What we have covered so far

- Vectors
- Functions
- Creating Scripts
- Logical Expressions

What we will cover today

- Accessing data files
- Data Frames

Useful Functions

- seq(start, end, skip): generate regular sequences
- sample(n) : generate a permutation of integers [1,..,n]
- sample(n,m): generate m integers in the range [1,...,n]
- 1:10 is a shortcut for seq(1,10)
- rnorm(n) : generate a vector of “n” numbers that satisfy a normal distribution of mean 0 and standard deviation 1
- mean(x) : compute the mean value of x
- sd(x) : compute the standard deviation of x
- var(x) : compute the variance of x

Useful Operators

- Arithmetic: +, -, /, :, and a new one: %%
  - %% : compute the remainder (7 %% 3 returns 1)
  - Seq(1:13) %% 3 returns

    [1] 1 2 0 1 2 0 1 2 0 1 2 0
- Logical: &, |, !

  and (&), or (|), not (!)
- All the above operators act on vectors

Data files

- R provides a set of data files to experiment with
- Use the function data() to see a list of all the files
- Select a particular file: “USArrests”
  - Read the data: data(USArrests)
  - Access the data: USArrests
- Type of data set:
  - class(USArrests) returns data.frame
  - The data is stored in a data frame
  - Alternatives: table, matrix

What is in USArrests

```
> USArrests
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
Connecticut   3.3     110       77 11.1
Delaware      5.9     238       72 15.8
Florida      15.4     335       80 31.9
Georgia      17.4     211       60 25.8
```

```
> class(USArrests)
[1] "data.frame"
```

What are data frames?

- Data frames are simply a collection of columns
- Each column must be the same type
- Each column can be thought of as a vector
  - In fact, vectors are used to construct data frames
- Each column has a header
- Create data frames using the function data.frame()

Create a data.frame

- Df = data.frame(c(1,2,3), c(100,90,80))

  creates a data frame with two columns
- Column 1: c(1,2,3) (vector of integers)
- Column 2: c(100,90,80) (vector of integers)

Data Frames

- heights = c(5.7, 6.2, 5.1, 5.5)
  names = c(“John”, “James”, “Sophie”, “Christine”)

  table = data.frame(names, heights)

  table = data.frame(heights, names)

  table = data.frame(nms=names, h=heights)
- (try the above out and discuss)
- One of the columns has numeric elements
- The other column string elements

More complex data.frame

- heights = c(5.7, 6.2, 5.1, 5.5)
  names = c(“John”, “James”, “Sophie”, “Christine”)

  table = data.frame(names, heights, heights > 5)

- The third column is a vector of logical elements

Data frame Creation

```
> Df = data.frame(c(1,2,3),
c(100,90,80))
> Df
  c.1..2..3. c.100..90..80.
1          1            100
2          2             90
3          3             80
```

- Note the column headers:

  c.1..2..3. : construct from the column data

  We would want better labels

Data frame creation

```
> Df = data.frame(index=c(1,2,3), grades=c(100,90,80))
> Df
  index grades
1     1    100
2     2     90
3     3     80
```

- Note that each column now has a header, as specified in the call to the data.frame function

Data frame creation

- Of course, one can use variable

```
> index = 1:3
> grades = seq(100,80,-10)
> data.frame(indices=index,
grades=grades)
  indices grades
1       1    100
2       2     90
3       3     80
```

Comments

- Each column has elements of the same type
  - Numbers, logicals (T/F/TRUE/FALSE), strings
- All columns MUST have the same length

How to access columns

- How does one extract various parts of a data frame?
  - One or more columns
  - One or more rows
  - A subset of the data frame
    - Using indices or using logical expressions
- We will examine these various approaches using one of the datasets provided in R

How to access columns

- How does one extract various parts of a data frame?
  - One or more columns
  - One or more rows
  - A subset of the data frame
    - Using indices or using logical expressions
- We will examine these various approaches using one of the datasets provided in R

Access columns

```
> head(USArrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

All columns
First few rows

```
> head(USArrests[1])
           Murder
Alabama      13.2
Alaska       10.0
Arizona       8.1
Arkansas      8.8
California    9.0
Colorado      7.9
```

Extract column 1
First few rows

Extract columns 2 and 4

```
> head(USArrests[2:4])
           Assault UrbanPop Rape
Alabama        236       58 21.2
Alaska         263       48 44.5
Arizona        294       80 31.0
Arkansas       190       50 19.5
California     276       91 40.6
Colorado       204       78 38.7
```

Extract rows **[indices,]**

```
> USArrests[10:20,]
         Murder Assault UrbanPop Rape
Georgia    17.4     211       60 25.8
Hawaii      5.3      46       83 20.2
Idaho       2.6     120       54 14.2
Illinois   10.4     249       83 24.0
Indiana     7.2     113       65 21.0
Iowa        2.2      56       57 11.3
Kansas      6.0     115       66 18.0
Kentucky    9.7     109       52 16.3
Louisiana  15.4     249       66 22.2
Maine       2.1      83       51  7.8
Maryland   11.3     300       67 27.8
```

Extract rows with logical expressions

```
> USArrests[USArrests[1] > 10, ]
             Murder Assault UrbanPop Rape
Alabama        13.2     236       58 21.2
Florida        15.4     335       80 31.9
Georgia        17.4     211       60 25.8
Illinois       10.4     249       83 24.0
Louisiana      15.4     249       66 22.2
Maryland       11.3     300       67 27.8
Michigan       12.1     255       74 35.1
Mississippi    16.1     259       44 17.1
Nevada         12.2     252       81 46.0
New Mexico     11.4     285       70 32.1
New York       11.1     254       86 26.1
North Carolina 13.0     337       45 16.1
South Carolina 14.4     279       48 22.5
Tennessee      13.2     188       59 26.9
Texas          12.7     201       80 25.5
```

Extract rows and columns

```
> USArrests[USArrests[1] > 10, 1:2]
               Murder Assault
Alabama          13.2     236
Florida          15.4     335
Georgia          17.4     211
Illinois         10.4     249
Louisiana        15.4     249
Maryland         11.3     300
Michigan         12.1     255
Mississippi      16.1     259
Nevada           12.2     252
New Mexico       11.4     285
New York         11.1     254
North Carolina   13.0     337
South Carolina   14.4     279
Tennessee        13.2     188
Texas            12.7     201
```

Extract with column names

```
> USArrests[USArrests$Murder > 12, 1:2]
               Murder Assault
Alabama          13.2     236
Florida          15.4     335
Georgia          17.4     211
Louisiana        15.4     249
Michigan         12.1     255
Mississippi      16.1     259
Nevada           12.2     252
North Carolina   13.0     337
South Carolina   14.4     279
Tennessee        13.2     188
Texas            12.7     201
```

But, ….

```
> USArrests[Murder > 12, 1:2]
Error in `[.data.frame`(USArrests, Murder > 12, 1:2) :
  object 'Murder' not found
```

The above is an error. The variable name Murder is not in the workspace. Murder is actually a column header of a dataframe.

One must write USArrests$Murder, or USArrests[“Murder”] to access the proper column.

Structure of the data.frame str() function

```
> str(USArrests)
'data.frame':  50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

. str() summarizes the structure of a variable.

In this case, the data frame is composed of 4 columns that are each numerical, two with integers, two with numbers with decimal points.

str() and class() functions

- str() provides information on the structure/content of a variable
- class() provides information on the “type” of a variable

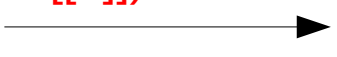
Vector: str and class

```
> v = c(10,15,20,30)
> str(v)
 num [1:4] 10 15 20 30
> class(v)
[1] "numeric"
```

[ ] versus [[ ]]

```
> class(USArrests[1])
[1] "data.frame"
> class(USArrests[[1]])
[1] "numeric"
> str(USArrests)
> class(USArrests[2])
[1] "data.frame"
> class(USArrests[[2]])
[1] "integer"
```

Data frame



Vector

```
str(USArrests)
'data.frame':  50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

Size of data.frame

```
> length(USArrests)
[1] 4

> nrow(USArrests)
[1] 50
```

New functions introduced

- . data.frame()
- . str()
- . class()
- . length()
- . nrow()
- . use of [ ] versus [[ ]]

Dataframes = list of columns
Each column has a header
Use the function data.frame
Arguments: a collection of vectors
Each argument: a variable that represents a vector, or the vector itself.
Example: data.frame(c(1,2,3), dosage)
Each column has the same number of elements.
Accessing columns: d = data.frame(c(1,2,3), dosage)
  column 1: d[1}
  column 2: d[2]
Accessing rows:
  row 1: d[1,]
  row 2: d[2,]
Need to make students understand the concept of rows and columns
Extract a subset of columns: d[c(1,2)] or d[c(T,F)] (all rows extracted)
Extract a subset of rows: d[c(3:5,7),] (all columns extracted)
Extract a subset of rows and columns: d[c(3:5,7), c(1,2)]
Extract a subset of rows: d[c(T,F,F,...),2]
   The row slot is a vector of logicals normally determined by some logical expression
Column names: d = data.frame(idx=c(1,2,3), dose=dosage)
  before equal "=" : the column name
  after equal "=" : the vector (collection of elements of the same type)
  d[1] and d$idx and d["idx"] are three ways of accomplishing the same result:
       return column 1
  d[d$dose < 5, 2] returns column 2, but only the rows where the conditional is true
Note: one cannot write: d[dose < 5, 2] because R has no idea what "dose" is.
   dose is not a variable name: it is a string, that is the column header

What is possible:
with(d, d[close < 5, 2])
  The "with" function adds new variables to the workspace: these new variables are labels for the column headers. The danger of course is that a column header has the name name as a variable already in existence (NOT IMPORTANT FOR NOW)

Read one of the included datasets, create a data.frame (perhaps using as.data.frame),
and