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

Dayton’s Weather in 2014

Temperature

2014 was an average year for daily temperatures. The daily data extends back to January 1, 1995. The data for 2014 is only available from December 16. The average temperature for the year was 51°F, making 2014 the 9th coldest year since 1995.

Incidence of Measles in the US

Cases per 1,000,000 people

Legend:
- >1,000
- 800-1,000
- 80-399
- 1-39
- <1
### Different uses of function `diag()`

1. `diag(3)`
   - 3x3 Identity matrix
2. `diag(c(3, -1, 5))`
   - 3x3 Square matrix with this diagonal

#### Diagonal of a matrix

```r
diag(A)  # Diagonal of a matrix
```

#### Calculate the trace of A

```r
sum(diag(A))  # Calculate the trace of A
```

#### Element by Element Product of Matrices

```r
(A <- matrix(1:4, nrow = 2))
(B <- matrix(2:5, nrow = 2))
A * B  # Element by element product
```

#### Matrices need to have same number of rows and columns

```r
D <- matrix(1:6, nrow = 2)
A * D  # Matrices need to have same number of rows
```

---

### Console

```
> a <- 1
> a * 100
```

```
> a <- 1
> a / sum(a)  # Sum of ALL elements of a
```

```
> seq(from = 1, to = 9.75, by = 0.5)
```

```
> print("hello world")
```

```
> print("Hi my name is Ajith")
```

```
> "Hi my name is Ajith"
```

How to seek help in R

```r
help("print")
> print("hello world")
[1] "hello world"
> print("Hi my name is Ajith")
[1] "Hi my name is Ajith"
> 1 + 1
[1] 2
> 4-3
[1] 1
> 3
> 9/3
> 1
> help("print")
```
Printing Text

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

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

```python
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 + 1</td>
</tr>
<tr>
<td>2</td>
<td>4 - 3</td>
</tr>
<tr>
<td>3</td>
<td>3 * 3</td>
</tr>
<tr>
<td>4</td>
<td>9 / 3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
</tr>
<tr>
<td>^ or **</td>
<td>exponentiation</td>
</tr>
<tr>
<td>x %/ y</td>
<td>modulus (x mod y)</td>
</tr>
<tr>
<td>x %/% y</td>
<td>integer division</td>
</tr>
</tbody>
</table>

- 5 % 2 is \(1\)
- 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

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

```R
> a <- 1
> b <- 2
> r <- "Hello"
> t <- "World"
```
Combining Assigned Variables

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

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

```r
> 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)
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

```r
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

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

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

```r
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

```
> 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
```
Vector Arithmetic

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td><code>a &lt;- c(1, 2, 3, 4, 5)</code></td>
</tr>
<tr>
<td>28</td>
<td><code>a + 1</code></td>
</tr>
<tr>
<td>29</td>
<td><code>a^2</code></td>
</tr>
<tr>
<td>30</td>
<td><code>sqrt(a)</code></td>
</tr>
<tr>
<td>31</td>
<td><code>sum(a)</code></td>
</tr>
</tbody>
</table>

```r
> 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

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

<table>
<thead>
<tr>
<th>track_name</th>
<th>track_popularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do I Wanna Know?</td>
<td>821</td>
</tr>
<tr>
<td>IR U Mine?</td>
<td>771</td>
</tr>
<tr>
<td>Why'd You Only Call Me When You're High?</td>
<td>761</td>
</tr>
<tr>
<td>Fluorescent Adolescent</td>
<td>751</td>
</tr>
<tr>
<td>Arabella</td>
<td>731</td>
</tr>
</tbody>
</table>
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,]  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

```r
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
result <- array(c(vector1,vector2),dim = c(3,3))
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

```r
my_array <- array(1:24, dim=c(4,6))
my_array
dim(my_array) <- c(6,4)
my_array
```

```
[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,]  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

Boston Monthly Temperatures 1900-2000

Temperature

Month Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

0 20 40 60 80
Matrices

• An array with two dimensions
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,]    1    2    3    4
[2,]    5    6    7    8
```
Matrices (3)

• You can select specific elements and subsets of a matrix

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

> A[7]
[1] 16

> A[1, 4]
[1] 16
```
Matrices (4)

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

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

> 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
```
Matrices (5)

- You can combine two matrices together

```r
(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,]   1   3   2   4
[2,]   2   4   3   5
```
Example of Matrix Use

```r
17  t <- c(108.1, 107.6, 106, 105.1, 104.8)
19  Matrix <- rbind(t,n)
20  Matrix

> t <- c(108.1, 107.6, 106, 105.1, 104.8)
> Matrix <- rbind(t,n)
> Matrix
 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

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

• Accessing elements of a list

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

> 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)

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

> 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

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

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

> 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

> df[[2]]
[1] "a" "b" "b" "a"

> df[, 1]
[1] 3 4 2 -1

> df$lgc
[1] TRUE TRUE FALSE TRUE
```
```r
Pitchers_Speeds <- data.frame(
  MPH = c(108.1, 107.6, 106, 105.1, 104.8),
  stringsAsFactors = FALSE
)
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

```r
ggplot(faithful, aes(x = faithful$eruptions, y = faithful$waiting, colour = faithful$waiting)) +
  geom_point(aes()) +
  scale_colour_gradientn(name = "Time Waited", colours=rainbow(4)) +
  labs(x = "Eruption duration", y = "Time waited") +
  ggtitle("Plot of Old Faithful Waiting and Eruption Times")
```
Congratulations!
You have passed the introduction to R