Open Source Open Science Workshop – Module 2: Matrices and Data Tables

Alex Howard, edited from Danielle Walkup

Last compiled on 21 August, 2024, 16:45

1 R Objects

1.1 Data Types

This handout is associated with the R file: "Module2.R"

R has a number of basic object types that we use.

R data type	Other terms	Examples
Numeric	Real, Continuous, Quantitative	4.69; 3.5, 3, 405, 285
Integer	Counts	1, 2, 3
Factor	Ordinal, Categorical, Nominal, Discrete	Low, Medium, High; Site1, Site2, Site3
Logical		TRUE, FALSE
Character		gravid, lizard, bird

We can identify data types by using class()

Some examples of the data types:

```
# The classes can vary depending on the type of data:
mydata_num <- c(1.4, 5.6, 8.3, 5.6, 5.0)
class(mydata_num)  #returns numbers
```

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

```
mydata_cha <- c("1.4", "5.6", "8.3", "5.6", "5.0")
class(mydata_cha)  #returns characters
```

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

```
# if you need to change the class, you can:
mydata_num2 <- as.numeric(mydata_cha)
class(mydata_num2)
```

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

```r
mydata_cha2 <- as.character(mydata_num)
class(mydata_cha2)
```

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

```r
# Notice what happens when we change them to integers:
mydata_int <- as.integer(mydata_cha)
class(mydata_int)
```

```
## [1] "integer"
```

```r
mydata_int
```

```
## [1] 1 5 8 5 5
```

Factors are a special type of character that is ordered

```r
## Factors
# Ordering of factors defaults to alphabetical:
fruit <- factor(c("apple","pear","banana","grape"))
fruit
```

```
## [1] apple  pear   banana grape
## Levels: apple banana grape pear
```

```r
# But we can change that if we want to:
fruit2 <- factor(c("apple","pear","banana","grape")
                           , levels = c("apple","pear","banana","grape"))
fruit2
```

```
## [1] apple  pear   banana grape
## Levels: apple pear banana grape
```

```r
#
# We can also create an ordinal variable:
fruit_ord <- factor(c("apple",'pear',"banana","grape"), ordered = TRUE)
#note the < between the levels now AND our levels changed back to alphabetical
fruit_ord
```

```
## [1] apple  pear   banana grape
## Levels: apple < banana < grape < pear
```

```r
# We can rearrange the order:
fruit_ord2 <- factor(c("apple",'pear',"banana","grape")
                                  , levels = c("apple","pear","banana","grape")
                                  , ordered = TRUE)
fruit_ord2  #Much better - if you like grapes...
```

```
## [1] apple  pear   banana grape
## Levels: apple < pear < banana < grape
```

```r
# Notice there are some constraints on classes:
fruit_ch <- as.character(fruit)
class(fruit_ch)
```

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

```r
fruit_ch
```

```
## [1] "apple"  "pear"   "banana" "grape"
```

```r
fruit_num <- as.numeric(fruit_ch)  #It'll do it, but it'll do it badly.
```

```
## Warning: NAs introduced by coercion
```

```r
class(fruit_num)
```

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

```r
fruit_num
```

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

```r
# BUT, be careful, because we can coerce factors to numbers or integers:
fruit_num2 <- as.numeric(fruit)  #using the alphabetically ordered fruit
class(fruit_num2)
```

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

```r
fruit_num2  #Note that it keeps the ordering
```

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

```r
fruit_int <- as.integer(fruit2)  #using our fruit orders
class(fruit_int)
```

```
## [1] "integer"
```

```r
fruit_int  #Again, note that it keeps the changed ordering
```

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

These data types can be combined in different ways to create different data objects.

1.2 Data Objects

Data.Structures	Description	Examples
Vectors	A collection of elements of one type (typically of character, logical, integer, or numeric).	a <- c(1, 2, 3, 4, 5) b <- c('apple', 'pear', 'banana', 'grape')
Lists	A special type of vector, each element can be a different type, it can even be a list of lists.	ex_list <- list(1, 'a', TRUE, 1+4i) ex_biglist <- list(a = 'Testing', b = c(1,3,5,6,7), flowers = head(iris))
Matrices	An extension of numeric or character vectors that has dimensions: rows and columns.	ex_mat <- matrix(1:10, nrow = 2) ex_mat2 <- matrix(1:10, nrow = 2, byrow = TRUE)
Data Frames	A special type of list where every element has the same length. This is generally the most commonly used data structure for tabular data and what we typically use for statistics.	ex_df <- data.frame(id = rep(c('a','b','c'),4), height = 1:12, width = 12:1)

1.2.1 Vectors

Vectors are a collection of elements of the same type. The data can be of any type, but all elements within them are forced to the same type.

```
# Set up a couple of example vectors:
vec1 <- c(1, 2, 3, 4, 5)
vec2 <- c("apple", "pear", "banana", "grape")
vec3 <- c("blue", 3, 5i, "lizard")

# We can see elements in the vectors:
vec2[3] #the single bracket lets us see the element in the 3rd place in the vector
```

```
## [1] "banana"
```

```
vec2[8] #there's nothing in here - because we only have the 4 fruits!
```

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

```
# We can get all sorts of information about the vectors:
class(vec1)
```

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

```r
class(vec3)  #all elements must be the same type, all elements in this vector are characters
```

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

```r
length(vec2)
```

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

```r
is.na(vec1)  #a handy function to check for missing values
```

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

```r
sum(vec1)    #we can run functions specific to the given class
```

```
## [1] 15
```

```r
#sum(vec2)   #no numbers to add!
str(vec1)
```

```
##  num [1:5] 1 2 3 4 5
```

1.2.2 Lists

Lists differ from vectors in that each element can be of a different type. We can even have lists of lists - a nested list.

List Examples:

```r
# set up a list
ex_list <- list(1, "a", TRUE, 1+4i)
class(ex_list)  #now we see this is a list, with its special list properties
```

```
## [1] "list"
```

```r
# We can still see the different elements and characteristics
ex_list[2]
```

```
## [[1]]
## [1] "a"
```

```r
length(ex_list)
```

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

```r
str(ex_list)
```

```
## List of 4
##  $ : num 1
##  $ : chr "a"
##  $ : logi TRUE
##  $ : cplx 1+4i
```

```r
# You can also name the elements in each list
ex_biglist <- list(a = "Testing", b = 1:10, flowers = head(iris))

# now if we want to see what's in the list:
names(ex_biglist)  #we can also see the names
```

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

```r
ex_biglist[2]  #and what's in the list
```

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

```r
ex_list[[2]] #we use double brackets to select specific elements
```

```
## [1] "a"
```

```r
ex_biglist$b  #alternatively, if we have elements in the list named, we can use
```

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

```r
#the '$' to choose the different names and show what's in them
length(ex_biglist)  #but we still only have 3 things in the list
```

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

1.2.3 Matrices

A matrix is a 2 dimensional dataset, made of rows and columns

Matrix Examples:

```r
# let's set up a matrix
ex_mat <- matrix(1:10, nrow = 2)
ex_mat
```

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

```r
ex_mat2 <- matrix(1:10, nrow = 2, byrow = TRUE)
#TIP: instead of writing out 1,2,3,4,5,6,7,8,9,10, we use 1:10 to give us the
#sequence of numbers
ex_mat2
```

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

```r
# and we can see characteristics:
class(ex_mat)
```

```
## [1] "matrix" "array"
```

```r
str(ex_mat)
```

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

```r
length(ex_mat)
```

```
## [1] 10
```

```r
dim(ex_mat)  #Now we can see the dimensions
```

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

```r
ex_mat[2,1]  #now we have 2 dimensions, we can use [row,column] to id an element
```

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

```r
ex_mat[,2]  #or see just a column
```

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

```r
ex_mat[2,]  #or see just a row
```

```
## [1]  2  4  6  8 10
```

```r
sum(ex_mat)  #can still do math on a numeric matrix -
```

```
## [1] 55
```

```r
t(ex_mat)  #can also transpose a matrix
```

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

1.2.4 Data Frames

A data frame is data displayed in a format of a table. Each column should have the same type of data.

Data Frame Examples:

```
# here we will create a data frame with 3 columns: id, height, and width
ex_df <- data.frame(id = rep(c("a","b","c"),4), height = 1:12, width = 12:1)
ex_df  #lets make sure it did what we expected
```

```
##    id height width
## 1   a      1    12
## 2   b      2    11
## 3   c      3    10
## 4   a      4     9
## 5   b      5     8
## 6   c      6     7
## 7   a      7     6
## 8   b      8     5
## 9   c      9     4
## 10  a     10     3
## 11  b     11     2
## 12  c     12     1
```

```
# and check out some of its characteristics:
class(ex_df)
```

```
## [1] "data.frame"
```

```
str(ex_df)
```

```
## 'data.frame':    12 obs. of  3 variables:
##  $ id    : chr  "a" "b" "c" "a" ...
##  $ height: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ width : int  12 11 10 9 8 7 6 5 4 3 ...
```

```
summary(ex_df)
```

```
##       id                  height          width
##  Length:12          Min.   : 1.00   Min.   : 1.00
##  Class :character   1st Qu.: 3.75   1st Qu.: 3.75
##  Mode  :character   Median : 6.50   Median : 6.50
##                     Mean   : 6.50   Mean   : 6.50
##                     3rd Qu.: 9.25   3rd Qu.: 9.25
##                     Max.   :12.00   Max.   :12.00
```

```
length(ex_df)  #we can still get a length, but it may not quite be the info we want
```

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

```r
dim(ex_df)
```

```
## [1] 12  3
```

```r
t(ex_df)  #we can still transpose the data frame
```

```
##        [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## id     "a"  "b"  "c"  "a"  "b"  "c"  "a"  "b"  "c"  "a"   "b"   "c"
## height " 1" " 2" " 3" " 4" " 5" " 6" " 7" " 8" " 9" "10"  "11"  "12"
## width  "12" "11" "10" " 9" " 8" " 7" " 6" " 5" " 4" " 3"  " 2"  " 1"
```

```r
# and we can see the different parts:
ex_df$id
```

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

```r
class(ex_df$id)
```

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

```r
class(ex_df$width)
```

```
## [1] "integer"
```

```r
#and it has defaulted width to integer, which we could change:
as.numeric(ex_df$width)
```

```
##  [1] 12 11 10  9  8  7  6  5  4  3  2  1
```

```r
class(ex_df$width)
```

```
## [1] "integer"
```

```r
# We could even do math within the data frame:
sum(ex_df$width)
```

```
## [1] 78
```

```r
mean(ex_df$width)
```

```
## [1] 6.5
```

```r
## Subsetting our data frame
# we've subsetted throughout this script using brackets [], but there are lots
# of ways to subset our database:
ex_df[1, ] #just a row
```

```
##   id height width
## 1  a      1    12
```

```r
ex_df[, 1] #just a column
```

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

```r
ex_df[2, 2] #row 2 and column 2
```

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

```r
# we can subset based on different criteria
ex_df[which(ex_df$id == "a"), ]
```

```
##    id height width
## 1   a      1    12
## 4   a      4     9
## 7   a      7     6
## 10  a     10     3
```

```r
ex_df[ex_df$id %in% "a", ]
```

```
##    id height width
## 1   a      1    12
## 4   a      4     9
## 7   a      7     6
## 10  a     10     3
```

```r
subset(ex_df, id == "a")
```

```
##    id height width
## 1   a      1    12
## 4   a      4     9
## 7   a      7     6
## 10  a     10     3
```

2 Reading and Writing Data

Reading and writing data are probably what you are going to be doing the most of. Here are some of the basic steps to get you started.

2.1 Importing Data

Although we can create data frames, like we did in the previous examples, for the most part, we are probably going to be importing our data from something like an excel sheet. Here's an example to get us started:

```
## read.csv or read.table are the typical functions we will use to upload data

## read.csv has a ton of options to read in your data set, check out
## the different arguments:
args(read.csv)
```

```
## function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
##     fill = TRUE, comment.char = "", ...)
## NULL
```

```
#help(read.csv)
# header  - tell it whether or not there is a head row
# col.names - can change the names of the columns
# row.names - or add row names
# colClasses - can specify the column classes
# strip.white - removes leading or trailing white spaces
# blank.lines.skip - skips empty rows
# na.strings - what values should be considered NA

# First, lets make sure the file is where we think it is
dir("data")  #looks good
```

```
## [1] "LizardData.csv"  "LizardData.txt"  "LizardData.xlsx"
```

```
# I'll set a file path here (makes it more easily accessible)
#NOTE: by using data/ I'm telling R to look in that folder for the file
liz_csv <- "data/LizardData.csv"

## If the file is saved as a .csv we can simply read it in:
liz_df <- read.csv(liz_csv)
head(liz_df, 2)  #double check that it worked
```

```
##     ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1 ASMA  101  EFB    4   M  94  221         27           N 5/26/2012
## 2 ASMA  201  EFB    7   M  92  213    86   25           N 26-May-12
```

```
# and we can check out our new data frame:
class(liz_df)
```

```
## [1] "data.frame"
```

```
head(liz_df)  #look it automatically reads our headings!
```

```
##      ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1  ASMA  101  EFB    4   M  94  221         27           N 5/26/2012
## 2  ASMA  201  EFB    7   M  92  213    86   25           N 26-May-12
## 3  ASMA  202  EFB    7   M  84  199    83   18           N 26-May-12
## 4  ASMA  203  EFC    8   M  92  228         23           N 26-May-12
## 5  ASMA  301  EBA    7   M  91  120         20          no 5/26/2012
## 6 ASMA   302  ECE    1   M 105  235         24          no 5/26/2012
```

```
str(liz_df)  #tells us the column classes and info about the data.frame
```

```
## 'data.frame':    107 obs. of  12 variables:
##  $ ID   : chr  "ASMA" "ASMA" "ASMA" "ASMA" ...
##  $ Mark : chr  "101" "201" "202" "203" ...
##  $ Grid : chr  "EFB" "EFB" "EFB" "EFC" ...
##  $ Trap : int  4 7 7 8 7 1 6 1 8 8 ...
##  $ Sex  : chr  "M" "M" "M" "M" ...
##  $ SVL  : int  94 92 84 92 91 105 89 82 93 74 ...
##  $ Tail : int  221 213 199 228 120 235 186 124 200 177 ...
##  $ Regen: chr  "" "86" "83" "" ...
##  $ Mass : num  27 25 18 23 20 24 22 19 27 15.5 ...
##  $ Notes: chr  "" "" "" "" ...
##  $ Recap: chr  "N" "N" "N" "N" ...
##  $ Date : chr  "5/26/2012" "26-May-12" "26-May-12" "26-May-12" ...
```

```
dim(liz_df)
```

```
## [1] 107  12
```

```
summary(liz_df)  #gives us a brief overview of each column, already, I can see
```

```
##       ID                Mark               Grid                Trap      
##  Length:107         Length:107         Length:107         Min.   :1.000  
##  Class :character   Class :character   Class :character   1st Qu.:3.000  
##  Mode  :character   Mode  :character   Mode  :character   Median :5.000  
##                                                           Mean   :5.038  
##                                                           3rd Qu.:7.000  
##                                                           Max.   :9.000  
##                                                           NA's   :1      
##      Sex                 SVL              Tail            Regen          
##  Length:107         Min.   : 13.00   Min.   : 18.00   Length:107        
##  Class :character   1st Qu.: 44.00   1st Qu.: 60.25   Class :character  
##  Mode  :character   Median : 48.00   Median : 74.00   Mode  :character  
##                     Mean   : 52.96   Mean   : 85.95                     
##                     3rd Qu.: 57.25   3rd Qu.: 88.75                     
##                     Max.   :105.00   Max.   :235.00                     
##                     NA's   :9        NA's   :9                          
##       Mass          Notes               Recap               Date          
##  Min.   : 1.40   Length:107         Length:107         Length:107        
##  1st Qu.: 2.90   Class :character   Class :character   Class :character  
##  Median : 3.90   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 6.46                                                           
##  3rd Qu.: 5.70                                                           
##  Max.   :27.00                                                           
##  NA's   :10
```

```
#some indicators of a messy data set: check out the Sex, Regen, and the Recap
#column summaries!

## If we have a text file, we can still read it in using read.csv
```

```
liz_txt <- "data/LizardData.txt"
#sep = "\t" tells R its a tab-delimited text file
liz_df_txt <- read.csv(liz_txt, sep = "\t")

head(liz_df_txt, 2)  #and it looks the same as above
```

```
##     ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1 ASMA  101  EFB    4   M  94  221         27           N 5/26/2012
## 2 ASMA  201  EFB    7   M  92  213    86   25           N 26-May-12
```

```
## Finally, we can also load the excel file using our package openxlsx:
liz_xl <- "data/LizardData.xlsx"
liz_df_xl <- read.xlsx(liz_xl)
head(liz_df_xl, 2)
```

```
##     ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap  Date
## 1 ASMA  101  EFB    4   M  94  221  <NA>   27  <NA>     N 41055
## 2 ASMA  201  EFB    7   M  92  213    86   25  <NA>     N 41055
```

```
liz_df_xl2 <- read_excel(liz_xl)
head(liz_df_xl2)
```

```
## # A tibble: 6 x 12
##   ID    Mark  Grid   Trap Sex     SVL  Tail Regen  Mass Notes Recap
##   <chr> <chr> <chr> <dbl> <chr> <dbl> <dbl> <chr> <dbl> <chr> <chr>
## 1 ASMA  101   EFB       4 M        94   221 <NA>     27 <NA>  N
## 2 ASMA  201   EFB       7 M        92   213 86       25 <NA>  N
## 3 ASMA  202   EFB       7 M        84   199 83       18 <NA>  N
## 4 ASMA  203   EFC       8 M        92   228 <NA>     23 <NA>  N
## 5 ASMA  301   EBA       7 M        91   120 <NA>     20 <NA>  no
## 6 ASMA  302   ECE       1 M       105   235 <NA>     24 <NA>  no
## # i 1 more variable: Date <dttm>
```

```
# but note the differences in how read.xlsx default loads the data:
str(liz_df)
```

```
## 'data.frame':    107 obs. of  12 variables:
##  $ ID   : chr  "ASMA" "ASMA" "ASMA" "ASMA" ...
##  $ Mark : chr  "101" "201" "202" "203" ...
##  $ Grid : chr  "EFB" "EFB" "EFB" "EFC" ...
##  $ Trap : int  4 7 7 8 7 1 6 1 8 8 ...
##  $ Sex  : chr  "M" "M" "M" "M" ...
##  $ SVL  : int  94 92 84 92 91 105 89 82 93 74 ...
##  $ Tail : int  221 213 199 228 120 235 186 124 200 177 ...
##  $ Regen: chr  "" "86" "83" "" ...
##  $ Mass : num  27 25 18 23 20 24 22 19 27 15.5 ...
##  $ Notes: chr  "" "" "" "" ...
##  $ Recap: chr  "N" "N" "N" "N" ...
##  $ Date : chr  "5/26/2012" "26-May-12" "26-May-12" "26-May-12" ...
```

```
str(liz_df_txt)
```

```
## 'data.frame':    107 obs. of  12 variables:
##  $ ID   : chr  "ASMA" "ASMA" "ASMA" "ASMA" ...
##  $ Mark : chr  "101" "201" "202" "203" ...
##  $ Grid : chr  "EFB" "EFB" "EFB" "EFC" ...
##  $ Trap : int  4 7 7 8 7 1 6 1 8 8 ...
##  $ Sex  : chr  "M" "M" "M" "M" ...
##  $ SVL  : int  94 92 84 92 91 105 89 82 93 74 ...
##  $ Tail : int  221 213 199 228 120 235 186 124 200 177 ...
##  $ Regen: chr  "" "86" "83" "" ...
##  $ Mass : num  27 25 18 23 20 24 22 19 27 15.5 ...
##  $ Notes: chr  "" "" "" "" ...
##  $ Recap: chr  "N" "N" "N" "N" ...
##  $ Date : chr  "5/26/2012" "26-May-12" "26-May-12" "26-May-12" ...
```

```
str(liz_df_xl)
```

```
## 'data.frame':    107 obs. of  12 variables:
##  $ ID   : chr  "ASMA" "ASMA" "ASMA" "ASMA" ...
##  $ Mark : chr  "101" "201" "202" "203" ...
##  $ Grid : chr  "EFB" "EFB" "EFB" "EFC" ...
##  $ Trap : num  4 7 7 8 7 1 6 1 8 8 ...
##  $ Sex  : chr  "M" "M" "M" "M" ...
##  $ SVL  : num  94 92 84 92 91 105 89 82 93 74 ...
##  $ Tail : num  221 213 199 228 120 235 186 124 200 177 ...
##  $ Regen: chr  NA "86" "83" NA ...
##  $ Mass : num  27 25 18 23 20 24 22 19 27 15.5 ...
##  $ Notes: chr  NA NA NA NA ...
##  $ Recap: chr  "N" "N" "N" "N" ...
##  $ Date : num  41055 41055 41055 41055 41055 ...
```

```
str(liz_df_xl2)
```

```
## tibble [107 x 12] (S3: tbl_df/tbl/data.frame)
##  $ ID   : chr [1:107] "ASMA" "ASMA" "ASMA" "ASMA" ...
##  $ Mark : chr [1:107] "101" "201" "202" "203" ...
##  $ Grid : chr [1:107] "EFB" "EFB" "EFB" "EFC" ...
##  $ Trap : num [1:107] 4 7 7 8 7 1 6 1 8 8 ...
##  $ Sex  : chr [1:107] "M" "M" "M" "M" ...
##  $ SVL  : num [1:107] 94 92 84 92 91 105 89 82 93 74 ...
##  $ Tail : num [1:107] 221 213 199 228 120 235 186 124 200 177 ...
##  $ Regen: chr [1:107] NA "86" "83" NA ...
##  $ Mass : num [1:107] 27 25 18 23 20 24 22 19 27 15.5 ...
##  $ Notes: chr [1:107] NA NA NA NA ...
##  $ Recap: chr [1:107] "N" "N" "N" "N" ...
##  $ Date : POSIXct[1:107], format: "2012-05-26" "2012-05-26" ...
```

TIP: Things to check before converting your Excel file to text: 1. You can only have one row of headers 2. Use short descriptive headers without spaces - Use "_" or caps between words (i.e. first_name or FirstName) - Headers are typed over, and over, and over, so keep them short - Don't start a header with a number;

R will accept it, but will put an X in front of it 3. Remove all summary data (e.g. Average, sum, max, min, etc.) 4. Check that all values in a column are of the same type (e.g. number, date, character, etc.) 5. Remove commas and # symbols throughout - Commas will mess up comma-delimited files - # indicates a comment in R 6. Missing data are okay, but keep an eye on it.

##2.2 Cleaning and Exporting Data

Working with the liz_data data set, we will do a little cleaning (you'll learn more about cleaning and manipulating data sets tomorrow), and output some summary data as a csv, table, and figures. When setting up data to work with analyses you typically end up rearranging, subsetting, and adding data columns to get a clean and tidy data set. (What does tidy data look like?). Here we are just going to run through a few examples to clean and summarize data in order to export a cleaned version of our data. You will learn more about data manipulation tomorrow.

```
# Let's check the summary again to help identify any issues
summary(liz_df)
```

```
##       ID                 Mark               Grid               Trap
##  Length:107         Length:107         Length:107         Min.   :1.000
##  Class :character   Class :character   Class :character   1st Qu.:3.000
##  Mode  :character   Mode  :character   Mode  :character   Median :5.000
##                                                           Mean   :5.038
##                                                           3rd Qu.:7.000
##                                                           Max.   :9.000
##                                                           NA's   :1
##      Sex                 SVL              Tail           Regen
##  Length:107         Min.   : 13.00   Min.   : 18.00   Length:107
##  Class :character   1st Qu.: 44.00   1st Qu.: 60.25   Class :character
##  Mode  :character   Median : 48.00   Median : 74.00   Mode  :character
##                     Mean   : 52.96   Mean   : 85.95
##                     3rd Qu.: 57.25   3rd Qu.: 88.75
##                     Max.   :105.00   Max.   :235.00
##                     NA's   :9        NA's   :9
##       Mass          Notes              Recap              Date
##  Min.   : 1.40   Length:107         Length:107         Length:107
##  1st Qu.: 2.90   Class :character   Class :character   Class :character
##  Median : 3.90   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 6.46
##  3rd Qu.: 5.70
##  Max.   :27.00
##  NA's   :10
```

```
# Some known issues:
# Sex - we have some duplicate categories (M vs. Male) and an age class (J)
# Regen - characters but should be numbers
# Recap - again, with the duplicate categories (N vs no vs No)
# Date is a character here, but we actually want a date. But because people
# entered the data in multiple formats, it's going to take some work

# We can use some functionality from tidyverse, particularly the dplyr package
# A collection of R packages for data science: https://www.tidyverse.org/
# R for Data Science by Hadley Wickham and Garrett Grolemund: http://r4ds.had.co.nz/
library(dplyr)
```

```
#
# First lets get rid of those no's (and probable yes's) in the Regen
# I'm just going to run a table to see what we have going on
table(liz_df$Regen)  #so we have 3 No's, 1 Yes, and the weird 38/6
```

```
##
##         1  121   13  130   14   19   20   21   23   28   32   34   35   36   38
##   82    1    1    2    1    1    1    1    1    1    1    1    2    1    2
## 38/6   83   86   No  Yes
##    1    1    1    3    1
```

```
#
# let's check out that weird 38/6
# We can use ([row,col]) here to find the row that matches the one weird record:
liz_df[liz_df$Regen == "38/6", ]
```

```
##      ID Mark Grid Trap Sex SVL Tail Regen Mass                            Notes
## 47 UTST  626  EAC    3   M  48   47  38/6  3.8 2 REGENS ONE AT 38 AND ONE AT 6
##    Recap     Date
## 47     Y 2/7/2013
```

```
# so the notes tells me that there were two areas of tail regeneration, so I'm just
# going to change this to 38 using dplyr::recode
liz_df$Regen <- recode(liz_df$Regen, "38/6" = "38")

# we also need to remove the no's and yes's as just blanks
liz_df$Regen <- recode(liz_df$Regen, "No" = "", "Yes" = "")
# and one last table to double check
table(liz_df$Regen)
```

```
##
##       1 121  13 130  14  19  20  21  23  28  32  34  35  36  38  83  86
##  86   1   1   2   1   1   1   1   1   1   1   1   1   2   1   3   1   1
```

```
# so lets make that numeric now.
liz_df$Regen <- as.numeric(liz_df$Regen)

## Let's clean up the sex column, because we do want to use that:
table(liz_df$Sex)
```

```
##
##         F    J    M Male
##    1   35    1   63    4
```

```
# lets change the Male to M - capitilization matters!
liz_df$Sex <- recode(liz_df$Sex, "Male" = "M", "J" = "")

## So now that we have those basic columns cleaned up, lets subset the data frame
## down to just the columns we want to use
head(liz_df, 1)
```

```
##     ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1 ASMA  101  EFB    4   M  94  221    NA   27           N 5/26/2012
```

```
liz_sub <- liz_df[, c(1:2, 5:9)]
head(liz_sub, 2)
```

```
##     ID Mark Sex SVL Tail Regen Mass
## 1 ASMA  101   M  94  221    NA   27
## 2 ASMA  201   M  92  213    86   25
```

```
# Alternative ways to subset:
liz_sub1 <- liz_df[, c("ID", "Mark", "Sex", "SVL", "Tail", "Regen", "Mass")]
liz_sub2 <- subset(liz_df, select = c("ID", "Mark", "Sex", "SVL", "Tail", "Regen", "Mass"))
liz_sub3 <- liz_df %>% select(ID, Mark, Sex, SVL, Tail, Regen, Mass)

## So many NA's - what if we wanted to create a subset of our data frame with
## no NA's?
sum(is.na(liz_sub$Sex)) #sum counts FALSE as 0 and TRUE as 1, letting us get
```

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

```
# a count of na's
liz_subna <- liz_sub[!is.na(liz_sub$Sex), ]
is.na(liz_subna$Sex)  #that's a lot of falses, lets summarize that
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
sum(is.na(liz_subna$Sex))
```

```
## [1] 0
```

```
dim(liz_sub)
```

```
## [1] 107   7
```

```
dim(liz_subna)
```

```
## [1] 104   7
```

```r
#We can also subset according to values in our dataset
#for example, say we only wanted to look at males:
liz_male <- liz_df[liz_df$Sex == "M",]
head(liz_male)
```

```
##      ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1  ASMA  101  EFB    4   M  94  221    NA   27           N 5/26/2012
## 2  ASMA  201  EFB    7   M  92  213    86   25           N 26-May-12
## 3  ASMA  202  EFB    7   M  84  199    83   18           N 26-May-12
## 4  ASMA  203  EFC    8   M  92  228    NA   23           N 26-May-12
## 5  ASMA  301  EBA    7   M  91  120    NA   20          no 5/26/2012
## 6 ASMA   302  ECE    1   M 105  235    NA   24          no 5/26/2012
```

```r
#there are still NA's
#we can add conditions using the & symbol
liz_male2 <- liz_df[liz_df$Sex == "M" & !is.na(liz_df$Sex),]
head(liz_male2)
```

```
##      ID Mark Grid Trap Sex SVL Tail Regen Mass Notes Recap      Date
## 1  ASMA  101  EFB    4   M  94  221    NA   27           N 5/26/2012
## 2  ASMA  201  EFB    7   M  92  213    86   25           N 26-May-12
## 3  ASMA  202  EFB    7   M  84  199    83   18           N 26-May-12
## 4  ASMA  203  EFC    8   M  92  228    NA   23           N 26-May-12
## 5  ASMA  301  EBA    7   M  91  120    NA   20          no 5/26/2012
## 6 ASMA   302  ECE    1   M 105  235    NA   24          no 5/26/2012
```

```r
#if we only want animals less than 60 SVL
liz_small <- liz_subna[liz_subna$SVL < 60,]
head(liz_small)
```

```
##        ID Mark  Sex SVL Tail Regen Mass
## NA   <NA> <NA> <NA>  NA   NA    NA   NA
## 15   SCAR  102    M  59   48    35  7.0
## 16   SCAR  103    M  54   63    NA  5.7
## 19   SCAR  302    F  55   55    NA  6.0
## NA.1 <NA> <NA> <NA>  NA   NA    NA   NA
## 21   SCAR  401    M  59   96    NA  8.0
```

Generally, you don't want to change your input data, but create a new, cleaned data set that you can use for analyses. So, now that we have a clean, subsetted version of our data, we can export a copy to use later.

```r
## first I'm going to add a folder to the directory
dir.create("outputs")

write.csv(liz_sub, "outputs/LizardData_Clean.csv")
```

We can also export summary data as files or tables.

```r
## What if you just want a table of summary information???
liz_summ <- liz_sub %>%
  group_by(ID) %>%
  summarise(avgSVL = mean(SVL, na.rm = T),
            sdSVL = sd(SVL, na.rm = T),
            maxSVL = max(SVL, na.rm = T),
            minSVL = min(SVL, na.rm = T),
            avgTail = mean(Tail, na.rm=T),
            sdTail=sd(Tail, na.rm = T),
            avgMass = mean(Mass, na.rm=T),
            sdMass=sd(Mass, na.rm=T),
            Nliz = n())

# and again we can save that summary output:
write.csv(liz_summ, "outputs/LizardSummaryData.csv")
```

3 Additional Resources

Great resources for learning and programming in R

- R for Data Science: https://r4ds.had.co.nz/index.html
- Software Carpentry - Programming with R: http://swcarpentry.github.io/r-novice-inflammation/
- Software Carpentry - Reproducible Research: http://swcarpentry.github.io/r-novice-gapminder/
- Quick-R Data Camp: https://www.statmethods.net/index.html
- An Introduction to R - CRAN (pdf): https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf