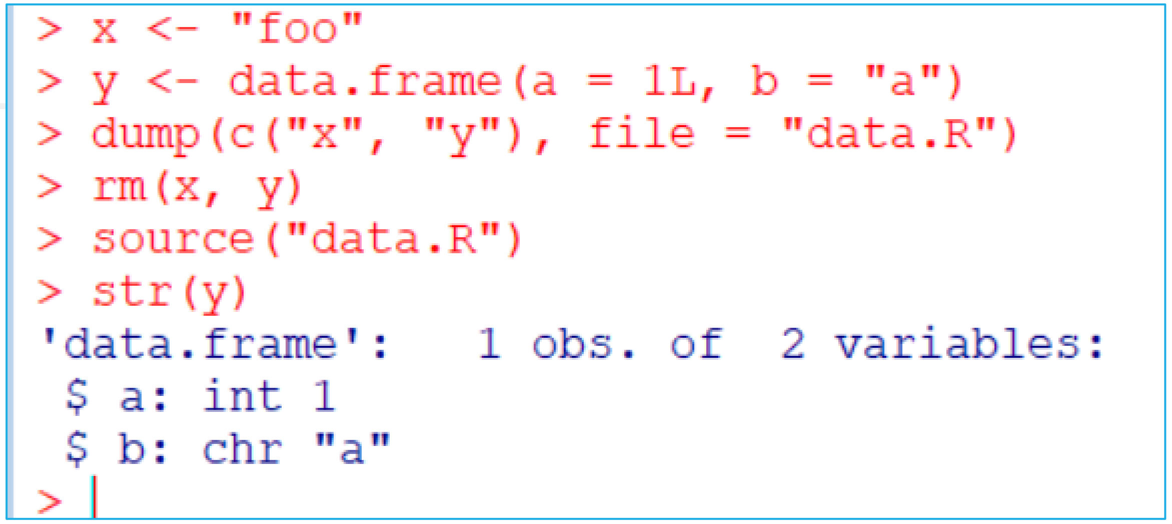


```
> double_quote <- "\"" # or '"'
> single_quote <- '\'' # or "'"
> double_quote
[1] "\""
> writeLines(double_quote)
"
> |
```

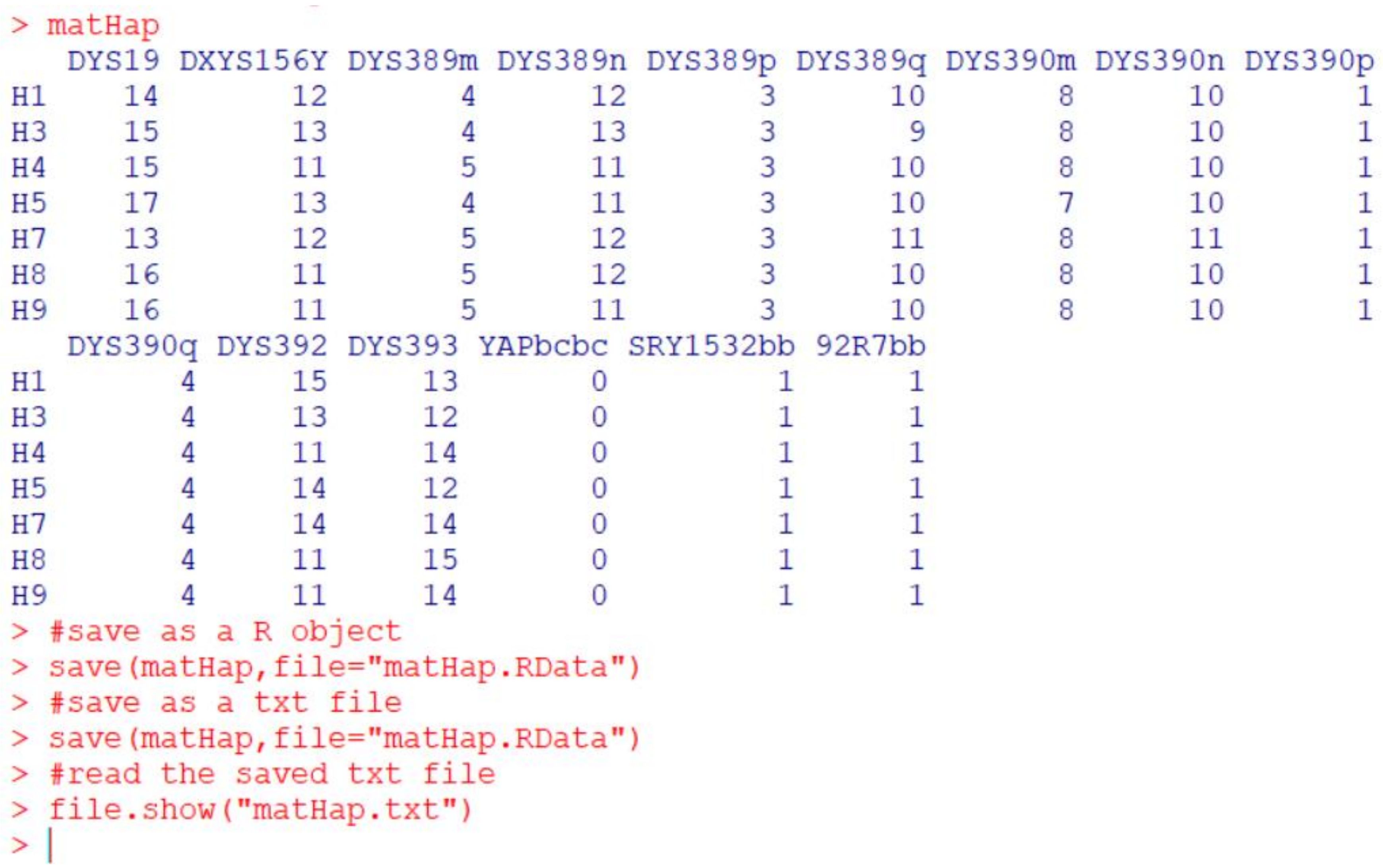
dput() and dump()



```
> y <- data.frame(a = 1, b = "a")
> dput(y)
structure(list(a = 1, b = "a"), class = "data.frame", row.names = c(NA,
-1L))
> ## Send 'dput' output to a file
> dput(y, file = "y.R")
> ## Read in 'dput' output from a file
> new.y <- dget("y.R")
> new.y
  a b
1 1 a
> 
```



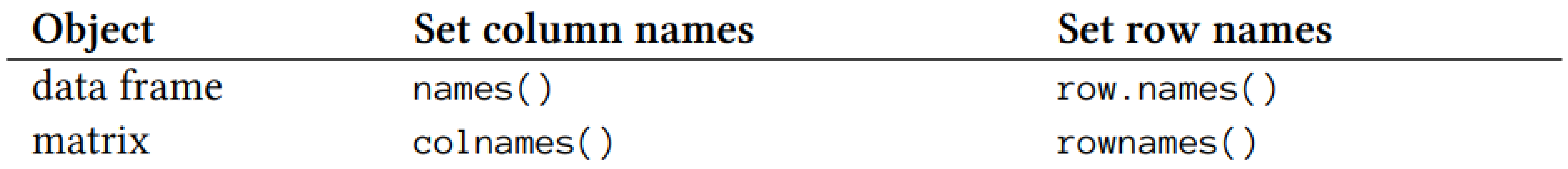
```
> x <- "foo"
> y <- data.frame(a = 1L, b = "a")
> dump(c("x", "y"), file = "data.R")
> rm(x, y)
> source("data.R")
> str(y)
'data.frame':   1 obs. of  2 variables:
 $ a: int 1
 $ b: chr "a"
> 
```



```
> matHap
   DYS19 DXYS156Y DYS389m DYS389n DYS389p DYS389q DYS390m DYS390n DYS390p
H1    14       12       4      12       3      10       8      10       1
H3    15       13       4      13       3       9       8      10       1
H4    15       11       5      11       3      10       8      10       1
H5    17       13       4      11       3      10       7      10       1
H7    13       12       5      12       3      11       8      11       1
H8    16       11       5      12       3      10       8      10       1
H9    16       11       5      11       3      10       8      10       1
   DYS390q DYS392 DYS393 YAPbcbc SRY1532bb 92R7bb
H1       4     15     13       0         1      1
H3       4     13     12       0         1      1
H4       4     11     14       0         1      1
H5       4     14     12       0         1      1
H7       4     14     14       0         1      1
H8       4     11     15       0         1      1
H9       4     11     14       0         1      1
> #save as a R object
> save(matHap,file="matHap.RData")
> #save as a txt file
> save(matHap,file="matHap.RData")
> #read the saved txt file
> file.show("matHap.txt")
> |
```

Working with data - Names

- R objects can have names.
- Matrices and data frames can have both column and row names.



Object	Set column names	Set row names
data frame	`names()`	`row.names()`
matrix	`colnames()`	`rownames()`

Working with data - Missing Values

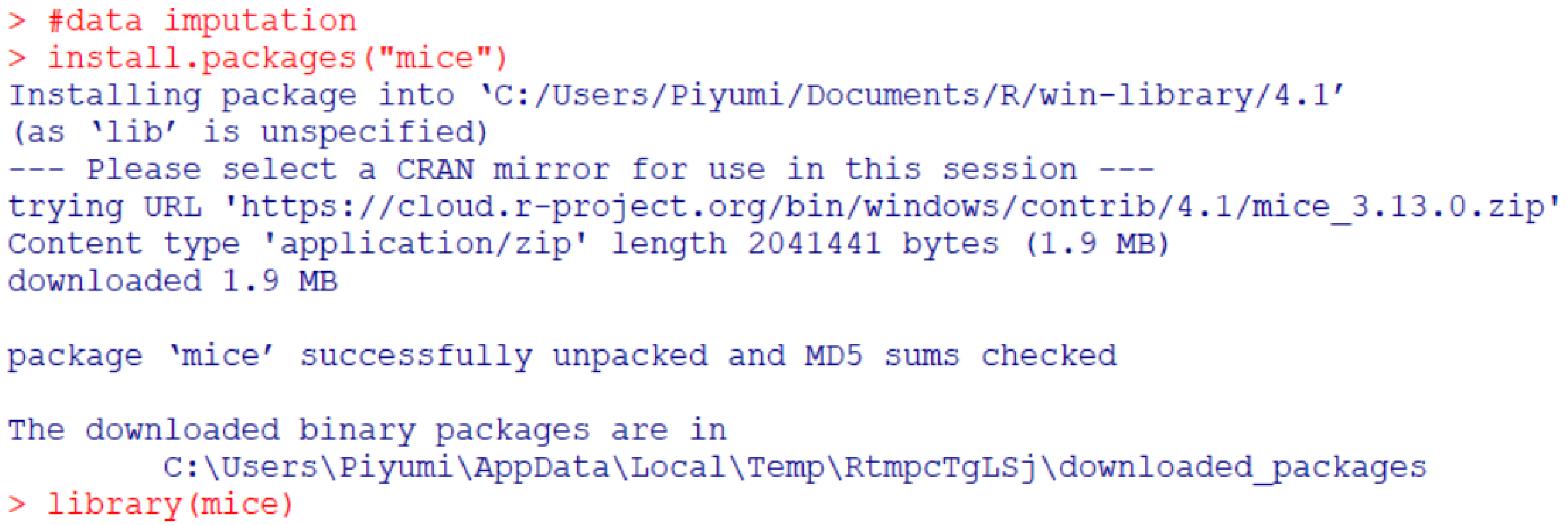
- NA
- NAN
- NA values have a class also, so there are integer NA, character NA, etc.
- We can use complete.cases(), na.omit (), na.rm() to clean missing values.

Working with data – Replacing Missing Values

- NA
- NAN
- NA values have a class also, so there are integer NA, character NA, etc.
- We can use complete.cases(), na.omit (), na.rm() to clean missing values.

Working with data – Replacing Missing Values

- MICE

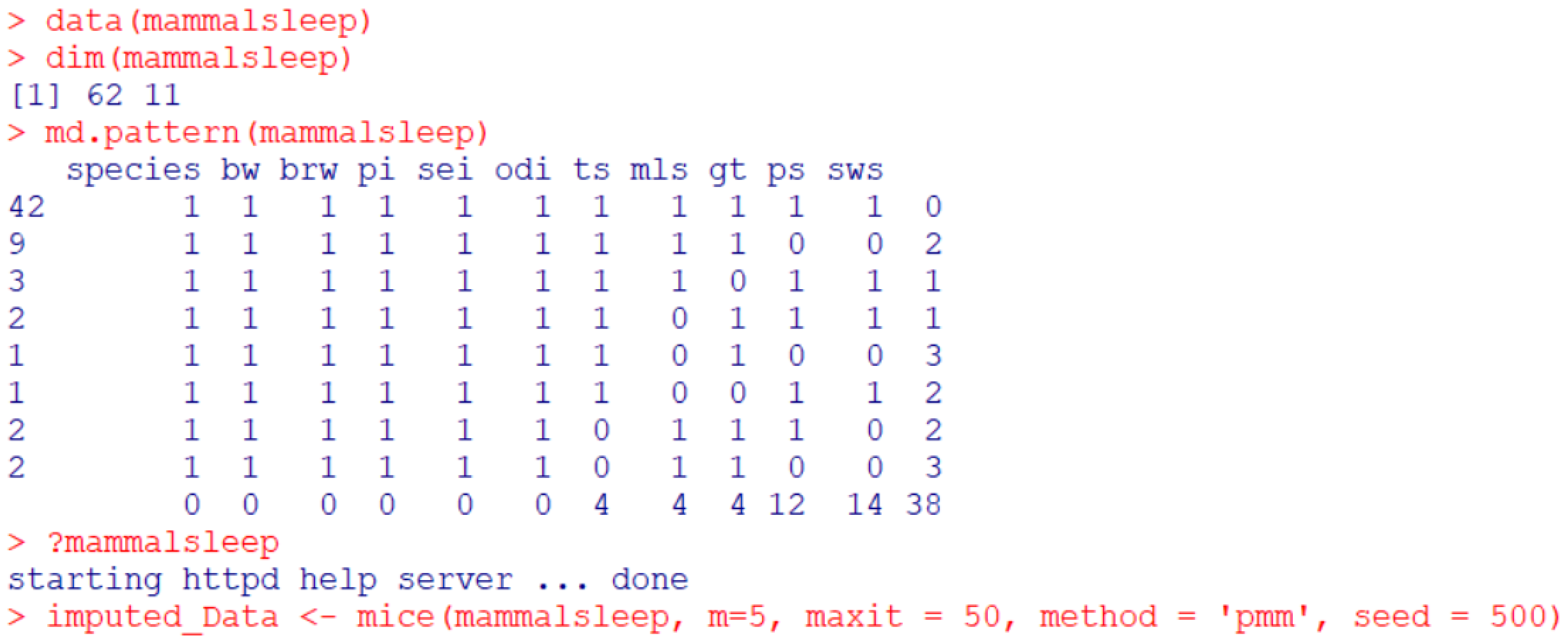


```
> #data imputation
> install.packages("mice")
Installing package into ‘C:/Users/Piyumi/Documents/R/win-library/4.1’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/bin/windows/contrib/4.1/mice_3.13.0.zip'
Content type 'application/zip' length 2041441 bytes (1.9 MB)
downloaded 1.9 MB

package ‘mice’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Piyumi\AppData\Local\Temp\RtmpcTgLSj\downloaded_packages
> library(mice)
```

Working with data – Replacing Missing Values



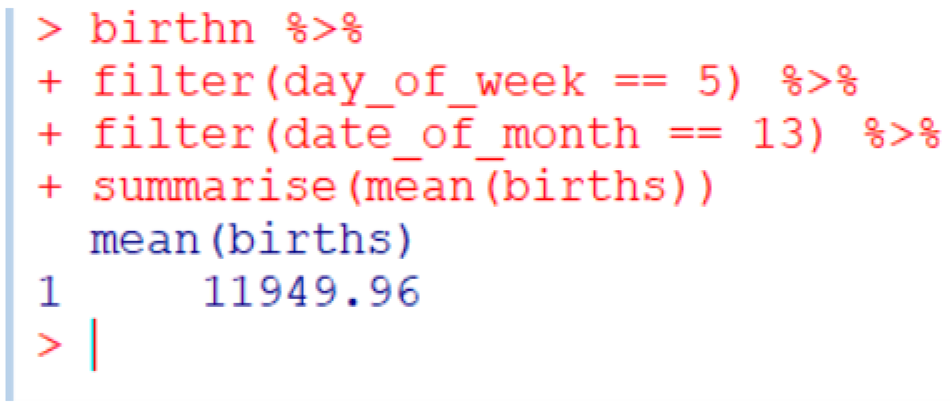
```
> data(mammalsleep)
> dim(mammalsleep)
[1] 62 11
> md.pattern(mammalsleep)
   species bw brw pi sei odi ts mls gt ps sws
42       1  1   1  1   1   1  1   1  1  1   1  0
9        1  1   1  1   1   1  1   1  1  0   0  2
3        1  1   1  1   1   1  1   1  0  1   1  1
2        1  1   1  1   1   1  1   0  1  1   1  1
1        1  1   1  1   1   1  1   0  1  0   0  3
1        1  1   1  1   1   1  1   0  0  1   1  2
2        1  1   1  1   1   1  0   1  1  1   0  2
2        1  1   1  1   1   1  0   1  1  0   0  3
         0  0   0  0   0   0  4   4  4 12  14 38
> ?mammalsleep
starting httpd help server ... done
> imputed_Data <- mice(mammalsleep, m=5, maxit = 50, method = 'pmm', seed = 500)
```

Working with data – Assignment 1

1. Download the resource file given in lms.
2. Read all the data in births into “birth_new” and save it as “birth_new.Rdata”.
3. Select the births that happen only on Saturday into sat1 and display 5 rows of the resulting data.
4. Use the “dplyr” package to follow next steps.
5. Filter the births happen on “ day_of_week == 6” into sat2.
6. Try using “Sat2 <- birthn %>% filter(day_of_week == 6)”. What did you get?
7. Format your data output received in step 3 using “as_tibble()”.
8. Group the births_new by the day_of_week.
9. Get the group means.
10. Sort the result of step 9.
11. Get the summary.
12. Repeat step 8,9,10 in a nested operation.

Working with data

- x %>% f(y) is equivalent to just executing f(x,y)
- If we need to execute a sequence of functions: h(g(f(x,y),z),m)
- We can use x %>% f(y) %>% g(z) %>% h(m) that gives the same answer.
- To find out the average of Friday 13th births:



```
> birthn %>%
+ filter(day_of_week == 5) %>%
+ filter(date_of_month == 13) %>%
+ summarise(mean(births))
  mean(births)
1     11949.96
> |
```