R-and-Data

R and Data

R provides several basic structures for working with data.

In these notes we will review some of the most important ones:

- vectors
- lists
- data frames

We will also learn how to work with subsets of our data by indexing into rows and/or columns.

Vectors

Our simplest kind of data is just a bunch of numbers. We can use a vector to hold numbers:

```
leafsG = c(47,31,26,21,16,13)
summary(leafsG)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.00   17.25   23.50   25.67   29.75   47.00
length(leafsG)
## [1] 6
```

The function `summary` gives us some statistical summaries of the numbers in the vector leafsG.

The function `length` tells us the length of the vector.

There are a bunch of ways to make vectors in R:

```
ind = 1:10 # 1 to 10
print("ind is:")
## [1] "ind is:"
ind
##  [1]  1  2  3  4  5  6  7  8  9 10
x = seq(1,20,2) # 1 to 20 by 2
print("x is :")
## [1] "x is :"
x
##  [1]  1  3  5  7  9 11 13 15 17 19
```

Indexing

Very often we need access to a subset of values in a vector.

We append square brackets to the vector name and put the indices of the elements we want inside the brackets.

```
leafsG[2] #the second number
## [1] 31
leafsG[2:4] # the second through 4th
## [1] 31 26 21
leafsG[c(1,3,5)] # the first, third, and fifth
## [1] 47 26 16
```

We can also assign to elements of a vector using indices:

```
x = 1:10
x[2]=20
x
##  [1]  1 20  3  4  5  6  7  8  9 10
x[c(5,7,9)] = c(50,70,90)
x
##  [1]  1 20  3  4 50  6 70  8 90 10
```

Arithmetic with Vectors

We can use arithmetic expressions with vectors.
The arithmetic is applied to all elements of the vector in a (usually) intuitive way.

```
x = c(2,5,7,9)
x^2
## [1]  4 25 49 81
1 + 2* x
## [1]  5 11 15 19
y = c(4,6,8,9)
x+y
## [1]  6 11 15 18
z = log(y)
z
## [1] 1.386294 1.791759 2.079442 2.197225
```

What happens when we add two vectors?

```
x = c(1,2,3)
y = c(20,30,40)
z = x  + y
z
## [1] 21 32 43
```

The first x is added to the first y, the second x to the second y, and so on.
This is very powerful.

```
1 + 2*x + y
## [1] 23 35 47
```

Here each element of x is doubled then we add the corresponding elements of y and then we add 1 to all of the elements.

Character and Logical Vectors

So far we have used vectors to contain a bunch of numbers.

Besides numbers, we often work with character strings and logical the logical values TRUE and FALSE.

```
x = "Rob"  # x refers to the character string "Rob"
x
## [1] "Rob"
x = TRUE
x
## [1] TRUE
```

Vectors have to be "all one thing", where the "one thing" could be numbers, character strings, or logical values.

```
pnms = c("Matthews","Nylander","Tavares", "Hyman", "Marner" ,"Kapenan")
length(pnms)
## [1] 6
pnms
## [1] "Matthews" "Nylander" "Tavares"  "Hyman"    "Marner"   "Kapenan"
```

We can label all the elements of a vector:

```
names(leafsG) = pnms
leafsG
## Matthews Nylander  Tavares    Hyman   Marner  Kapenan
##       47       31       26       21       16       13
```

A logical values tells us if something is TRUE or FALSE:

```
leafsG[1] == 47 # is the first value equal to 47, note the double =
## Matthews
##     TRUE
leafsG[1] >= 50
## Matthews
##    FALSE
g20 = leafsG >=20
g20
## Matthews Nylander  Tavares    Hyman   Marner  Kapenan
##     TRUE     TRUE     TRUE     TRUE    FALSE    FALSE
```

g20 is a vector of logical values indicating whether or not the value of leafsG is greater than or equal to 20.

You can select observations using a logical vector:

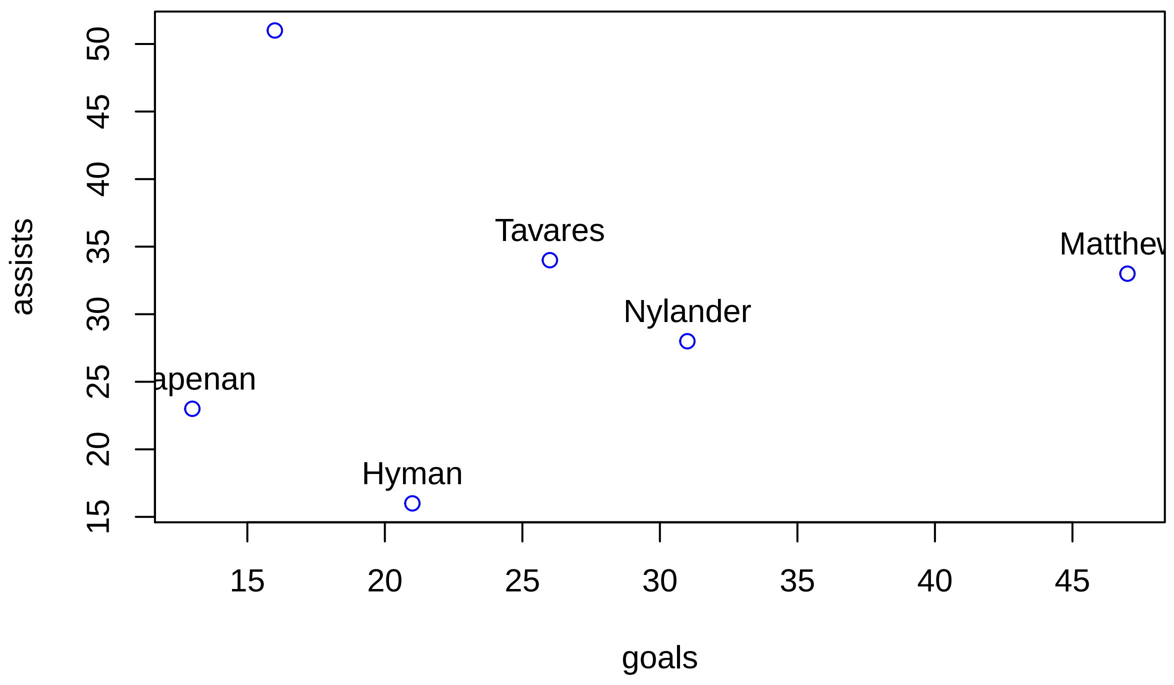
```
leafsG[g20]
## Matthews Nylander  Tavares    Hyman
##       47       31       26       21
# or more succinctly
leafsG[leafsG > 20]
## Matthews Nylander  Tavares    Hyman
##       47       31       26       21
```

More on Plotting

You can add points, lines, and text to and existing plot.
This allows us to build up interesting plots.

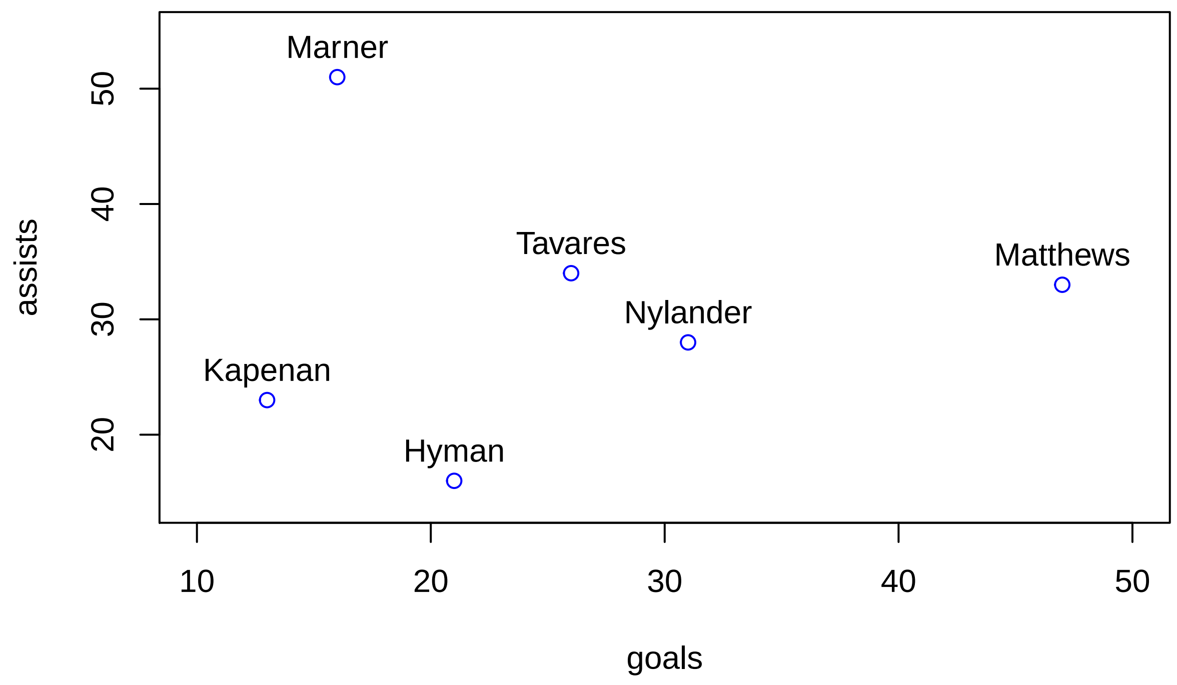
```
leafsG = c(47,31,26,21,16,13) # goals
leafsA = c(33,28,34,16,51,23) # assists
pnms = c("Matthews","Nylander","Tavares", "Hyman", "Marner" ,"Kapenan") #player names
plot(leafsG,leafsA,xlab="goals",ylab = "assists",col="blue")

# now I can add text to the plot
text(leafsG,leafsA,pnms,pos=3)
```



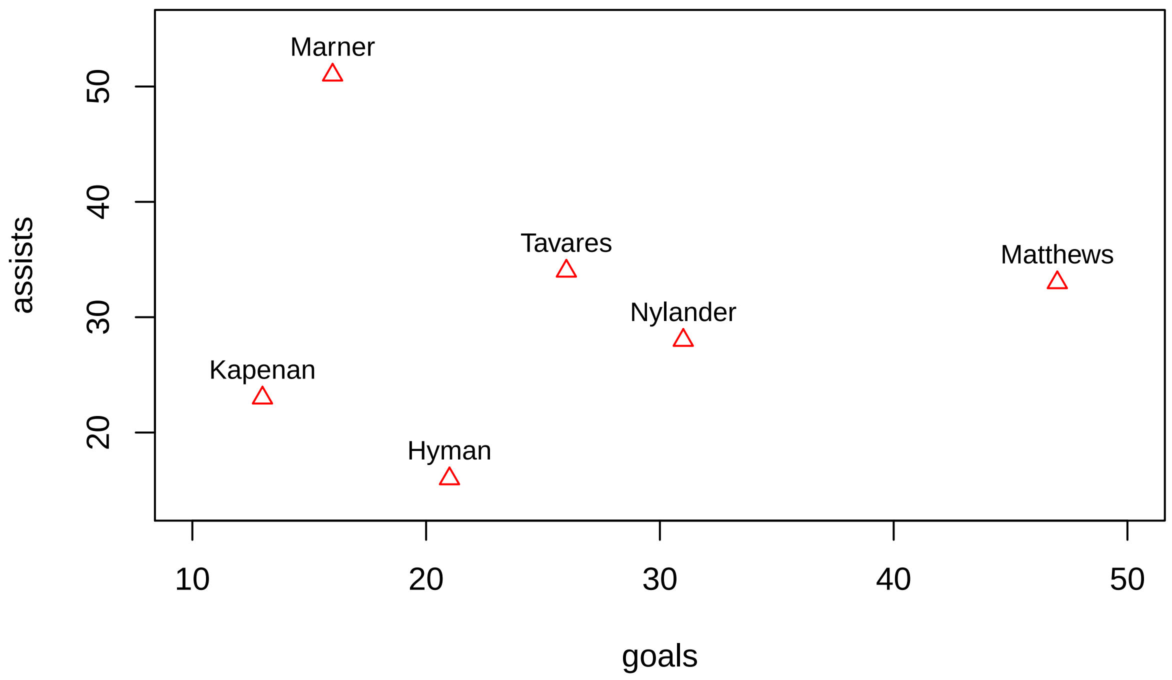
That did not quite work. We need to change the plot limits.

```r
plot(leafsG,leafsA,xlab="goals",ylab = "assists",col="blue",ylim=c(14,55),xlim=c(10,50))
text(leafsG,leafsA,pnms,pos=3)
```



You can also add points and lines.
I make an initial plot with the argument `type="n"`. This will make the plot axes but do no plotting.
Then I add the points and text.

```r
plot(leafsG,leafsA,xlab="goals",ylab = "assists",col="blue",
     ylim=c(14,55),xlim=c(10,50),type="n")
text(leafsG,leafsA,pnms,pos=3,cex=.8)
points(leafsG,leafsA,col="red",pch=2)
```



Notes on the arguments:

- cex control the size of the plot character or text being plotted
- pos controls how the text is placed (left, right, up , down)
- xlim controls the range of the x-axis and ylim controls the range of the y-axis

Lists

Sometimes we need to store a variety of types of information in one place. We can use a list to to this.

```
player = list(first = "Auston", last = "Mathews", pts = c(47,33))
player
## $first
## [1] "Auston"
##
## $last
## [1] "Mathews"
##
## $pts
## [1] 47 33
```

The list player has the first and last name as well a vector containing the goals and assists scored by the the player.

You can access a component of a list using the $ notation or a numerical index:

```
player$pts # the (goal, assists) vector component
## [1] 47 33
player[1:2] # a list with the first two components
## $first
## [1] "Auston"
##
## $last
## [1] "Mathews"
player[[3]] # same as player$pts, note the double [[]]
## [1] 47 33
```

Data Frames

Very often data is represented as a rectangular array where rows correspond to different observations and columns correspond to different variables. In R, we use a data.frame to represent this.

Let's say each observation corresponds to a Leafs player (as in our vector example) and the variables we want to think about are number of goals and number of assists.

First let's make a vector of the assists:

```
leafsA = c(33,28,34,16,51,23)
```

Now we can make a data.frame holding the goals and assists:

```
leafsTop = data.frame(leafsG,leafsA)
leafsTop
##   leafsG leafsA
## 1     47     33
## 2     31     28
## 3     26     34
## 4     21     16
## 5     16     51
## 6     13     23
```

When you make the data frame, you can choose new names for the variables:

```
leafsTop = data.frame(G = leafsG, A = leafsA)
leafsTop
##    G  A
## 1 47 33
## 2 31 28
## 3 26 34
## 4 21 16
## 5 16 51
## 6 13 23
```

You can pull of variables by name or index and you can pull off rows by index:

```
leafsTop$G
## [1] 47 31 26 21 16 13
```

```
leafsTop[1:3,1] # first three rows of first column
## [1] 47 31 26
leafsTop[,1] # the first column, note the [,1] says we want all the rows
## [1] 47 31 26 21 16 13
leafsTop[1:3,2] # first three rows of second column
## [1] 33 28 34
tp3 = leafsTop[1:3,] # first three rows, all the columns
tp3
##    G  A
## 1 47 33
## 2 31 28
## 3 26 34
```

You can get a summary of each variable in a data.frame:

```
summary(leafsTop)
##        G               A        
##  Min.   :13.00   Min.   :16.00  
##  1st Qu.:17.25   1st Qu.:24.25  
##  Median :23.50   Median :30.50  
##  Mean   :25.67   Mean   :30.83  
##  3rd Qu.:29.75   3rd Qu.:33.75  
##  Max.   :47.00   Max.   :51.00
```

You can apply a function to each column of a data.frame:

```
apply(leafsTop,2,sd)
##        G        A 
## 12.32342 11.92337
```

You can assign using indexing:

```
leafsTop[1,1]=50  #Auston should have got to 50!!
leafsTop
##    G  A
## 1 50 33
## 2 31 28
## 3 26 34
## 4 21 16
## 5 16 51
## 6 13 23
```

As with vectors, we can select using logic:

```
leafsTop[leafsTop$G >20,]
##    G  A
## 1 50 33
## 2 31 28
## 3 26 34
## 4 21 16
```