Introduction to R

TAMU HPRC Short Courses – Spring 2018

Dr. Noushin Ghaffari

This course will provide an introduction to R. We use the Jupyter Notebook to run the commands. We will also demonstrate command execution using Ada. Parts of this course are based on Software Carpentry “Programming with R” and “R for Reproducible Scientific Analysis” lessons.

What is R?

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing. It supports multiple platforms and can be easily extended.

The “Comprehensive R Archive Network” (CRAN) is a collection of sites which carry identical material, consisting of the R distribution(s), the contributed extensions, documentation for R, and binaries.

The CRAN master site at WU (Wirtschaftsuniversität Wien) in Austria can be found at the URL https://CRAN.R-project.org/

and is mirrored daily to many sites around the world. See https://CRAN.R-project.org/mirrors.html for a complete list of mirrors. Please use the CRAN site closest to you to reduce network load.

Basic Usage of R

Using R as a Calculator

R can be used as a simple calculator.

In [1]:
    100*3.5

450

One should note the order of operations, which affects the results. From highest to lowest precedence:

- Parentheses: (,)
- Exponents: ^ or **
- Divide: /
- Multiply: *
- Add: +
- Subtract: -

In [2]:
    3^2+10.6

16.6
Mathematical Operations in R

R offers many mathematical functions, mainly as part of the “base” package. Here is how you can list all the functions:

```r
In [6]:
2 + pi
```

```r
3.14159265358979
```

```r
In [8]:
sin(90*pi/180) #converting radian to degree by pi/180
```

```r
1
```

```r
In [9]:
log10(10)
```

```r
1
```

```r
In [10]:
log2(10)
```

```r
3.32192809488736
```

```r
In [11]:
exp(2.6)
```

```r
13.4637380350017
```

Programming with R

Every programming language provides variables, data structure and set of commands. The purpose of these elements is to enable users to write commands that are understandable by the computer and will generate reproducible results. Let’s look at R data types:

- character: “R”, “test”
- numeric: 1, 10, 11.1
- integer: 2L (the L tells R to store this as an integer)
- logical: TRUE, FALSE
- complex: 1+4i (complex numbers with real and imaginary parts)

```r
In [12]:
#Assignment to a variable
x <- 1.8
```
R provides many built-in functions to examine features of vectors and other objects, for example:

- `class()` - what kind of object is it (high-level)?
- `typeof()` - what is the object’s data type (low-level)?
- `length()` - how long is it? What about two dimensional objects?
- `attributes()` - does it have any metadata?

```
In [13]: (x)
    1 2 3 4 5

In [14]: a=1
    print(a)
    [1] 1

In [15]: b=T
    print(b)
    TRUE

In [16]: c="This is a test"
    print(c)
    [1] "This is a test"
```

Most important R data structures:

- vector (atomic or list)
  - vector is a collection of elements that are most commonly of mode character, logical, integer or numeric
  - Lists can encompass any mixture of data types vs atomic vector that hold only one data type
- matrix
  - In R matrices are an extension of the numeric or character vectors. They are not a separate type of object but simply an atomic vector with dimensions; the number of rows and columns.
- data frame
  - The de facto data structure for most tabular data and what we use for statistics. A data frame is a special type of list where every element of the list has same length

```
In [17]: y <- 5:15
    (y)
    class(y)
    5 6 7 8 9 10 11 12 13 14 15
    'integer'

In [18]: length(y)
    11

#making a vector
v<- vector("character", length=5)
typeof(v)
'character'
```
In [20]:
   class(v)

'character'

In [21]:
   print(v)

[1] "a" "b" "c" "d" "e"

In [22]:
   print(v)

[1] "a" "b" "c" "d" "e"

In [23]:
   #easy addition to the list
   v <- c(v,"f","g")
   print(v)

[1] "a" "b" "c" "d" "e" "f" "g"

In [24]:
   #Matrix in R
   m = matrix(data = 1:10, nrow = 2, ncol = 5)
   print(m)

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

In [25]:
   nrow(m)

2

In [26]:
   ncol(m)

5

In [27]:
   m <- cbind(m,33:34)
   m

   1  3  5  7  9  33
   2  4  6  8 10  34

In [28]:
   #Data frames
   df <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
   df

       id  x  y
   a    a  1 11
   b    b  2 12
   c    c  3 13
   d    d  4 14
   e    e  5 15
   f    f  6 16
   g    g  7 17
   h    h  8 18
   i    i  9 19
   j    j 10 20
In [29]: names(df)
   'id' 'x' 'y'

In [30]: str(df)
'data.frame': 10 obs. of 3 variables:
$ id: Factor w/ 10 levels "a","b","c","d",...: 1 2 3 4 5 6 7 8 9 10
$ x : int 1 2 3 4 5 6 7 8 9 10
$ y : int 11 12 13 14 15 16 17 18 19 20

In [31]: class(df)
'factor'

**Working with Data**

In [32]: download.file(url="https://raw.githubusercontent.com/swcarpentry/files/master/inflammation-01.csv")

In [33]: input=read.csv("inflammation-01.csv", header=F)

In [34]: head(input)

<p>| | | | | | | | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In [35]: getwd()
'/home/noushin'

In [36]: dim(input)
60 40

In [37]: input[6,40]
1

In [38]: input[4,27]
2

In [39]: input[16,3]
1 2 1 2 1 1
In [40]:
colnames(input)

'V1' 'V2' 'V3' 'V4' 'V5' 'V6' 'V7' 'V8' 'V9' 'V10' 'V11' 'V12' 'V13' 'V14' 'V15' 'V16'
'V17' 'V18' 'V19' 'V20' 'V21' 'V22' 'V23' 'V24' 'V25' 'V26' 'V27' 'V28' 'V29' 'V30' 'V31'
'V32' 'V33' 'V34' 'V35' 'V36' 'V37' 'V38' 'V39' 'V40'

In [41]:
colnames(input) <- c(paste("C", 1:40, sep=""))

In [42]:
colnames(input)

'C1' 'C2' 'C3' 'C4' 'C5' 'C6' 'C7' 'C8' 'C9' 'C10' 'C11' 'C12' 'C13' 'C14' 'C15' 'C16'
'C17' 'C18' 'C19' 'C20' 'C21' 'C22' 'C23' 'C24' 'C25' 'C26' 'C27' 'C28' 'C29' 'C30'
'C31' 'C32' 'C33' 'C34' 'C35' 'C36' 'C37' 'C38' 'C39' 'C40'

In [43]:
min(input[,1])

0

In [44]:
max(input[,10])

9
In [45]:  

    #useful command  
    str(input)  

'data.frame': 60 obs. of 40 variables:  
$ C1 : int 0 0 0 0 0 0 0 0 0 ...  
$ C2 : int 0 1 1 0 1 0 0 0 1 ...  
$ C3 : int 1 2 1 2 1 1 2 1 0 1 ...  
$ C4 : int 3 1 3 0 3 2 2 2 3 ...  
$ C5 : int 1 2 3 4 3 2 4 3 1 1 ...  
$ C6 : int 2 1 2 2 1 4 2 1 5 3 ...  
$ C7 : int 4 3 6 2 3 2 2 2 6 5 ...  
$ C8 : int 7 2 2 1 5 1 5 3 5 3 ...  
$ C9 : int 8 2 5 6 2 6 5 5 5 5 ...  
$ C10: int 3 6 9 7 4 4 8 3 8 8 ...  
$ C11: int 3 10 5 10 4 7 6 7 2 6 ...  
$ C12: int 3 11 7 7 6 5 8 4 8 ...  
$ C13: int 10 5 4 9 6 6 11 8 11 12 ...  
$ C14: int 5 9 5 13 5 9 5 12 5 ...  
$ C15: int 7 4 4 8 3 9 4 10 10 13 ...  
$ C16: int 4 4 15 8 10 15 13 9 11 6 ...  
$ C17: int 7 7 5 15 8 4 5 15 9 13 ...  
$ C18: int 7 16 11 10 10 16 12 11 10 8 ...  
$ C19: int 12 8 9 10 6 18 10 18 17 16 ...  
$ C20: int 18 6 10 7 17 12 6 19 11 8 ...  
$ C21: int 6 18 19 17 9 12 9 20 6 18 ...  
$ C22: int 13 4 14 4 14 5 17 8 16 15 ...  
$ C23: int 11 12 12 4 9 18 15 5 12 16 ...  
$ C24: int 11 5 17 7 7 9 8 13 6 14 ...  
$ C25: int 7 12 7 6 13 5 9 15 8 12 ...  
$ C26: int 7 7 12 15 9 3 3 10 14 7 ...  
$ C27: int 4 11 11 6 12 10 13 6 6 3 ...  
$ C28: int 6 5 7 4 6 3 7 10 13 8 ...  
$ C29: int 8 11 4 9 7 12 8 6 10 9 ...  
$ C30: int 8 3 2 11 7 7 2 7 11 11 ...  
$ C31: int 4 3 10 3 9 8 8 4 4 2 ...  
$ C32: int 4 5 5 5 6 4 8 9 6 5 ...  
$ C33: int 5 4 4 6 3 7 4 3 4 4 ...  
$ C34: int 7 4 2 3 2 3 2 5 7 5 ...  
$ C35: int 3 5 2 3 2 5 3 2 6 1 ...  
$ C36: int 4 5 3 4 4 4 5 5 3 4 ...  
$ C37: int 2 1 2 2 2 4 4 3 2 1 ...  
$ C38: int 3 1 2 3 0 3 1 2 1 2 ...  
$ C39: int 0 0 1 2 1 2 1 2 0 0 ...  
$ C40: int 0 1 1 1 1 1 1 1 0 0 ...
<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0</td>
<td>Min.</td>
<td>0.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0</td>
<td>1st Qu.</td>
<td>0.00</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>0</td>
<td>Median</td>
<td>1.000</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>0</td>
<td>Mean</td>
<td>0.45</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0</td>
<td>3rd Qu.</td>
<td>1.000</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>0</td>
<td>Max.</td>
<td>1.000</td>
<td>Max.</td>
</tr>
<tr>
<td>C6</td>
<td>C7</td>
<td>C8</td>
<td>C9</td>
<td>C10</td>
</tr>
<tr>
<td>Min.</td>
<td>1.00</td>
<td>Min.</td>
<td>1.0</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>2.00</td>
<td>1st Qu.</td>
<td>2.0</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>3.00</td>
<td>Median</td>
<td>4.0</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>3.15</td>
<td>Mean</td>
<td>3.8</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>4.00</td>
<td>3rd Qu.</td>
<td>5.0</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>5.00</td>
<td>Max.</td>
<td>6.0</td>
<td>Max.</td>
</tr>
<tr>
<td>C11</td>
<td>C12</td>
<td>C13</td>
<td>C14</td>
<td></td>
</tr>
<tr>
<td>Min.</td>
<td>2.00</td>
<td>Min.</td>
<td>2.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>4.00</td>
<td>1st Qu.</td>
<td>3.75</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>6.00</td>
<td>Median</td>
<td>5.50</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>5.95</td>
<td>Mean</td>
<td>5.90</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>9.00</td>
<td>3rd Qu.</td>
<td>8.00</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>10.00</td>
<td>Max.</td>
<td>11.00</td>
<td>Max.</td>
</tr>
<tr>
<td>C15</td>
<td>C16</td>
<td>C17</td>
<td>C18</td>
<td></td>
</tr>
<tr>
<td>Min.</td>
<td>3.000</td>
<td>Min.</td>
<td>3.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>5.000</td>
<td>1st Qu.</td>
<td>6.0</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>8.000</td>
<td>Median</td>
<td>10.0</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>8.367</td>
<td>Mean</td>
<td>9.5</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>12.000</td>
<td>3rd Qu.</td>
<td>13.0</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>14.000</td>
<td>Max.</td>
<td>15.0</td>
<td>Max.</td>
</tr>
<tr>
<td>C19</td>
<td>C20</td>
<td>C21</td>
<td>C22</td>
<td></td>
</tr>
<tr>
<td>Min.</td>
<td>5.00</td>
<td>Min.</td>
<td>5.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>8.00</td>
<td>1st Qu.</td>
<td>8.75</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>11.50</td>
<td>Median</td>
<td>13.00</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>11.57</td>
<td>Mean</td>
<td>12.35</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>15.00</td>
<td>3rd Qu.</td>
<td>16.00</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>18.00</td>
<td>Max.</td>
<td>19.00</td>
<td>Max.</td>
</tr>
<tr>
<td>C23</td>
<td>C24</td>
<td>C25</td>
<td>C26</td>
<td></td>
</tr>
<tr>
<td>Min.</td>
<td>4.00</td>
<td>Min.</td>
<td>4.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>7.75</td>
<td>1st Qu.</td>
<td>6.75</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>11.00</td>
<td>Median</td>
<td>10.5</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>11.03</td>
<td>Mean</td>
<td>10.17</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>15.00</td>
<td>3rd Qu.</td>
<td>13.25</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>18.00</td>
<td>Max.</td>
<td>17.00</td>
<td>Max.</td>
</tr>
<tr>
<td>C27</td>
<td>C28</td>
<td>C29</td>
<td>C30</td>
<td></td>
</tr>
<tr>
<td>Min.</td>
<td>3.00</td>
<td>Min.</td>
<td>3.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>7.00</td>
<td>1st Qu.</td>
<td>5.00</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>10.00</td>
<td>Median</td>
<td>7.00</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>9.15</td>
<td>Mean</td>
<td>7.25</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>12.00</td>
<td>3rd Qu.</td>
<td>9.00</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>14.00</td>
<td>Max.</td>
<td>13.00</td>
<td>Max.</td>
</tr>
<tr>
<td>C31</td>
<td>C32</td>
<td>C33</td>
<td>C34</td>
<td>C35</td>
</tr>
<tr>
<td>Min.</td>
<td>2.000</td>
<td>Min.</td>
<td>2.00</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>4.000</td>
<td>1st Qu.</td>
<td>4.000</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>6.000</td>
<td>Median</td>
<td>6.00</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>6.067</td>
<td>Mean</td>
<td>5.95</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>8.000</td>
<td>3rd Qu.</td>
<td>8.000</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>10.000</td>
<td>Max.</td>
<td>9.000</td>
<td>Max.</td>
</tr>
<tr>
<td>C36</td>
<td>C37</td>
<td>C38</td>
<td>C39</td>
<td>C40</td>
</tr>
<tr>
<td>Min.</td>
<td>1.000</td>
<td>Min.</td>
<td>1.000</td>
<td>Min.</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>2.000</td>
<td>1st Qu.</td>
<td>2.000</td>
<td>1st Qu.</td>
</tr>
<tr>
<td>Median</td>
<td>4.000</td>
<td>Median</td>
<td>2.000</td>
<td>Median</td>
</tr>
<tr>
<td>Mean</td>
<td>3.567</td>
<td>Mean</td>
<td>2.483</td>
<td>Mean</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>5.000</td>
<td>3rd Qu.</td>
<td>4.000</td>
<td>3rd Qu.</td>
</tr>
<tr>
<td>Max.</td>
<td>5.000</td>
<td>Max.</td>
<td>4.000</td>
<td>Max.</td>
</tr>
</tbody>
</table>
In [47]: `avg_day_inflammation <- apply(input_2, mean)`

In [48]: `avg_day_inflammation`

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>0.45</td>
</tr>
<tr>
<td>C3</td>
<td>1.11666666666667</td>
</tr>
<tr>
<td>C4</td>
<td>1.75</td>
</tr>
<tr>
<td>C5</td>
<td>2.43333333333333</td>
</tr>
<tr>
<td>C6</td>
<td>3.15</td>
</tr>
<tr>
<td>C7</td>
<td>3.8</td>
</tr>
<tr>
<td>C8</td>
<td>3.88333333333333</td>
</tr>
<tr>
<td>C9</td>
<td>5.23333333333333</td>
</tr>
<tr>
<td>C10</td>
<td>5.51666666666667</td>
</tr>
<tr>
<td>C11</td>
<td>5.95</td>
</tr>
<tr>
<td>C12</td>
<td>5.9</td>
</tr>
<tr>
<td>C13</td>
<td>8.35</td>
</tr>
<tr>
<td>C14</td>
<td>7.73333333333333</td>
</tr>
<tr>
<td>C15</td>
<td>8.36666666666667</td>
</tr>
<tr>
<td>C16</td>
<td>9.5</td>
</tr>
<tr>
<td>C17</td>
<td>9.58333333333333</td>
</tr>
<tr>
<td>C18</td>
<td>10.6333333333333</td>
</tr>
<tr>
<td>C19</td>
<td>11.5666666666667</td>
</tr>
<tr>
<td>C20</td>
<td>12.35</td>
</tr>
<tr>
<td>C21</td>
<td>13.25</td>
</tr>
<tr>
<td>C22</td>
<td>11.9666666666667</td>
</tr>
<tr>
<td>C23</td>
<td>11.0333333333333</td>
</tr>
<tr>
<td>C24</td>
<td>10.1666666666667</td>
</tr>
<tr>
<td>C25</td>
<td>10</td>
</tr>
<tr>
<td>C26</td>
<td>8.66666666666667</td>
</tr>
<tr>
<td>C27</td>
<td>9.15</td>
</tr>
<tr>
<td>C28</td>
<td>7.25</td>
</tr>
<tr>
<td>C29</td>
<td>7.33333333333333</td>
</tr>
<tr>
<td>C30</td>
<td>6.58333333333333</td>
</tr>
<tr>
<td>C31</td>
<td>6.06666666666667</td>
</tr>
<tr>
<td>C32</td>
<td>5.95</td>
</tr>
<tr>
<td>C33</td>
<td>5.11666666666667</td>
</tr>
<tr>
<td>C34</td>
<td>3.6</td>
</tr>
<tr>
<td>C35</td>
<td>3.3</td>
</tr>
<tr>
<td>C36</td>
<td>3.56666666666667</td>
</tr>
<tr>
<td>C37</td>
<td>2.48333333333333</td>
</tr>
<tr>
<td>C38</td>
<td>1.5</td>
</tr>
<tr>
<td>C39</td>
<td>1.13333333333333</td>
</tr>
<tr>
<td>C40</td>
<td>0.56666666666667</td>
</tr>
</tbody>
</table>
In [49]: plot(avg_day_inflammation)

In [50]: max_day_inflammation <- apply(input, 2, max)
In this section we download a dataset based on gapminder data and generate informative and sophisticated plots.

```r
plot(max_day_inflammation)
```

**More sophisticated plots**

R can generate complicated plots with high quality. User can customize R functions and take advantage of R packages to generate publication ready plots. In this section we download a dataset based on gapminder data and generate informative and sophisticated plots.

```r
download.file("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd")
gapminder <- read.csv("gapminder-FiveYearData.csv")
```
In this example, we load a package called “ggplot2” to create our plots.

```r
library("ggplot2")
```

The `str()` function shows that `gapminder` data includes:
- `country`: factor with 142 levels
- `year`: ranges from 1952 to 2007 in increments of 5 years
- `lifeExp`: life expectancy at birth, in years
- `pop`: population
- `gdpPercap`: GDP per capita, where GDP is gross domestic product.
In [56]: `ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) + geom_point()`
Functions in R

If there are more than one data set to be analyzed and the operations will be repeated, one can write a function so that we can repeat several operations with a single command.

Calculating BMI by writing a function

```r
BMI_Calculator_English_System <- function(weight, height)
{
    BMI <- (weight/ height/ height) * 703
    return(BMI)
}
```

```r
BMI_Calculator_English_System(145, 60)
```

28.31527777778
Loops in R

Loops are useful commands for iterations. In R those can be used to operate on items of vectors, matrices, and data frames. Let's create a function that uses a “for” loop:

```r
math_ops <- function(a, b)
{
  c <- matrix(nrow=nrow(a),ncol=3,data=NA)
  colnames(c) <- c("sum","multiply","divid")
  for (i in 1:nrow(a))
  {
    c[i,1] <- a[i] + b[i]
    c[i,2] <- a[i] * b[i]
    c[i,3] <- a[i] / b[i]
    print(c)
  }
}
```

```r
a <- matrix(nrow=4,ncol=1,data=c(1,10,100,50))
b <- matrix(nrow=4,ncol=1,data=c(4,8,12,16))
math_ops(a,b)
```

Conditional Statements in R

We use a conditional statement to make a choice based on values of a variable. R syntax for conditional statements is:

```r
if (condition) to start a conditional statement
else if (condition) to provide additional tests
else to provide a default
```

The bodies of conditional statements must be surrounded by curly braces {}.

- Use == to test for equality.
- X & Y is only true if both X and Y are true.
- X | Y is true if either X or Y, or both, are true.

```r
x <- max(input[1,])
y <- max(input[15,])
```
Recall the gapminder data that we used in above examples. Let's write a code that calculates the average population for all the countries. We will use a loop and conditional statement to achieve this goal. Let's look at the first few rows of the data.

```python
In [67]:
if (x > y):
   print("x is greater")
else if (y > x):
   print("y is greater")
else:
   print("x and y are equal")

print(c("x equals to: ",x))
print(c("y equals to: ",y))
```

[1] "x is greater"
[1] "x equals to: "18"
[1] "y equals to: "17"

Recall the gapminder data that we used in above examples. Let's write a code that calculates the average population for all the countries. We will use a loop and conditional statement to achieve this goal. Let's look at the first few rows of the data.
Calculating the average population for each country and saving the results:

<table>
<thead>
<tr>
<th>country</th>
<th>year</th>
<th>pop</th>
<th>continent</th>
<th>lifeExp</th>
<th>gdpPercap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>1952</td>
<td>8425333</td>
<td>Asia</td>
<td>28.801</td>
<td>779.4453</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1957</td>
<td>9240934</td>
<td>Asia</td>
<td>30.332</td>
<td>820.8530</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1962</td>
<td>10267083</td>
<td>Asia</td>
<td>31.997</td>
<td>853.1007</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1967</td>
<td>11537966</td>
<td>Asia</td>
<td>34.020</td>
<td>836.1971</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1972</td>
<td>13079460</td>
<td>Asia</td>
<td>36.088</td>
<td>739.9811</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1977</td>
<td>14880372</td>
<td>Asia</td>
<td>38.438</td>
<td>786.1134</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1982</td>
<td>12881816</td>
<td>Asia</td>
<td>39.854</td>
<td>978.0114</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1987</td>
<td>13867957</td>
<td>Asia</td>
<td>40.822</td>
<td>852.3959</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1992</td>
<td>16317921</td>
<td>Asia</td>
<td>41.674</td>
<td>649.3414</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>1997</td>
<td>22227415</td>
<td>Asia</td>
<td>41.763</td>
<td>635.3414</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>2002</td>
<td>25268405</td>
<td>Asia</td>
<td>42.129</td>
<td>726.7341</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>2007</td>
<td>31889923</td>
<td>Asia</td>
<td>43.828</td>
<td>974.5803</td>
</tr>
<tr>
<td>Albania</td>
<td>1952</td>
<td>1282697</td>
<td>Europe</td>
<td>55.230</td>
<td>1601.0561</td>
</tr>
<tr>
<td>Albania</td>
<td>1957</td>
<td>1476505</td>
<td>Europe</td>
<td>59.280</td>
<td>1942.2842</td>
</tr>
<tr>
<td>Albania</td>
<td>1962</td>
<td>1728137</td>
<td>Europe</td>
<td>64.820</td>
<td>2312.8890</td>
</tr>
<tr>
<td>Albania</td>
<td>1967</td>
<td>1984060</td>
<td>Europe</td>
<td>66.220</td>
<td>2760.1969</td>
</tr>
<tr>
<td>Albania</td>
<td>1972</td>
<td>2263554</td>
<td>Europe</td>
<td>67.690</td>
<td>3313.4222</td>
</tr>
<tr>
<td>Albania</td>
<td>1977</td>
<td>2509048</td>
<td>Europe</td>
<td>68.930</td>
<td>3533.0039</td>
</tr>
<tr>
<td>Albania</td>
<td>1982</td>
<td>2780097</td>
<td>Europe</td>
<td>70.420</td>
<td>3630.8807</td>
</tr>
<tr>
<td>Albania</td>
<td>1987</td>
<td>3075321</td>
<td>Europe</td>
<td>72.000</td>
<td>3738.9327</td>
</tr>
<tr>
<td>Albania</td>
<td>1992</td>
<td>3326498</td>
<td>Europe</td>
<td>71.581</td>
<td>2497.4379</td>
</tr>
<tr>
<td>Albania</td>
<td>1997</td>
<td>3428038</td>
<td>Europe</td>
<td>72.950</td>
<td>3193.0546</td>
</tr>
<tr>
<td>Albania</td>
<td>2002</td>
<td>3508512</td>
<td>Europe</td>
<td>75.851</td>
<td>4604.2117</td>
</tr>
<tr>
<td>Albania</td>
<td>2007</td>
<td>3600523</td>
<td>Europe</td>
<td>76.423</td>
<td>5937.0295</td>
</tr>
<tr>
<td>Algeria</td>
<td>1952</td>
<td>9279525</td>
<td>Africa</td>
<td>43.077</td>
<td>2449.0082</td>
</tr>
</tbody>
</table>

Calculating the average population for each country and saving the results:
In [69]:

#Making template data structures for output data

gapminder_country_pop_average <- matrix(nrow=1, data=c("country","average pop"))
country_mean <- matrix(nrow=1, ncol=2)

#Initiating the start variable
start <- 1

#For loop through the countries
for (c in 1:(nrow(gapminder)-1))
{
  if(gapminder[c,1]!=gapminder[c+1,1])
  {
    end <- c
    mean <- mean(gapminder[start:end,3])
    country_mean[1,1] <- as.character(gapminder[c,1])
    country_mean[1,2] <- as.numeric(mean)
    gapminder_country_pop_average <- rbind(gapminder_country_pop_average, country_mean)
    start <- c+1
  }
}

#Getting the mean for last country
mean <- mean(gapminder[start:nrow(gapminder),3])
country_mean[1,1] <- as.character(gapminder[c,1])
country_mean[1,2] <- as.numeric(mean)
gapminder_country_pop_average <- rbind(gapminder_country_pop_average, country_mean)

print(gapminder_country_pop_average)
write.csv(gapminder_country_pop_average, "gapminder_country_pop_average.csv")


Useful Commands

match()
sessionInfo()
ls()
list.files()
In [70]:
(a <- 1:10)
(b <- 5:14)
print(match(a, b))

1 2 3 4 5 6 7 8 9 10
5 6 7 8 9 10 11 12 13 14
[1] NA NA NA NA  1  2  3  4  5  6

In [71]:

sessionInfo()

R version 3.4.2 (2017-09-28)
Platform: x86_64-conda_cos6-linux-gnu (64-bit)
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS: /opt/anaconda2/5.0.0/lib/R/lib/libRblas.so
LAPACK: /opt/anaconda2/5.0.0/lib/R/lib/libRlapack.so

locale:
[1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8       LC_NAME=C
[9] LC_ADDRESS=C               LC_TELEPHONE=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_2.2.1

loaded via a namespace (and not attached):
[1] Rcpp_0.12.13     digest_0.6.12    crayon_1.3.4     IRdisplay_0.4.4
[5] plyr_1.8.4       repr_0.12.0      grid_3.4.2       R6_2.2.2
[9] jsonlite_1.5     gtable_0.2.0     magrittr_1.5     scales_0.5.0
[13] evaluate_0.10.1  rlang_0.1.2      stringi_1.1.5    lazyeval_0.2.0
[17] pdb2XM_0.2-6     tibble_1.3.4

In [72]:

'a' 'avg_day_inflammation' 'b' 'BMI_Calculator_English_System' 'c' 'country_mean' 'df' 'end'
'gapminder' 'gapminder_country_pop_average' 'input' 'm' 'math_ops' 'max_day_inflammation'
'mean' 'start' 'v' 'x' 'y'

In [73]:

list.files()

'Fall_2017_Intro_Perl' 'Fall_2017_Introduction_to_Python.ipynb' 'gapminder_country_pop_average.csv'
'gapminder-FiveYearData.csv' 'inflammation-01.csv' 'Intro_to_DL_with_TensorFlow' 'intro_to_linux'
'Introduction_to_Database' 'Introduction_to_Python' 'Introduction_to_R_HPRC_TAMU_April2018.ipynb'
'Introduction_to_R_HPRC_TAMU_December2017.ipynb' 'Introduction_to_Scientific_Python' 'perli'
'Spring_2018_Perl' 'Spring_2018_Python4Matlab_Users'

R Packages
It is possible to add functions to R by writing a package, or by obtaining a package written by someone else. As of this writing, there are over 10,000 packages available on CRAN (the comprehensive R archive network). R and RStudio have functionality for managing packages:

- You can see what packages are installed by typing `installed.packages()`.
- You can install packages by typing `install.packages("packagename")`, where `packagename` is the package name, in quotes.
- You can update installed packages by typing `update.packages()`.
- You can remove a package with `remove.packages("packagename")`.
- You can make a package available for use with `library(packagename)`.