Module 3: Creating, Importing, & Exporting Data

1 Creating Data

1.1 Vectors

To create a vector of values in R you use the concatenate "c" function.

```
# numeric vector
nv <- c(2, 3, 4, 2)
print(nv)
```

```
## [1] 2 3 4 2
```

```
# character vector
cv <- c("put", "some", "words", "here")
print(cv)
```

```
## [1] "put"   "some"  "words" "here"
```

```
# logical vector
lv <- c(T, F, F, T)
print(lv)
```

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

1.2 Matrices

The main methods to create matrices are the "matrix" function and the use of the "cbind" or "rbind" functions.

The matrix function takes a vector (creatd with the "c" function) and then provides information on how many rows and/or columns to divide it into.

```
mat <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
print(mat)
```

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

If you supply a number or rows that doesn't evenly divide the elements of the vector and error will appear.

```
matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 2)
```

```
## Warning: data length [9] is not a sub-multiple or multiple of the number
## of rows [2]
```

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

Using cbind or rbind to bind columns or rows will also work. If the vectors you are binding are of the same class (numeric, character, logical), the resulting matrix will also be that class. However, if the vectors differ the matrix will defualt to the most compatible type.

```
mat2 <- cbind(nv, cv, lv)
print(mat2)
```

```
##      nv  cv      lv
## [1,] "2" "put"   "TRUE"
## [2,] "3" "some"  "FALSE"
## [3,] "4" "words" "FALSE"
## [4,] "2" "here"  "TRUE"
```

```
mode(mat2)
```

```
## [1] "character"
```

```
mat3 <- cbind(nv, lv)
print(mat3)
```

```
##      nv lv
## [1,]  2  1
## [2,]  3  0
## [3,]  4  0
## [4,]  2  1
```

```
mode(mat3)
```

```
## [1] "numeric"
```

1.3 Data Frames

Most imported data will end up a data frame by default. However, you can create a data frame from other R objects using the "data.frame" function. When making a data frame, the columns need to all be the same length and the rows need to be the same length.

```
names <- c("Fred", "Bob", "Bill", "Jim")
weight <- c(129, 145, 234, 198)
height <- c(64, 68, 72, 70)

data <- data.frame(names, weight, height)
print(data)
```

```
##   names weight height
## 1  Fred    129     64
## 2   Bob    145     68
## 3  Bill    234     72
## 4   Jim    198     70
```

1.4 Lists

```
data2 <- list(name, weight, height)
print(data2)
```

```
## [[1]]
## [1] "module3"
##
## [[2]]
## [1] 129 145 234 198
##
## [[3]]
## [1] 64 68 72 70
```

```
data3 <- list(name, weight, height, data)
```

1.5 Converting between types

```
a <- c("10", "20", "30")
print(a)
```

```
## [1] "10" "20" "30"
```

```
class(a)
```

```
## [1] "character"
```

```
b <- as.numeric(a)
print(b)
```

```
## [1] 10 20 30
```

```
class(b)
```

```
## [1] "numeric"
```

```
c <- as.factor(b)
print(c)
```

```
## [1] 10 20 30
## Levels: 10 20 30
```

```
class(c)
```

```
## [1] "factor"
```

2 Importing Data

Getting your data into R can sometimes be the hardest part. Luckily there are packages and functions to help with this.

2.1 Copy/Paste

If you try to copy and paste some data in from a text file or an Excel spreadsheet and assign it to an object you just end up with a character vector with a single element made up of long string of characters.

```
HaresLynx <- "Year  Hares  Lynx\n1900 30 4\n1901 47.2 6.1\n1902 70.2 9.8\n1903 77.4 35.2\n1904 36.3 59.4\n1
```

```
print(HaresLynx)
```

```
## [1] "Year  Hares  Lynx\n1900 30 4\n1901 47.2 6.1\n1902 70.2 9.8\n1903 77.4 35.2\n1904 36.3 59.4\n1905 20
```

Probably not what you wanted.

To remedy this we use read.table and TextConnection to parse the characters into a data frame

```
HaresLynx2 <- read.table(textConnection(HaresLynx), header = TRUE)
print(HaresLynx2)
```

```
##   Year Hares Lynx
## 1 1900  30.0  4.0
## 2 1901  47.2  6.1
## 3 1902  70.2  9.8
## 4 1903  77.4 35.2
## 5 1904  36.3 59.4
## 6 1905  20.6 41.7
```

2.2 Text Files

Probably the easiest way to import data is via text file. I find CSV files to work the best..

```
sites <- read.csv("data/wk1sites.csv")
print(sites)
```

```
##   siteID        site            lake   lat    lon
## 1      1 Boat Ramp A Henry Hagg Lake 45.48 -123.2
## 2      2 Boat Ramp C Henry Hagg Lake 45.49 -123.2
```

2.3 Database Connection

Connections can be made directly with a database. However, which databases you can connect to is dependent on your operating system. For example, you need to be using Windows to connect to a MS Access database.

One common database is MySQL. We can access MySQL databases using either the "RODBC" library or the "RMySQL" library

```
library(RMySQL)
# make a connection to the database
con <- dbConnect(MySQL(), user = "root", password = "", dbname = "nemesis_development",
    host = "localhost")

# list the tables in that database
dbListTables(con)
```

```
##  [1] "citations"         "comments"          "common_names"
##  [4] "dailyinvaders"     "distributions"     "ecologies"
##  [7] "images"            "journals"          "journals_old"
## [10] "news_items"        "occurrences"       "pages"
## [13] "references"        "references_old"    "regions"
## [16] "schema_migrations" "synonyms"          "taxa"
## [19] "users"
```

```
# run a select query on a table in the database and limit it to 10 records
taxa <- dbGetQuery(con, "select id, binomial, taxa_group from taxa limit 10;")
print(taxa)
```

```
##      id              binomial              taxa_group
## 1  -578           Dasya sp. A     Crustaceans-Copepods
## 2  -577        Dasya sessilis                   Algae
## 3  -576       Pkea yoshizakii                   Algae
## 4  -575 Chondracanthus teedei                   Algae
## 5  -574                  <NA>                    <NA>
## 6  -572 Tricellaria inopinata               Ectoprocts
## 7  -571 Pachycordyle michaeli Coelenterates-Hydrozoans
## 8  -570    Gambusia holbrooki                  Fishes
## 9  -569       Corella inflata                Tunicates
## 10 -568                  <NA>                    <NA>
```

2.4 Other formats (JSON, XML, etc..)

Sometimes you'll find data in other formats. It's worth looking to see if a package has been made to help import that kind of data.

3 Exporting Data

3.1 Text Files

An analog to read.csv is write.csv. You can write just about any data frame into a csv file with this command

```
write.csv(taxa, "taxa.csv")
```

3.2 Database Connection

You can run append, insert, and update queries using a connection to the database. You can also write a data frame into a table useing "dbWriteTable(con,"table name", a.data.frame)

3.3 Other formats (JSON, XML, etc..)

The same specialized R packages you used to import other types of data will likely have a function for writing that type of data from R objects.

3.4 dput function

It is often useful to useful to export an R object into an R statement that can be used to recreate that object elsewhere. For example, say you wanted to ask an R question on http://stackoverflow.com/ and needed to include some small snippit of your data to help explain your issue. Just doing a copynig and pasting will cause formatting issues and it will be difficult for others to usee your data to help you out. However, if we use the "dput" function we can transform an R object into a the command we'd need to recreate that structure.

```
dput(taxa)
```

```
## structure(list(id = c(-578L, -577L, -576L, -575L, -574L, -572L,
## -571L, -570L, -569L, -568L), binomial = c("Dasya sp. A", "Dasya sessilis",
## "Pkea yoshizakii", "Chondracanthus teedei", NA, "Tricellaria inopinata",
## "Pachycordyle michaeli", "Gambusia holbrooki", "Corella inflata",
## NA), taxa_group = c("Crustaceans-Copepods", "Algae", "Algae",
## "Algae", NA, "Ectoprocts", "Coelenterates-Hydrozoans", "Fishes",
## "Tunicates", NA)), .Names = c("id", "binomial", "taxa_group"), row.names = c(NA,
## 10L), class = "data.frame")
```

```
# If I copy the output to that and assign it to an object..

taxanew <- structure(list(id = c(-578L, -577L, -576L, -575L, -574L, -572L, -571L,
    -570L, -569L, -568L), binomial = c("Dasya sp. A", "Dasya sessilis", "Pkea yoshizakii",
    "Chondracanthus teedei", NA, "Tricellaria inopinata", "Pachycordyle michaeli",
    "Gambusia holbrooki", "Corella inflata", NA), taxa_group = c("Crustaceans-Copepods",
    "Algae", "Algae", "Algae", NA, "Ectoprocts", "Coelenterates-Hydrozoans",
    "Fishes", "Tunicates", NA)), .Names = c("id", "binomial", "taxa_group"),
    row.names = c(NA, 10L), class = "data.frame")


# I get a copy of the original object
print(taxanew)
```

```
##      id              binomial               taxa_group
## 1  -578          Dasya sp. A     Crustaceans-Copepods
## 2  -577        Dasya sessilis                   Algae
## 3  -576       Pkea yoshizakii                   Algae
## 4  -575 Chondracanthus teedei                   Algae
## 5  -574                  <NA>                    <NA>
## 6  -572 Tricellaria inopinata               Ectoprocts
## 7  -571 Pachycordyle michaeli Coelenterates-Hydrozoans
## 8  -570    Gambusia holbrooki                  Fishes
## 9  -569       Corella inflata                Tunicates
## 10 -568                  <NA>                    <NA>
```

Homework

1. Import the dataset you found for you last homework into R.
2. Submit both the your found datast and an R script of the code you used to import it onto d2l