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 (https://CRAN.R-project.org/mirrors.html)](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 [R documetation].

Basic Usage of R

Using R as a Calculator

R can be used as a simple calculator.

In [1]:
```
100*4.5
```

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

In [3]:
```
3*(2+10.6)
```

37.8

In [4]:
```
3*(2+10.6)/2
```

18.9

In [5]:
```
3*2+10.6/2
```

11.3

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:

In [6]:
```
?Math
```

In [7]:
```
pi
```

3.14159265358979

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

1

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

1

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

3.32192809488736

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

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)
```

In [12]:
```
#Assignment to a variable
x <- 1:5
```

In [13]:
```
(x)
```

1 2 3 4 5

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

```
[1] 1
```

In [15]:
```
b=T
(b)
```

TRUE

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

```
[1] "This is a test"
```

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 [17]:
```
y <- 5:15
(y)
class(y)
```

5 6 7 8 9 10 11 12 13 14 15

'integer'

In [18]:
```
length(y)
```

11

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 [19]:
```
#making a vector
v<- vector("character", length=5)
typeof(v)
```

'character'

In [20]:
```
class(v)
```

'character'

In [21]:
```
v<- c("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] [,2] [,3] [,4] [,5]
[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	1	11
b	2	12
c	3	13
d	4	14
e	5	15
f	6	16
g	7	17
h	8	18
i	9	19
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[,1])
```

'factor'

Working with Data

In [32]:
```
download.file(url="https://raw.githubusercontent.com/swcarpentry/files/master/inflamm
```

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

In [34]:
```
head(input)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	⋯	V31	V32	V33	V34	V35	V36	V37	V38	V39	V40
0	0	1	3	1	2	4	7	8	3	⋯	4	4	5	7	3	4	2	3	0	0
0	1	2	1	2	1	3	2	2	6	⋯	3	5	4	4	5	5	1	1	0	1
0	1	1	3	3	2	6	2	5	9	⋯	10	5	4	2	2	3	2	2	1	1
0	0	2	0	4	2	2	1	6	7	⋯	3	5	6	3	3	4	2	3	2	1
0	1	1	3	3	1	3	5	2	4	⋯	9	6	3	2	2	4	2	0	1	1
0	0	1	2	2	4	2	1	6	4	⋯	8	4	7	3	5	4	4	3	2	1

In [35]:
```
getwd()
```

'/home/noushin'

In [36]:
```
dim(input)
```

60 40

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

1

In [38]:
```
input[4,37]
```

2

In [39]:
```
input[1:6,3]
```

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 0 ...
 $ C2 : int  0 1 1 0 1 0 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 2 ...
 $ 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 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 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 ...
```

In [46]:
```
summary(input)
```

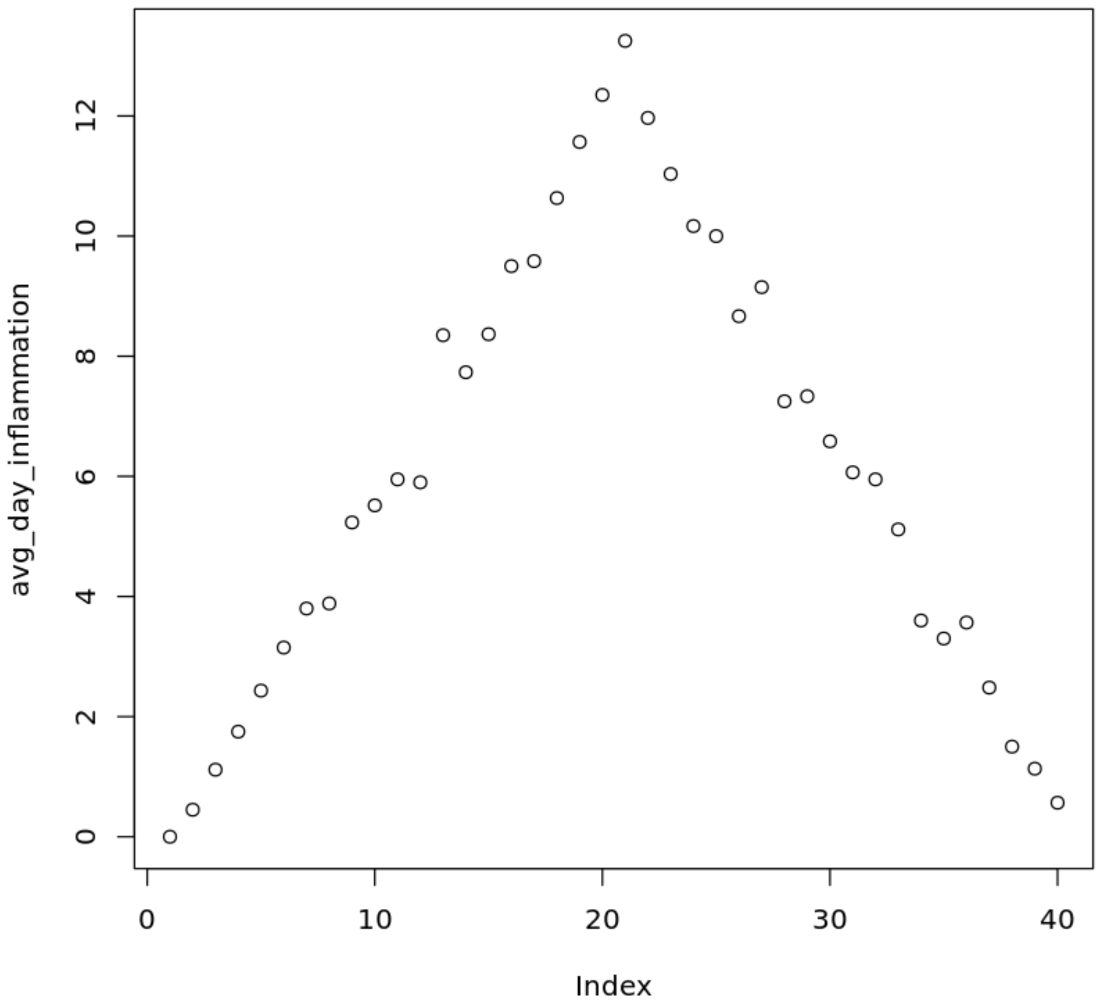
```
       C1          C2              C3               C4              C5       
 Min.   :0   Min.   :0.00   Min.   :0.000   Min.   :0.00   Min.   :1.000  
 1st Qu.:0   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:1.00   1st Qu.:1.000  
 Median :0   Median :0.00   Median :1.000   Median :2.00   Median :2.000  
 Mean   :0   Mean   :0.45   Mean   :1.117   Mean   :1.75   Mean   :2.433  
 3rd Qu.:0   3rd Qu.:1.00   3rd Qu.:2.000   3rd Qu.:3.00   3rd Qu.:3.000  
 Max.   :0   Max.   :1.00   Max.   :2.000   Max.   :3.00   Max.   :4.000  
       C6             C7            C8              C9             C10       
 Min.   :1.00   Min.   :1.0   Min.   :1.000   Min.   :2.000   Min.   :2.000  
 1st Qu.:2.00   1st Qu.:2.0   1st Qu.:2.000   1st Qu.:4.000   1st Qu.:3.750  
 Median :3.00   Median :4.0   Median :4.000   Median :5.000   Median :6.000  
 Mean   :3.15   Mean   :3.8   Mean   :3.883   Mean   :5.233   Mean   :5.517  
 3rd Qu.:4.00   3rd Qu.:5.0   3rd Qu.:5.250   3rd Qu.:7.000   3rd Qu.:7.000  
 Max.   :5.00   Max.   :6.0   Max.   :7.000   Max.   :8.000   Max.   :9.000  
      C11             C12             C13             C14        
 Min.   : 2.00   Min.   : 2.00   Min.   : 3.00   Min.   : 3.000  
 1st Qu.: 4.00   1st Qu.: 3.75   1st Qu.: 5.00   1st Qu.: 5.000  
 Median : 6.00   Median : 5.50   Median : 9.50   Median : 8.000  
 Mean   : 5.95   Mean   : 5.90   Mean   : 8.35   Mean   : 7.733  
 3rd Qu.: 9.00   3rd Qu.: 8.00   3rd Qu.:11.00   3rd Qu.:10.000  
 Max.   :10.00   Max.   :11.00   Max.   :12.00   Max.   :13.000  
      C15              C16             C17              C18       
 Min.   : 3.000   Min.   : 3.0   Min.   : 4.000   Min.   : 5.00  
 1st Qu.: 5.000   1st Qu.: 6.0   1st Qu.: 6.750   1st Qu.: 7.75  
 Median : 8.000   Median :10.0   Median : 8.500   Median :11.00  
 Mean   : 8.367   Mean   : 9.5   Mean   : 9.583   Mean   :10.63  
 3rd Qu.:12.000   3rd Qu.:13.0   3rd Qu.:13.000   3rd Qu.:13.00  
 Max.   :14.000   Max.   :15.0   Max.   :16.000   Max.   :17.00  
      C19             C20             C21             C22       
 Min.   : 5.00   Min.   : 5.00   Min.   : 5.00   Min.   : 4.00  
 1st Qu.: 8.00   1st Qu.: 8.75   1st Qu.: 9.00   1st Qu.: 8.00  
 Median :11.50   Median :13.00   Median :14.00   Median :13.00  
 Mean   :11.57   Mean   :12.35   Mean   :13.25   Mean   :11.97  
 3rd Qu.:15.00   3rd Qu.:16.00   3rd Qu.:16.25   3rd Qu.:15.25  
 Max.   :18.00   Max.   :19.00   Max.   :20.00   Max.   :19.00  
      C23             C24             C25            C26        
 Min.   : 4.00   Min.   : 4.00   Min.   : 4.0   Min.   : 3.000  
 1st Qu.: 7.75   1st Qu.: 6.75   1st Qu.: 7.0   1st Qu.: 5.750  
 Median :11.00   Median :10.00   Median :10.5   Median : 9.000  
 Mean   :11.03   Mean   :10.17   Mean   :10.0   Mean   : 8.667  
 3rd Qu.:15.00   3rd Qu.:13.25   3rd Qu.:13.0   3rd Qu.:12.000  
 Max.   :18.00   Max.   :17.00   Max.   :16.0   Max.   :15.000  
      C27             C28             C29              C30        
 Min.   : 3.00   Min.   : 3.00   Min.   : 3.000   Min.   : 2.000  
 1st Qu.: 7.00   1st Qu.: 5.00   1st Qu.: 5.000   1st Qu.: 3.000  
 Median :10.00   Median : 7.00   Median : 7.000   Median : 7.000  
 Mean   : 9.15   Mean   : 7.25   Mean   : 7.333   Mean   : 6.583  
 3rd Qu.:12.00   3rd Qu.: 9.00   3rd Qu.:10.000   3rd Qu.:10.000  
 Max.   :14.00   Max.   :13.00   Max.   :12.000   Max.   :11.000  
      C31              C32             C33             C34           C35     
 Min.   : 2.000   Min.   :2.00   Min.   :2.000   Min.   :1.0   Min.   :1.0  
 1st Qu.: 4.000   1st Qu.:4.00   1st Qu.:4.000   1st Qu.:2.0   1st Qu.:2.0  
 Median : 6.000   Median :6.00   Median :5.000   Median :4.0   Median :3.0  
 Mean   : 6.067   Mean   :5.95   Mean   :5.117   Mean   :3.6   Mean   :3.3  
 3rd Qu.: 8.000   3rd Qu.:8.00   3rd Qu.:6.000   3rd Qu.:5.0   3rd Qu.:5.0  
 Max.   :10.000   Max.   :9.00   Max.   :8.000   Max.   :7.0   Max.   :6.0  
      C36             C37             C38           C39             C40        
 Min.   :1.000   Min.   :1.000   Min.   :0.0   Min.   :0.000   Min.   :0.0000  
 1st Qu.:2.000   1st Qu.:2.000   1st Qu.:0.0   1st Qu.:0.000   1st Qu.:0.0000  
 Median :4.000   Median :2.000   Median :1.0   Median :1.000   Median :1.0000  
 Mean   :3.567   Mean   :2.483   Mean   :1.5   Mean   :1.133   Mean   :0.5667  
 3rd Qu.:5.000   3rd Qu.:4.000   3rd Qu.:3.0   3rd Qu.:2.000   3rd Qu.:1.0000  
 Max.   :5.000   Max.   :4.000   Max.   :3.0   Max.   :2.000   Max.   :1.0000  
```

In [47]:
```
avg_day_inflammation <- apply(input, 2, mean)
```

In [48]:
```
avg_day_inflammation
```

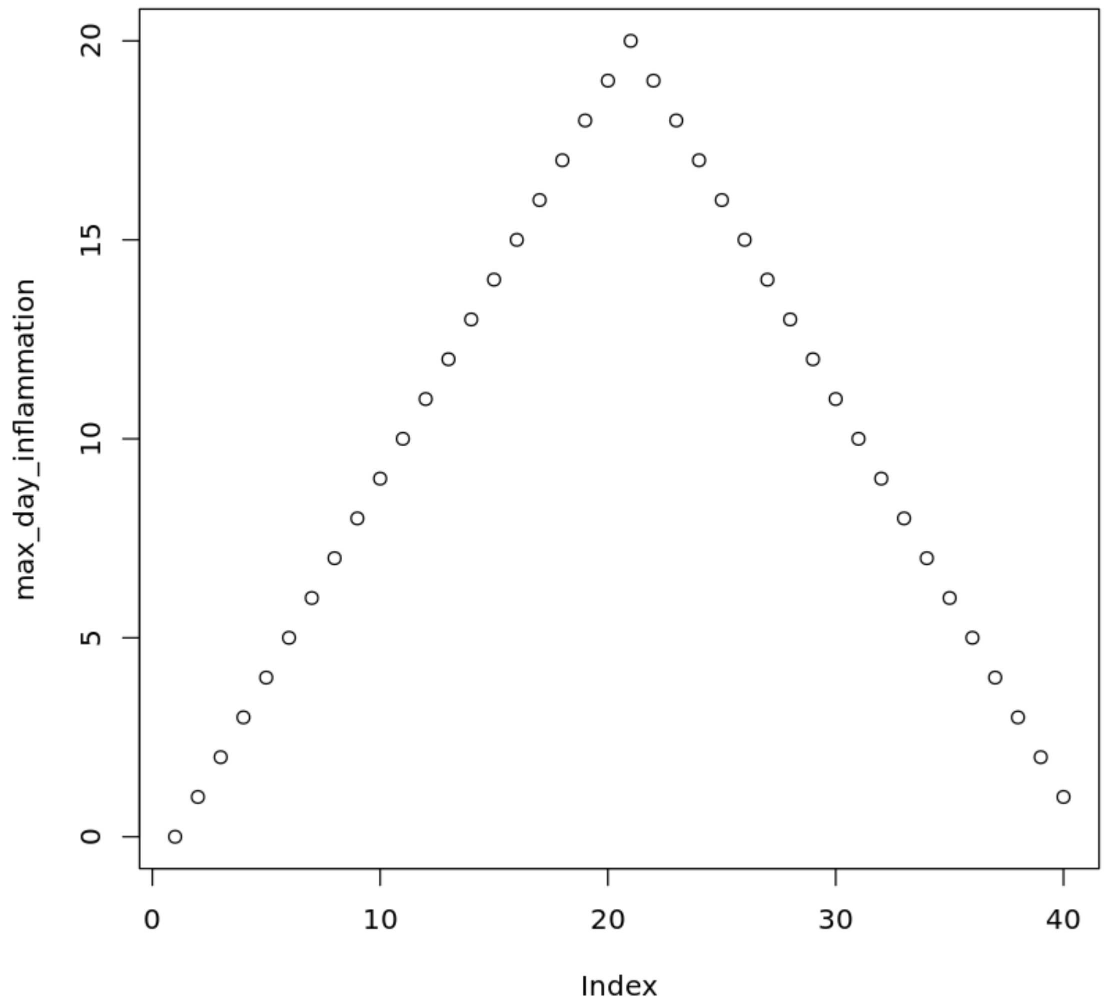
C1	0
C2	0.45
C3	1.11666666666667
C4	1.75
C5	2.43333333333333
C6	3.15
C7	3.8
C8	3.88333333333333
C9	5.23333333333333
C10	5.51666666666667
C11	5.95
C12	5.9
C13	8.35
C14	7.73333333333333
C15	8.36666666666667
C16	9.5
C17	9.58333333333333
C18	10.6333333333333
C19	11.5666666666667
C20	12.35
C21	13.25
C22	11.9666666666667
C23	11.0333333333333
C24	10.1666666666667
C25	10
C26	8.66666666666667
C27	9.15
C28	7.25
C29	7.33333333333333
C30	6.58333333333333
C31	6.06666666666667
C32	5.95
C33	5.11666666666667
C34	3.6
C35	3.3
C36	3.56666666666667
C37	2.48333333333333
C38	1.5
C39	1.13333333333333
C40	0.566666666666667

In [49]:
```
plot(avg_day_inflammation)
```



In [50]:
```
max_day_inflammation <- apply(input, 2, max)
```

In [51]:
```
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.

In [52]:
```
download.file("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pa
```

In [53]:
```
gapminder <- read.csv("gapminder_FiveYearData.csv")
```

In [54]:
```
str(gapminder)
```

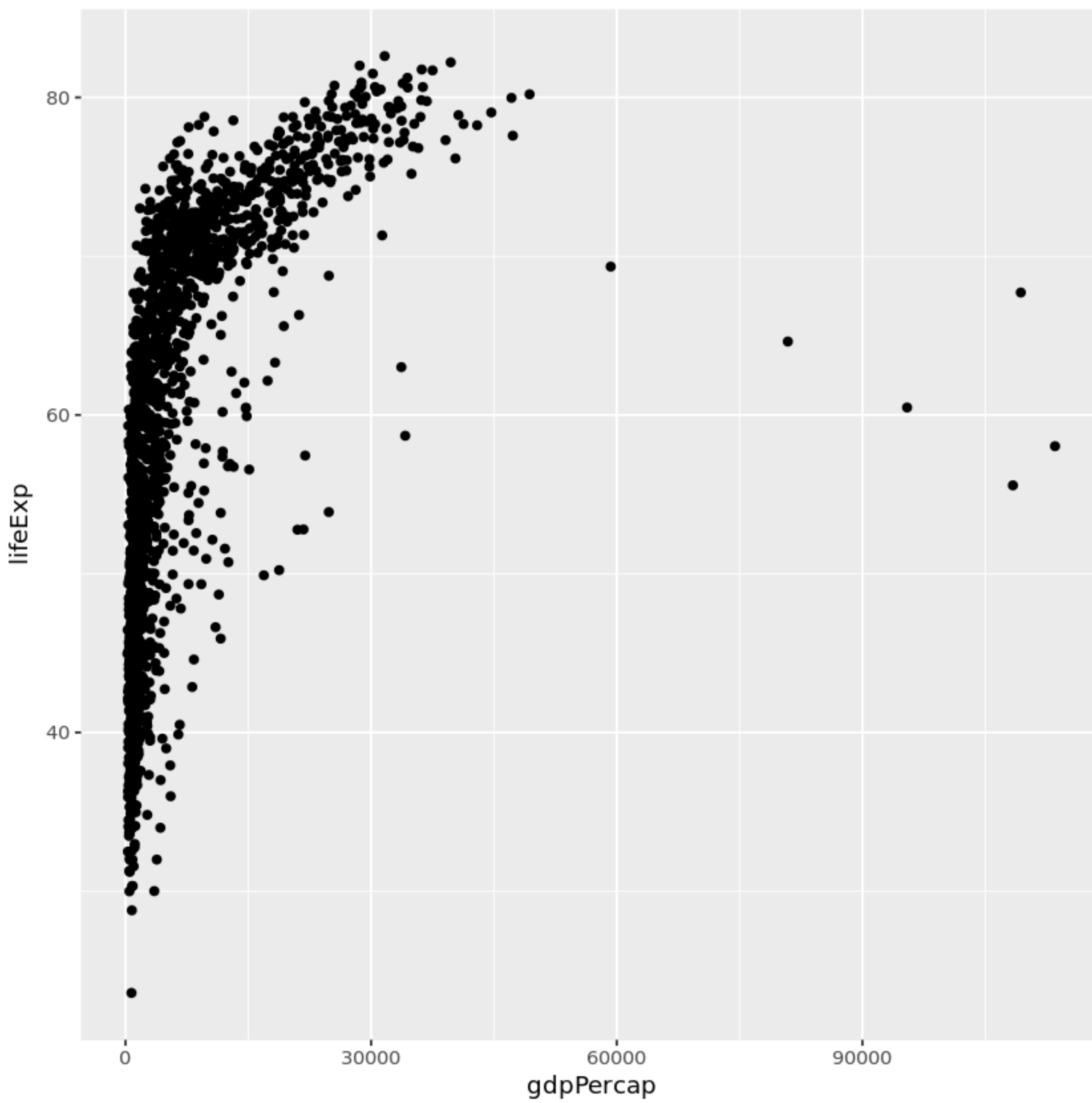
```
'data.frame':	1704 obs. of  6 variables:
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
 $ gdpPercap: num  779 821 853 836 740 ...
```

```
The str function shows that gapminder data includes
country: factor with 142 levels
continent: factor with 5 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
```

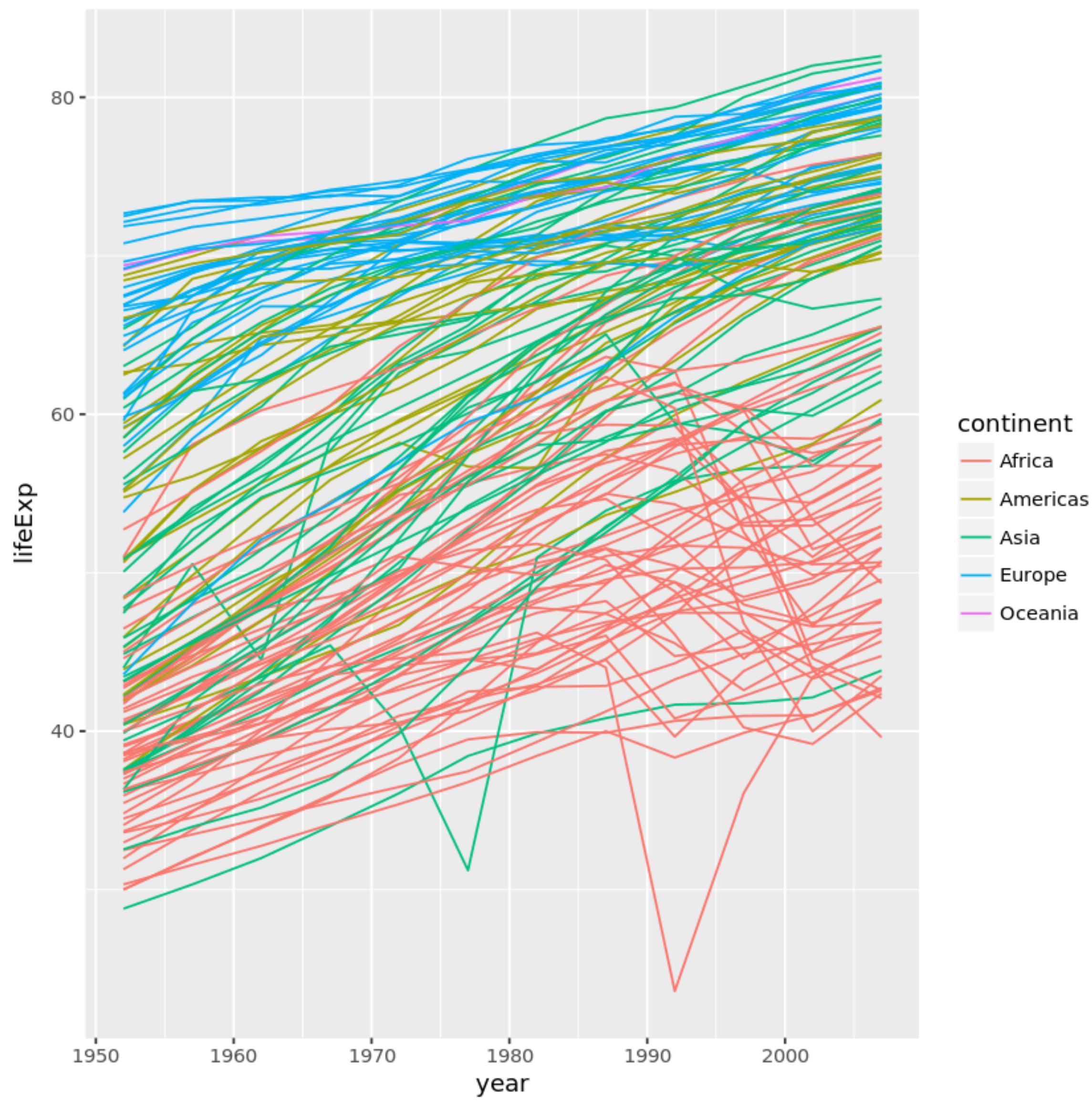
In this example, we load a package called "ggplot2" to create our plots.

In [55]:
```
library("ggplot2")
```

In [56]:
```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) + geom_point()
```



In [57]:
```
ggplot(data = gapminder, aes(x=year, y=lifeExp, by=country, color=continent)) +   geo
```



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

In [58]:
```
BMI_Calculator_English_System <- function(weight, height)
{
    BMI <- (weight/ height/ height) *703
    return(BMI)
}
```

In [59]:
```
BMI_Calculator_English_System(145,60)
```

28.3152777777778

Loops in R

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

In [60]:
```
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)
}
```

In [61]:
```
a <- matrix(nrow=4,ncol=1,data=c(1,10,100,50))
```

In [62]:
```
b <- matrix(nrow=4,ncol=1,c(4,8,12,16))
```

In [63]:
```
math_ops(a,b)
```

```
     sum multiply    divid
[1,]   5        4 0.250000
[2,]  18       80 1.250000
[3,] 112     1200 8.333333
[4,]  66      800 3.125000
```

In [64]:
```
#useful function especially to be added at the end of your functions to save the exac
#of running the function
date()
```

'Thu Apr 19 23:57:03 2018'

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:

```
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.
```

In [65]:
```
x <- max(input[1,1])
```

In [66]:
```
y <- max(input[15,1])
```

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 gapmider 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.

In [68]:
```
head(gapminder,25)
```

country	year	pop	continent	lifeExp	gdpPercap
Afghanistan	1952	8425333	Asia	28.801	779.4453
Afghanistan	1957	9240934	Asia	30.332	820.8530
Afghanistan	1962	10267083	Asia	31.997	853.1007
Afghanistan	1967	11537966	Asia	34.020	836.1971
Afghanistan	1972	13079460	Asia	36.088	739.9811
Afghanistan	1977	14880372	Asia	38.438	786.1134
Afghanistan	1982	12881816	Asia	39.854	978.0114
Afghanistan	1987	13867957	Asia	40.822	852.3959
Afghanistan	1992	16317921	Asia	41.674	649.3414
Afghanistan	1997	22227415	Asia	41.763	635.3414
Afghanistan	2002	25268405	Asia	42.129	726.7341
Afghanistan	2007	31889923	Asia	43.828	974.5803
Albania	1952	1282697	Europe	55.230	1601.0561
Albania	1957	1476505	Europe	59.280	1942.2842
Albania	1962	1728137	Europe	64.820	2312.8890
Albania	1967	1984060	Europe	66.220	2760.1969
Albania	1972	2263554	Europe	67.690	3313.4222
Albania	1977	2509048	Europe	68.930	3533.0039
Albania	1982	2780097	Europe	70.420	3630.8807
Albania	1987	3075321	Europe	72.000	3738.9327
Albania	1992	3326498	Europe	71.581	2497.4379
Albania	1997	3428038	Europe	72.950	3193.0546
Albania	2002	3508512	Europe	75.651	4604.2117
Albania	2007	3600523	Europe	76.423	5937.0295
Algeria	1952	9279525	Africa	43.077	2449.0082

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_
        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")
```

```
      [,1]                     [,2]
 [1,] "country"                "average pop"
 [2,] "Afghanistan"            "15823715.4166667"
 [3,] "Albania"                "2580249.16666667"
 [4,] "Algeria"                "19875406.1666667"
 [5,] "Angola"                 "7309390.08333333"
 [6,] "Argentina"              "28602239.9166667"
 [7,] "Australia"              "14649312.5"
 [8,] "Austria"                "7583298.41666667"
 [9,] "Bahrain"                "373913.166666667"
[10,] "Bangladesh"             "90755395.3333333"
[11,] "Belgium"                "9725118.66666667"
[12,] "Benin"                  "4017496.66666667"
[13,] "Bolivia"                "5610395.16666667"
[14,] "Bosnia and Herzegovina" "3816524.75"
[15,] "Botswana"               "971186.166666667"
[16,] "Brazil"                 "122312126.666667"
[17,] "Bulgaria"               "8182985.33333333"
[18,] "Burkina Faso"           "7548677.25"
[19,] "Burundi"                "4651608.33333333"
```

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
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=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] uuid_0.1-2       IRkernel_0.8.9   labeling_0.3     tools_3.4.2
[21] stringr_1.2.0    munsell_0.4.3    compiler_3.4.2   colorspace_1.3-2
[25] pbdZMQ_0.2-6     tibble_1.3.4
```

In [72]:
```
ls()
```

'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)