Intro to R Programming

What is R

- Programming Language for Statistical Computing
- Widely used among statisticians and data miners for developing statistical software and data analysis
- Can perform a variety of Visual and Modeling Functions useful for conceptualizing data analysis***

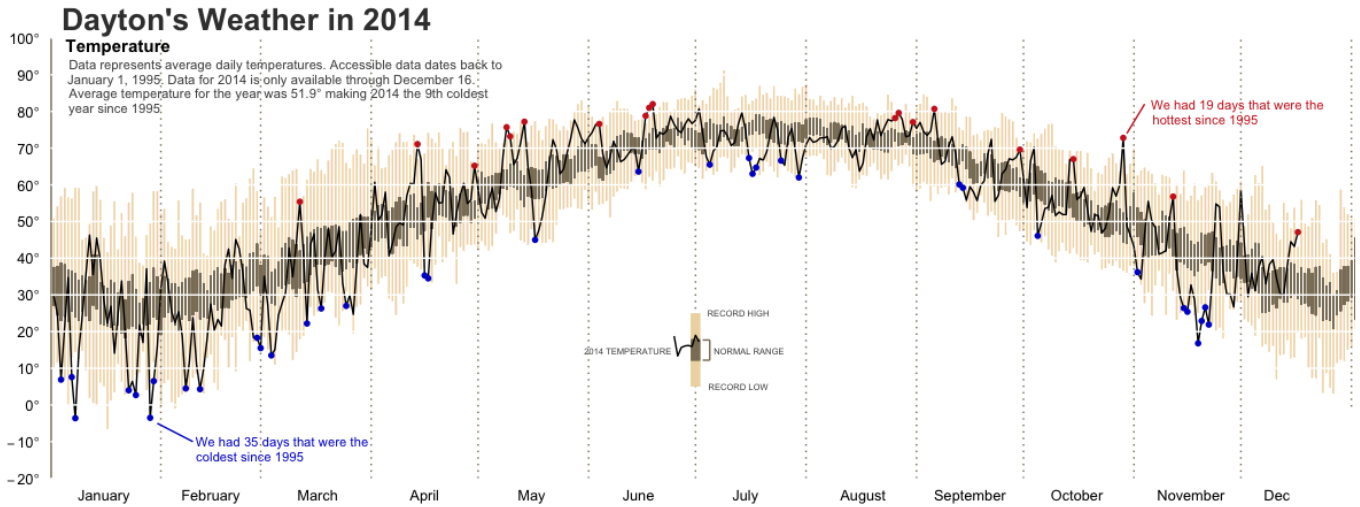
R vs Excel

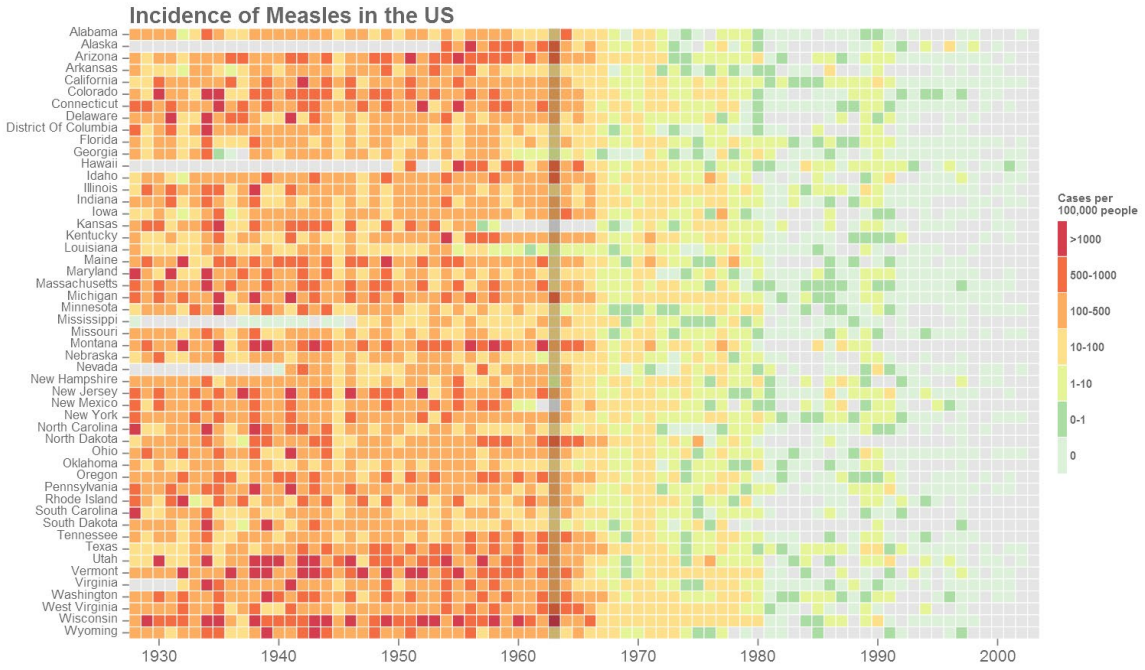
- R utilizes a lot of datasets that are imported from excel
- R can handle much larger data sets
- R is used for more advanced statistical analysis and data visualization
- Libraries in R

R vs Python

- Python heavily used in software engineering and Web Design while R is more specific to statistical modeling and data analysis
- R has a more comprehensive data modeling and visualization base
- Both can perform similar functions

Examples of Data Analysis with R

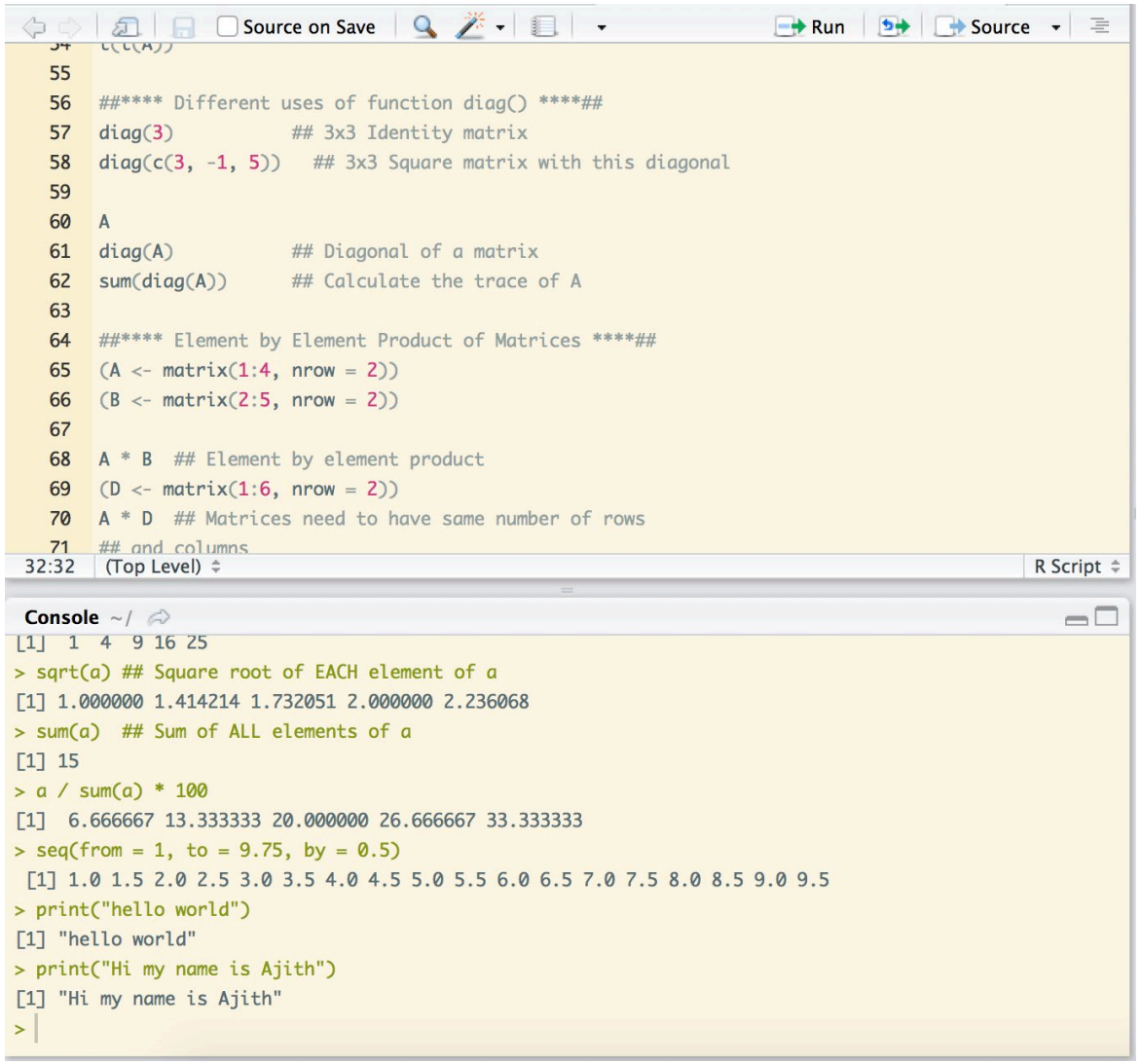




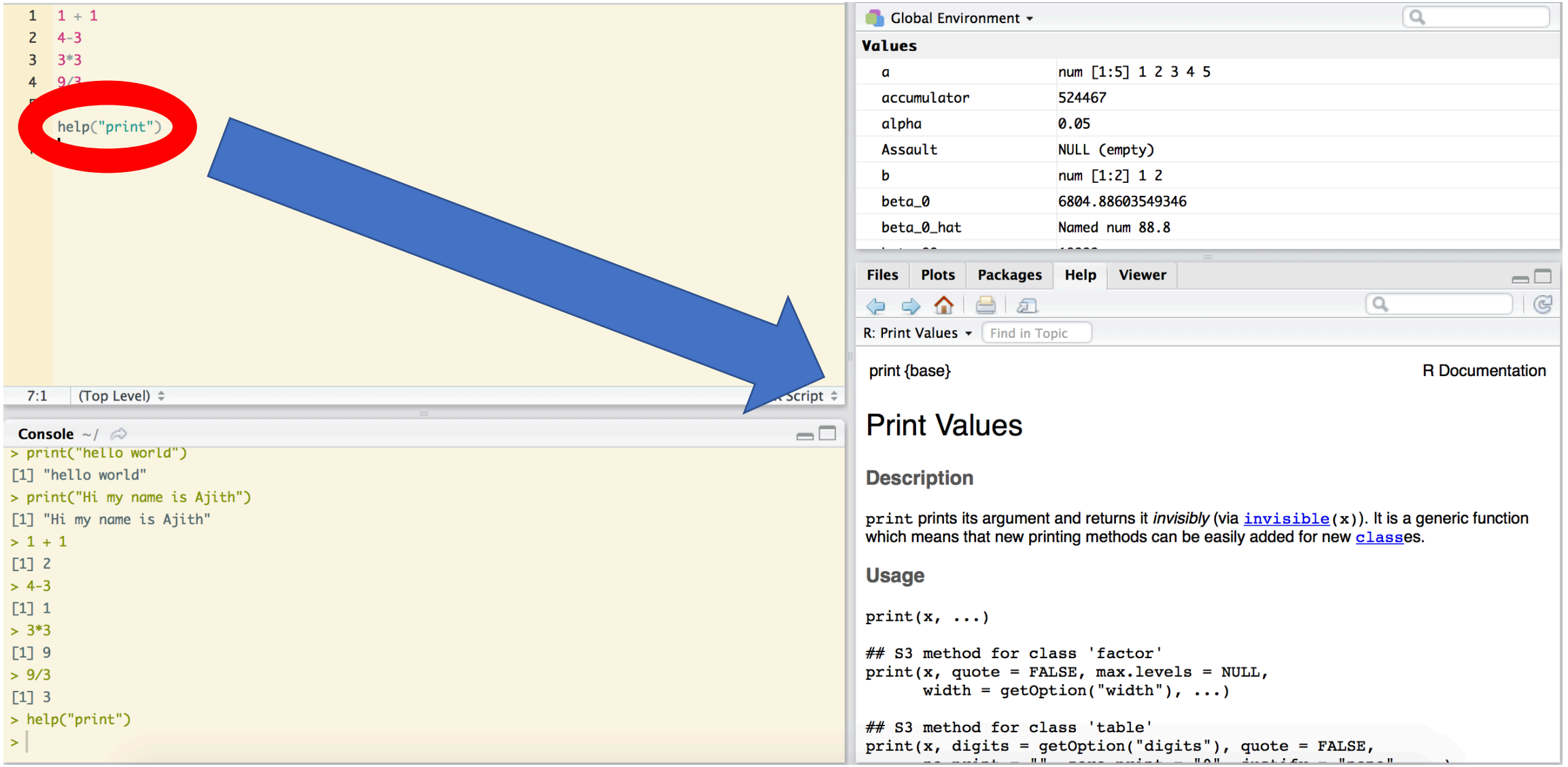
Layout of R

Script

Command Console



How to seek help in R



Printing Text

```
1  print("hello world")
2  print("Hi my name is Ajith")
```

```
> print("hello world")
[1] "hello world"
> print("Hi my name is Ajith")
[1] "Hi my name is Ajith"
```

Answer

```
1  print("Hello my name is Ajith and I was born in Singapore")
```

```
> print("Hello my name is Ajith and I was born in Singapore")
[1] "Hello my name is Ajith and I was born in Singapore"
```

```
1  "Hello my name is Ajith and I was born in Singapore"
```

```
> "Hello my name is Ajith and I was born in Singapore"
[1] "Hello my name is Ajith and I was born in Singapore"
```

Calculations

```
1  1 + 1
2  4-3
3  3*3
4  9/3
```

```
> 1 + 1
[1] 2
> 4-3
[1] 1
> 3*3
[1] 9
> 9/3
[1] 3
```

Math Operators

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Data Types

- Logical – True/False
- Integer – 4, 6, 2
- Numeric – 3.4, 566, 2.34
- Character – “a”, “hello”

Assigning Variables in R

- Assign variables to any data type in R for efficiency and data manipulation (take into account data type when doing so)

```
1  a <- 1
2  b <- 2
```

```
> a <- 1
> b <- 2
```

```
4  r <- "Hello"
5  t <- "World"
```

```
> r <- "Hello"
> t <- "World"
```

Combining Assigned Variables

- Make sure to take into account data type when doing this***

```
1  a <- 1
2  b <- 2
3  a+b
4  r <- "Hello"
5  t <- "World"
6  paste(r,t, sep = " ")
```

```
> a <- 1
> b <- 2
> a+b
[1] 3
> r <- "Hello"
> t <- "World"
> paste(r,t, sep = " ")
[1] "Hello World"
```

Objects in R

- Vectors
- Arrays
- Matrices
- Lists
- Dataframes

Vectors

- A vector is a single entity consisting of an ordered collection of elements of the *same* type
- To define a vector use the function c() (concatenate)

```
 9  (a <- c(3, 1, 8.5, -1))
10  str(a)
11
12  (a <- c("Hello", "world"))
13  str(a)
14
15  (a <- c(TRUE, FALSE, FALSE, T, F))
16  str(a)
```

```
> str(a)
 num [1:4] 3 1 8.5 -1
>
> (a <- c("Hello", "world"))
[1] "Hello" "world"
> str(a)
 chr [1:2] "Hello" "world"
>
> (a <- c(TRUE, FALSE, FALSE, T, F))
[1]  TRUE FALSE FALSE  TRUE FALSE
> str(a)
 logi [1:5] TRUE FALSE FALSE TRUE FALSE
```

Vectors (2)

- You can include inputs to a vector of different data type however the resulting vector will make them all the same type

```
18  str( c(1, "Hello", 4, TRUE) )
19
```

```
> str( c(1, "Hello", 4, TRUE) )
 chr [1:4] "1" "Hello" "4" "TRUE"
```

Vectors (3)

- You can transform the vector to any data type as well

```
22  a <- as.numeric(a)
23  str(a)
24  a <- as.character(a)
25  str(a)
```

```
> a <- as.numeric(a)
> str(a)
 num [1:4] 3 1 8 -1
> a <- as.character(a)
> str(a)
 chr [1:4] "3" "1" "8" "-1"
```

Vectors (4)

- Creating sequences of numbers

```
 9   x <- 1:7
10   x
```

```
> x
[1] 1 2 3 4 5 6 7
```

```
13   x <- seq(1, 3, by=0.2)
14   x
```

```
> x
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
```

Vectors

- You can access specific elements of a vector

```
16  x
17  x[8]
18  x[c(3,5)]
```

```
> x
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
> x[8]
[1] 2.4
> x[c(3,5)]
[1] 1.4 1.8
```

Modifying Vectors

- You can change, add, or delete parts to a vector

```
13  x <- seq(1, 3, by=0.2)
14  x
15  x[8] <- 34
16  x
17  x[12] <- 3.2
18  x
19  x <- x[c(-8, -12)]
20  x
```

```
> x <- seq(1, 3, by=0.2)
> x
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
> x[8] <- 34
> x
 [1]  1.0  1.2  1.4  1.6  1.8  2.0  2.2 34.0  2.6  2.8  3.0
> x[12] <- 3.2
> x
 [1]  1.0  1.2  1.4  1.6  1.8  2.0  2.2 34.0  2.6  2.8  3.0  3.2
> x <- x[c(-8, -12)]
> x
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.6 2.8 3.0
```

Vector Arithmetic

```
27  a <- c(1, 2, 3, 4, 5)
28  a + 1
29  a^2
30  sqrt(a)
31  sum(a)
```

```
> a <- c(1, 2, 3, 4, 5)
> a + 1
[1] 2 3 4 5 6
> a^2
[1]  1  4  9 16 25
> sqrt(a)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> sum(a)
[1] 15
```

Example of Vector Use

```
25  am %>%
26    arrange(-track_popularity) %>%
27    select(track_name, track_popularity) %>%
28    head(5) %>%
29    kable()
```

track_name	track_popularity
Do I Wanna Know?	82
R U Mine?	77
Why'd You Only Call Me When You're High?	76
Fluorescent Adolescent	75
Arabella	73

Arrays

- Arrays are the R data objects which can store data in two or more dimensions

```
35  my_array <- array(1:24, dim=c(4,6))
36  my_array
37
38  my_array <- array(c(1, 4, 8, 10), dim=c(2,2))
39  my_array
```

```
> my_array <- array(1:24, dim=c(4,6))
> my_array
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
>
> my_array <- array(c(1, 4, 8, 10), dim=c(2,2))
> my_array
     [,1] [,2]
[1,]    1    8
[2,]    4   10
```

Arrays (2)

- You can combine individual vectors to create an array of the elements

```
41  vector1 <- c(5,9,3)
42  vector2 <- c(10,11,12,13,14,15)
43  vector1
44  vector2
45  result <- array(c(vector1,vector2),dim = c(3,3))
46  print(result)
```

```
> vector1 <- c(5,9,3)
> vector2 <- c(10,11,12,13,14,15)
> vector1
[1] 5 9 3
> vector2
[1] 10 11 12 13 14 15
> result <- array(c(vector1,vector2),dim = c(3,3))
> print(result)
     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15
```

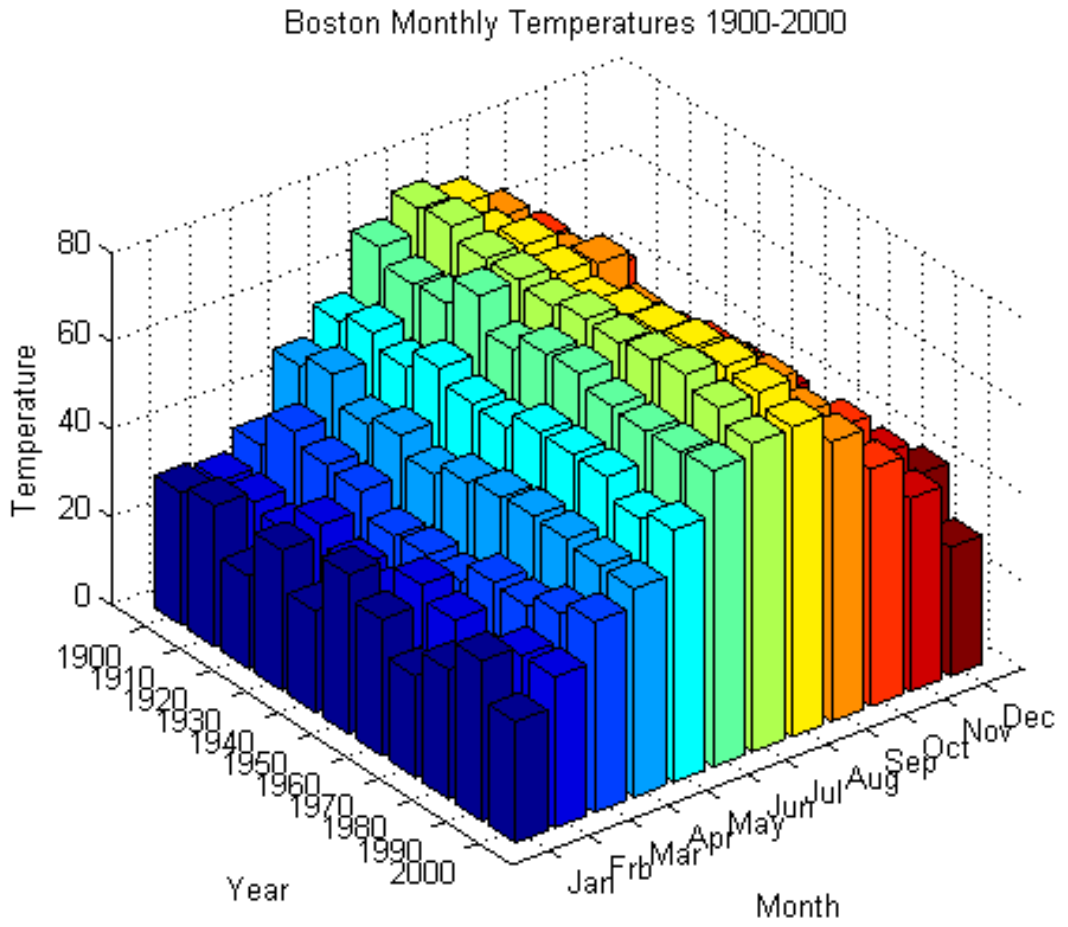
Arrays (3)

- You can always change the dimensions of an existing array

```
48  my_array <- array(1:24, dim=c(4,6))
49  my_array
50  dim(my_array) <- c(6,4)
51  my_array
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
> dim(my_array) <- c(6,4)
> my_array
     [,1] [,2] [,3] [,4]
[1,]    1    7   13   19
[2,]    2    8   14   20
[3,]    3    9   15   21
[4,]    4   10   16   22
[5,]    5   11   17   23
[6,]    6   12   18   24
```

Example of Array Use



Matrices

- An array with two dimensions

```
54  matrix(1:8, nrow = 2)
55  matrix(1:8, ncol = 4)
```

```
> matrix(1:8, nrow = 2)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> matrix(1:8, ncol = 4)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

Matrices (2)

- You can change the ordering of the matrix to order by row

```
1  matrix(1:8, nrow = 2, byrow = TRUE)
```

```
> matrix(1:8, nrow = 2, byrow = TRUE)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Matrices (3)

- You can select specific elements and subsets of a matrix

```
57  (A <- matrix(10:19, nrow = 2))
58  A[7]
59  A[1, 4]
```

```
> (A <- matrix(10:19, nrow = 2))
     [,1] [,2] [,3] [,4] [,5]
[1,]   10   12   14   16   18
[2,]   11   13   15   17   19
> A[7]
[1] 16
> A[1, 4]
[1] 16
```

Matrices (4)

```
70  A <- matrix(1:16, nrow = 4)
71  A[, 2]
72  A[2,]
73  A[c(1, 3), c(2, 4)]
```

```
> A <- matrix(1:16, nrow = 4)
> A[, 2]
[1] 5 6 7 8
> A[2,]
[1]  2  6 10 14
> A[c(1, 3), c(2, 4)]
     [,1] [,2]
[1,]    5   13
[2,]    7   15
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Matrices (5)

- You can combine two matrices together

```
(A <- matrix(1:4, nrow = 2))
(B <- matrix(2:5, nrow = 2))

rbind(A,B)
cbind(A,B)
```

```
> rbind(A,B)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    2    4
[4,]    3    5
> cbind(A,B)
     [,1] [,2] [,3] [,4]
[1,]    1    3    2    4
[2,]    2    4    3    5
```

Example of Matrix Use

```
17  t <- c(108.1, 107.6, 106, 105.1, 104.8)
18  n <- c("Nolan Ryan", "Bob Feller", "Aroldis Chapman", "Aroldis Chapman", "Joel Zumaya")
19  Matrix <- rbind(t,n)
20  Matrix
```

```
> t <- c(108.1, 107.6, 106, 105.1, 104.8)
> n <- c("Nolan Ryan", "Bob Feller", "Aroldis Chapman", "Aroldis Chapman", "Joel Zumaya")
> Matrix <- rbind(t,n)
> Matrix
  [,1]         [,2]         [,3]              [,4]              [,5]
t "108.1"      "107.6"      "106"             "105.1"           "104.8"
n "Nolan Ryan" "Bob Feller" "Aroldis Chapman" "Aroldis Chapman" "Joel Zumaya"
```

Lists

- A list is an object containing elements of any type, including other objects

```
L <- list(c(1, 3, 2),
          c("Two", "Words"),
          list(c(TRUE, FALSE, FALSE),
               "something"))
L
```

Lists (2)

- Accessing elements of a list

```
L <- list(number = c(1, 3, 2),
          char = c("Two", "Words"),
          other_list=list(logical = c(TRUE,FALSE,FALSE),
                          morechar = "something"))
```

```
5  L[1]
6  str(L[1])
7  L$number
8  str(L$number)
9  L["number"]
```

```
> L[1]
$number
[1] 1 3 2

> L[1]
$number
[1] 1 3 2

> str(L[1])
List of 1
 $ number: num [1:3] 1 3 2
> L$number
[1] 1 3 2
> str(L$number)
 num [1:3] 1 3 2
> L["number"]
$number
[1] 1 3 2
```

Lists (3)

```
L <- list(number = c(1, 3, 2),
          char = c("Two", "Words"),
          other_list=list(logical = c(TRUE,FALSE,FALSE),
                          morechar = "something"))
```

```
94  L[[1]][2]
95  L$number[2]
96  L[[3]][[1]][2]
97  L$other_list$logical[2]
98  L[["other_list"]][["logical"]][2]
```

```
> L[[1]][2]
[1] 3
> L$number[2]
[1] 3
> L[[3]][[1]][2]
[1] FALSE
> L$other_list$logical[2]
[1] FALSE
> L[["other_list"]][["logical"]][2]
[1] FALSE
```

Example of List Use

```
27  L <- list(c("eggs", "apples", "milk"), c(2, 3, 2), c("cartons", "lbs", "cartons"))
28  L
```

```
> L <- list(c("eggs", "apples", "milk"), c(2, 3, 2), c("cartons", "lbs", "cartons"))
> L
[[1]]
[1] "eggs"   "apples" "milk"

[[2]]
[1] 2 3 2

[[3]]
[1] "cartons" "lbs"     "cartons"
```

Dataframes

- Similar to Matrices except they can hold all forms of data types

```
df <- data.frame(num=c(3,4,2,-1),
                 char=c("a","b","b","a"),
                 lgc=c(T,T,F,T), stringsAsFactors = FALSE)

df
df[1]
df[[2]]
df[, 1]
df$lgc
```

```
> df
  num char   lgc
1   3    a  TRUE
2   4    b  TRUE
3   2    b FALSE
4  -1    a  TRUE
```

```
> df[1]
  num
1   3
2   4
3   2
4  -1
> df[[2]]
[1] "a" "b" "b" "a"
> df[, 1]
[1]  3  4  2 -1
> df$lgc
[1]  TRUE  TRUE FALSE  TRUE
```

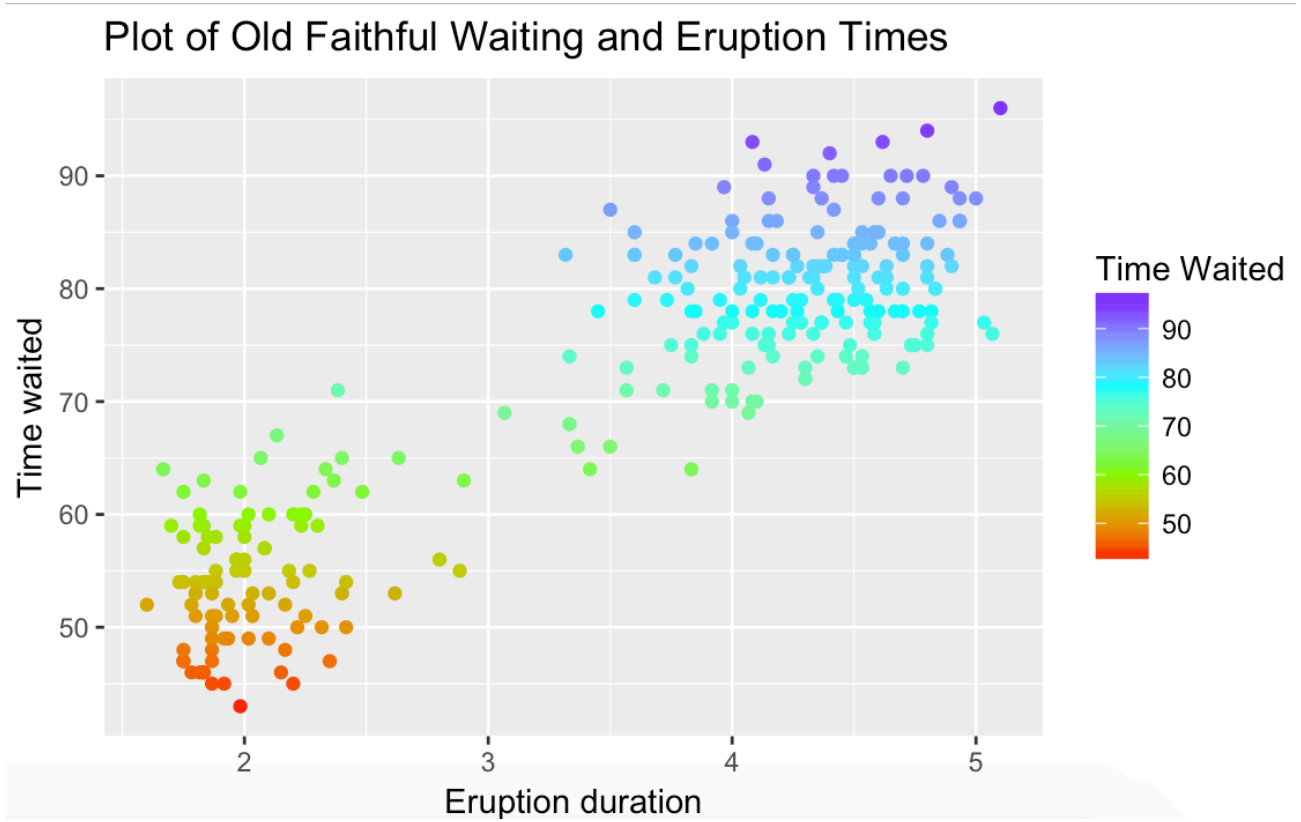
Dataframes (2)

```
53  Pitchers_Speeds <- data.frame(
54    Pitchers_Name = c("Nolan Ryan","Bob Feller","Aroldis Chapman","Aroldis Chapman","Joel Zumaya"),
55    MPH = c(108.1, 107.6, 106, 105.1, 104.8),
56    stringsAsFactors = FALSE
57  )
58  Pitchers_Speeds
```

```
> Pitchers_Speeds
    Pitchers_Name   MPH
1      Nolan Ryan 108.1
2      Bob Feller 107.6
3 Aroldis Chapman 106.0
4 Aroldis Chapman 105.1
5     Joel Zumaya 104.8
```

Example of Dataframe Use

```
21  ggplot(faithful, aes(x = faithful$eruptions, y = faithful$waiting, colour = faithful$waiting)) +
22    geom_point(aes()) +
23    scale_colour_gradientn(name = "Time Waited", colours=rainbow(4)) +
24    labs(x = "Eruption duration", y = "Time waited") +
25    ggtitle("Plot of Old Faithful Waiting and Eruption Times")
```



Congratulations! You have passed the introduction to R