Character vectors Character/string – each element in the vector is a string of one or more characters. Built in character vectors are letters and LETTERS which provide the 26 lower (and upper) case letters, respecitively. > y = c("a", "bc", "def") > length(y) [1] 2	Regardless of the levels/labels of the factor, the numeric storage is an integer with 1 corresponding to the first level (in alph-order). > kids + 1 [1] NA NA NA NA NA > as.numeric(kids) [1] 2 1 2 1 1 1
<pre>> nchar(y) [1] 1 2 3 > y == "a" [1] TRUE FALSE FALSE > y == "b" [1] FALSE FALSE FALSE</pre>	<pre>> 1 + as.numeric(kids) [1] 3 2 3 2 2 2 > kids2 = factor(c("boy","girl","boy","girl","boy","boy")) > kids2 [1] boy girl boy girl boy boy Levels: boy girl > as.numeric(kids2) [1] 1 2 1 2 1 1</pre>
– Typeset by FoilT _E X – 2	– Typeset by FoilT _E X – 4
<pre>R Data Types A supports a few basic data types: integer, numeric, logical, character/string, factor, and complex Logical - binary, two possible values represented by TRUE and FALSE > x = c(3, 7, 1, 2) > x > 2 [1] TRUE TRUE FALSE FALSE > x == 2 [1] FALSE FALSE FALSE TRUE > !(x < 3) [1] TRUE TRUE FALSE FALSE > which(x > 2) [1] 1 2</pre>	<pre>Factor A factor- type vector contains a set of numeric codes with character-valued levels. Example - a family of two girls (1) and four boys (0), > kids = factor(c(1,0,1,0,0,0), levels = c(0, 1),</pre>

Functions to Provide Information about Vectors			
	Logical Operators		
 length(x) - number of elements in a vector or list 	Logical operators are extremely useful in subsetting vectors and in controlling program flow.		
 Aggregator functions - sum, mean, range, min, max, summary, table, cut, 	We will cover these ideas soon.		
• class(x) – returns the type of an object.	• The usual arithemic operators return logicals $> < > = < = and ! $		
 is.logical(x) – tells us whether the object is a logical type. There is also is.numeric, is character is integer 	 Work element-wise on the two inputs. 		
 is.null – determines whether an object is empty, i.e. has no content. 'NULL' is used mainly to represent the lists with zero length, and is often returned by expressions and functions whose value is undefined. 	 Output is a vector of logical elements (TRUE and FALSE) where the elements correspond to the test on the relevant elements on the inputs Conditions can be combined with the use of % for AND and for OD 		
 is.na – NA represents a value that is not available. 	 Conditions can be combined with the use of & for AND and 100 CR. any() returns TRUE if any of the values are TRUE, and all() returns TRUE if all of the 		
> x	values are TRUE.		
[1] 3 1 NA			
> 1S.na(X) [1] FALSE FALSE TRUE			
 as.numeric(x) – we use the as-type functions to coerce objects from one type (e.g. logical) to another, in this case numeric. There are several of these functions, including as.integer, as.character, as.logical, as.POSIXct. 			
- Typeset by FolTEX - 6	- Typeset by FolfIEX - 8		
	1		
Coercion	Missing Values		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this 	Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. 	Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) c(1.2, x) [1] 1.2 1.0 0.0 1.0 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. 		
<pre>Coercion • All elements in a vectors must be of the same type. • R coerces the elements to a common type, in this c(1.2, 3, TRUE) - In this case all elements are coerced to numeric, 1.2, 3, and 1. > x = c(TRUE, FALSE, TRUE) > c(1.2, x) [1] 1.2 1.0 0.0 1.0 > y = c("2", "3", ".2") > c(1.2, y, x)</pre>	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. > mean(x) [1] NA 		
Coercion • All elements in a vectors must be of the same type. • R coerces the elements to a common type, in this c(1.2, 3, TRUE) - In this case all elements are coerced to numeric, 1.2, 3, and 1. • $x = c(TRUE, FALSE, TRUE)$ • $c(1.2, x)$ [1] 1.2 1.0 0.0 1.0 • $y = c("2", "3", ".2")$ • $c(1.2, y, x)$ [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE"	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. > mean(x) [1] NA > mean(x,na.rm = TRUE) 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) c(1.2, x) [1] 1.2 1.0 0.0 1.0 y = c("2", "3", ".2") c(1.2, y, x) [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE" Sometimes this coercion occurs inorder to perform an arithmetic operation: 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. > mean(x) [1] NA > mean(x,na.rm = TRUE) [1] 2 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) c(1.2, x) [1] 1.2 1.0 0.0 1.0 y = c("2", "3", ".2") c(1.2, y, x) [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE" Sometimes this coercion occurs inorder to perform an arithmetic operation: 1 + x 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. > mean(x) [1] NA > mean(x,na.rm = TRUE) [1] 2 Note that NA is not a character value. In facti, it has meaning for character vectors too. 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) – In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) c(1.2, x) [1] 1.2 1.0 0.0 1.0 y = c("2", "3", ".2") c(1.2, y, x) [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE" Sometimes this coercion occurs inorder to perform an arithmetic operation: 1 + x [1] 2 1 2 Other times we need to perform the coercion 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. mean(x) [1] NA mean(x,na.rm = TRUE) [1] 2 Note that NA is not a character value. In facti, it has meaning for character vectors too. y = c("A", "d", NA, "ab", "NA") Notice that the two uses, NA and ""NA" mean very different things. The first is an NA value 		
 All elements in a vectors must be of the same type. R coerces the elements to a common type, in this c(1.2, 3, TRUE) - In this case all elements are coerced to numeric, 1.2, 3, and 1. x = c(TRUE, FALSE, TRUE) c(1.2, x) 1] 1.2 1.0 0.0 1.0 y = c("2", "3", ".2") c(1.2, y, x) [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE" Sometimes this coercion occurs inorder to perform an arithmetic operation: 1 + x [1] 2 1 2 Other times we need to perform the coercion c(1.2, y) [1] "1.2" "2" "3" ".2" 	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. mean(x) NA mean(x,na.rm = TRUE) 2 Note that NA is not a character value. In facti, it has meaning for character vectors too. y = c("A", "d", NA, "ab", "NA") Notice that the two uses, NA and ""NA" mean very different things. The first is an NA value and the second is a character string. 		
Coercion • All elements in a vectors must be of the same type. • R coerces the elements to a common type, in this $c(1.2, 3, TRUE) - In this case all elements are coerced to numeric, 1.2, 3, and 1. > x = c(TRUE, FALSE, TRUE) > c(1.2, x) [1] 1.2 1.0 0.0 1.0 > y = c("2", "3", ".2") > c(1.2, y, x) [1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE" • Sometimes this coercion occurs inorder to perform an arithmetic operation: > 1 + x [1] 2 1 2 • Other times we need to perform the coercion > c(1.2, y) [1] "1.2" "2" "3" ".2" > c(1.2, as.numeric(y)) [1] 1.2 2.0 3.0 0.2$	 Missing Values NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context NA is a recognized element in R x = c(3, 1, NA) Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs. mean(x) mean(x, na.rm = TRUE) Note that NA is not a character value. In facti, it has meaning for character vectors too. y = c("A", "d", NA, "ab", "NA") Notice that the two uses, NA and ""NA" mean very different things. The first is an NA value and the second is a character string. na.omit(), na.exclude(), and na.fail() are for dealling manually with NAs in a dataset. 		

Return values	Vectors, Matrices, Arrays, Lists, and Data Frames		
<pre>> nchar(y) [1] 1 2 2 > nchar("y") [1] 1</pre>	 Vector – a collection of ordered homogeneous elements. We can think of matrices, arrays, lists and data frames as deviations from a vector. The deviaitions are related to the two characteristics order and homogeneity. Matrix - a vector with two-dimensional shape information. 		
> x + 2 a z 5 9 3 4 > x + z a z 4 7 1 3 > $c(x, NA)$ a z 3 7 1 2 NA > $c(x = NA =)$	<pre>> xx = matrix(1:6, nrow=3, ncol =2) > xx [,1] [,2] [1,] 1 4 [2,] 2 5 [3,] 3 6 > class(x) [1] "numeric" > is.vector(xx) [1] FALSE > is.matrix(xx) [1] TRUE > length(xx) [1] TRUE > length(xx) [1] 6 > dim(xx) [1] 3 2</pre>		
 > C (X , " NA ") – Typeset by FoilT_EX – 10 	– Typeset by FoilTi _E X – 12		
<pre>The object x versus the character string "x" > x = c(a = 3, z = 7, 1, 2) > y = c("a", "bc", "NA") > z = c(TRUE, FALSE, FALSE, TRUE) What is the return value for each of the following expressions? nchar(y) nchar("y") x + 2 x + z c(x, NA) c(x, "NA") x[z] x['z"] x[x]</pre>	a z "3" "7" "1" "2" "NA" > x[z] a 3 2 > x["z"] z 7 > is.na(y) [1] FALSE FALSE FALSE > x[x] <na> a z 1 NA 3 7 > is.na(x[x]) <na> a z FALSE TRUE FALSE FALSE</na></na>		
 is.na(y) is.na(x[x]) 			

	Lists	
A vector with possible heterogeneous vectors, character vectors, matrices, a	us elements. The elements of a list can be numeric arrays, and lists.	 names(intel) – returns the element names of the list, which are the names of each of the vectors: "Date" "Transistors" "Microns" etc.
myl ist – list(a – 1:10, b – "def", c(T		 class(intel) – a "data frame"
		 dim(intel) – as a rectangular list, the data frame supports some matrix features; 10.7
\$a [1] 1 2 3 4 5 6 7 8 \$b	B 9 10	 length(intel) – the length is the number of elements in the list, NOT the combined number of elements in the vectors, i.e. it is ?
[1] "def"		 class of intel["Date"] versus intel[["Date"]] – recall the [] returns an object of the same
[[3]]		type, i.e. a list but [[]] returns the element in the list.
[1] TRUE FALSE TRUE		What is the class of the speed element in intel?
 length(myList) – there are 3 elem class(myList) – the class is a "list" names(myList) – are "a", "b" and t myList[1:2] – returns a list with two myList[1] – returns a list with one myList[[1]] – returns a vector w 	ents in the list " the empty character "" o elements element. What is length(myList[1]) ? vith ten elements, the numbers 1, 2,, 10 What is	> intel[["speed"]] [1] MHz MHz MHz MHz MHz MHz MHz GHz GHz Levels: GHz MHz
length(myList[[1]]) ?		
– Typeset by $\mbox{Foi}T_{E}\!X$ –	14	– Typeset by FoilT _E X – 16
		Data Frames
> yy = array(1:12, c(2,3,2)) > yy , , 1		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1</pre>		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1]</pre>		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel Date Transistors Microps Clock speed Data
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6</pre>		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel Date Transistors Microns Clock speed Data MIPS 8080 1974 6000 6.00 2.0 MHz 8 0.64
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6</pre>		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel Date Transistors Microns Clock speed Data MIPS 8080 1974 6000 6.00 2.0 MHz 8 0.64 8088 1979 29000 3.00 5.0 MHz 16 0.33
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2</pre>		Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel Date Transistors Microns Clock speed Data MIPS 8080 1974 6000 6.00 2.0 MHz 8 0.64 8088 1979 29000 3.00 5.0 MHz 16 0.33 80286 1982 134000 1.50 6.0 MHz 16 1.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2</pre>		Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.080881979290003.005.0MHz160.338028619821340001.506.0MHz161.008038619852750001.5016.0MHz325.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3]</pre>		Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.080881979290003.005.0MHz168028619821340001.506.0MHz161.008038619852750001.5016.0MHz325.0080486198912000001.0025.0MHz3220.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11</pre>		Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.080881979290003.005.08028619821340001.506.0MHz168038619852750001.5016.0MHz325.0080486198912000001.0025.0MHz32100.00Pentium19933100000.8060.0MHz32100.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11 [2,] 8 10 12</pre>		Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.0MHz80.6480881979290003.005.0MHz160.338028619821340001.506.0MHz161.008038619852750001.5016.0MHz325.0080486198912000001.0025.0MHz32100.00Pentium19933100000.8060.0MHz32100.00PentiumII19977500000.35233.0MHz32300.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11 [2,] 8 10 12</pre>		Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.08080197460006.002.0808619821340001.506.08038619852750001.5016.08038619852750001.5016.080486198912000001.0025.08048619891200000.8060.0Pentium19933100000.8060.0PentiumII19977500000.35233.0PentiumIII19999500000.25450.0
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11 [2,] 8 10 12 > length(yy)</pre>	[1] 12	Data FramesA list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.> intelDate Transistors Microns Clock speed Data MIPS8080197460006.002.08080197460006.002.0808619821340001.506.08038619852750001.5016.08038619852750001.5016.08048619891200001.0025.08048619891200001.0025.0Pentium19933100000.8060.0PentiumII19977500000.35233.0PentiumIII19999500000.25450.0PentiumIII19999500000.181.5GHZ321700.00
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11 [2,] 8 10 12 > length(yy) > dim(yy)</pre>	$\begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 $	Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel
<pre>> yy = array(1:12, c(2,3,2)) > yy , , 1 [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6 , , 2 [,1] [,2] [,3] [1,] 7 9 11 [2,] 8 10 12 > length(yy) > dim(yy) > is.matrix(yy) > is.array(yy)</pre>	[1] 12 [1] 2 3 2 [1] FALSE [1] TRUE	Data Frames A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length. > intel

13

– Typeset by $\operatorname{FoilT}_{E}X$ –

15

Subsetting a Data Frame

Using the fact that a data frame is a list which also support some matrix features, fill in the table specifying the **class** (data.frame or ineger) and the **length** and **dim** of the subset of the data frame. Note that some responses will be NULL.

Subset	class	length	dim
intel			
intel[1]			
intel[[1]]			
intel[,1]			
intel["Date"]			
intel[, "Date"]			
intel\$Date			