joining-vectors

2023-03-16

library(dyplr)

What are vectors?

- they are unidimensional matrices
- they can only hold one type of data, either numeric (integer or double), character, or logical(complex numbers)

##What are data frames

- It is a two dimensional matrix, rows, and columns
- it can hold any type of data
- it can only hold different of data in a certain way"
- only columns can have differnt data types
- within a column, all rows have the same data type
- A data frame can also be defined as a colection of vectors (they can be of different or the same type) all of the same length!

```
surveys <- read.csv(file = "../data raw/surveys.csv")
species <- read.csv (file = "../data raw/species.csv")
str(surveys)
```

```
## 'data.frame':    35549 obs. of  9 variables:
##  $ record_id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ month          : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ day            : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ year           : int  1977 1977 1977 1977 1977 1977 1977 1977 1977 1977 ...
##  $ plot_id        : int  2 3 2 7 3 1 2 1 1 6 ...
##  $ species_id     : chr  "NL" "NL" "DM" "DM" ...
##  $ sex            : chr  "M" "M" "F" "M" ...
##  $ hindfoot_length: int  32 33 37 36 35 14 NA 37 34 20 ...
##  $ weight         : int  NA NA NA NA NA NA NA NA NA NA ...
```

Creating vectors - review

```
c("luna", "Avi", "Anita", "James", "Charles", "Damian", "Davinder") -> our_names
1:7 # the colon operator creates a vector of numbers
```

```
## [1] 1 2 3 4 5 6 7
```

```
1:7 -> my_numbers
-100:200
```

```
##   [1] -100  -99  -98  -97  -96  -95  -94  -93  -92  -91  -90  -89  -88  -87  -86
##  [16]  -85  -84  -83  -82  -81  -80  -79  -78  -77  -76  -75  -74  -73  -72  -71
##  [31]  -70  -69  -68  -67  -66  -65  -64  -63  -62  -61  -60  -59  -58  -57  -56
##  [46]  -55  -54  -53  -52  -51  -50  -49  -48  -47  -46  -45  -44  -43  -42  -41
##  [61]  -40  -39  -38  -37  -36  -35  -34  -33  -32  -31  -30  -29  -28  -27  -26
##  [76]  -25  -24  -23  -22  -21  -20  -19  -18  -17  -16  -15  -14  -13  -12  -11
##  [91]  -10   -9   -8   -7   -6   -5   -4   -3   -2   -1    0    1    2    3    4
## [106]    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19
## [121]   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
## [136]   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49
## [151]   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
## [166]   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79
## [181]   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94
## [196]   95   96   97   98   99  100  101  102  103  104  105  106  107  108  109
## [211]  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124
## [226]  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139
## [241]  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154
## [256]  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169
## [271]  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184
## [286]  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199
## [301]  200
```

#In how steps does the colon operator increase? It increases in a unit of 1.

What do we do if I want to create a numeric sequence that increases in steps different than 1?

```
seq
```

```
## function (...)
## UseMethod("seq")
## <bytecode: 0x7fc0b6f571c8>
## <environment: namespace:base>
```

```
seq(-100,200, by = 0.1) -> my_numbers
str(my_numbers)
```

```
##  num [1:3001] -100 -99.9 -99.8 -99.7 -99.6 -99.5 -99.4 -99.3 -99.2 -99.1 ...
```

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

Creating data frames from vectors

The most general way to do this is with teh function `data.frame()` :

```
data.frame(names = our_names, numbers = my_numbers)
```

```
## Error in data.frame(names = our_names, numbers = my_numbers): arguments imply differing number of ro
```

Remember: Vectors must have the same length (or be a multiple) to be part of a data frame!

```
data.frame(names = our_names, numbers =1:7)
```

```
##      names numbers
## 1     luna       1
## 2      Avi       2
## 3    Anita       3
## 4    James       4
## 5  Charles       5
## 6   Damian       6
## 7 Davinder       7
```

R will only recycle the values only if they are multipe of the vector:

```
data.frame(names = our_names, numbers = 1)
```

```
##      names numbers
## 1     luna       1
## 2      Avi       1
## 3    Anita       1
## 4    James       1
## 5  Charles       1
## 6   Damian       1
## 7 Davinder       1
```

To recycle the values of a numeric vector of length 2, we have to repeat the vector of names two times, so it is a multiple of 2.

```
data.frame(names = rep(our_names, 2), numbers = c(2, 5.5))
```

```
##       names numbers
## 1      luna     2.0
## 2       Avi     5.5
## 3     Anita     2.0
## 4     James     5.5
## 5   Charles     2.0
## 6    Damian     5.5
## 7  Davinder     2.0
## 8      luna     5.5
## 9       Avi     2.0
## 10    Anita     5.5
## 11    James     2.0
## 12  Charles     5.5
## 13   Damian     2.0
## 14 Davinder     5.5
```

You have data on the length, width, and height of 10 individuals of the yew Taxus baccata stored in the following vectors: Make a data frame that contains these three vectors as columns along with a "genus" column containing the genus name Taxus on all rows and a "species" column containing the species epithet baccata on all rows.

```
length <- c(2.2, 2.1, 2.7, 3.0, 3.1, 2.5, 1.9, 1.1, 3.5, 2.9)
width <- c(1.3, 2.2, 1.5, 4.5, 3.1, NA, 1.8, 0.5, 2.0, 2.7)
height <- c(9.6, 7.6, 2.2, 1.5, 4.0, 3.0, 4.5, 2.3, 7.5, 3.2)
str(species)
```

```
## 'data.frame':    54 obs. of  4 variables:
##  $ species_id: chr  "AB" "AH" "AS" "BA" ...
##  $ genus     : chr  "Amphispiza" "Ammospermophilus" "Ammodramus" "Baiomys" ...
##  $ species   : chr  "bilineata" "harrisi" "savannarum" "taylori" ...
##  $ taxa      : chr  "Bird" "Rodent" "Bird" "Rodent" ...
```

```
data.frame(names = length, width, height, genus = "Taxus", species = "baccata")
```

```
##    names width height genus species
## 1    2.2   1.3    9.6 Taxus baccata
## 2    2.1   2.2    7.6 Taxus baccata
## 3    2.7   1.5    2.2 Taxus baccata
## 4    3.0   4.5    1.5 Taxus baccata
## 5    3.1   3.1    4.0 Taxus baccata
## 6    2.5    NA    3.0 Taxus baccata
## 7    1.9   1.8    4.5 Taxus baccata
## 8    1.1   0.5    2.3 Taxus baccata
## 9    3.5   2.0    7.5 Taxus baccata
## 10   2.9   2.7    3.2 Taxus baccata
```

Extracting/accessing values from vectors and data frames

```
surveys <- read.csv(file = "../data raw/surveys.csv")
species <- read.csv (file = "../data raw/species.csv")

str(surveys)
```

```
## 'data.frame':    35549 obs. of  9 variables:
##  $ record_id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ month          : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ day            : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ year           : int  1977 1977 1977 1977 1977 1977 1977 1977 1977 1977 ...
##  $ plot_id        : int  2 3 2 7 3 1 2 1 1 6 ...
##  $ species_id     : chr  "NL" "NL" "DM" "DM" ...
##  $ sex            : chr  "M" "M" "F" "M" ...
##  $ hindfoot_length: int  32 33 37 36 35 14 NA 37 34 20 ...
##  $ weight         : int  NA NA NA NA NA NA NA NA NA NA ...
```

One common way to extract or access vectors from column in a data frame is the dollar sign symbol $

```
surveys$record_id -> record_id
```

Another way is with the square brackets `[]`

```
surveys[1:10, "hindfoot_length"]
```

```
##  [1] 32 33 37 36 35 14 NA 37 34 20
```

If I want all the values from the rows of column hindfoot length:

```
surveys[, "hindfoot_length"] -> hindfoot_length
```

Another way is to use double square brackets

```
surveys[["record_id"]] %>% head()
```

```
## [1] 1 2 3 4 5 6
```

```
surveys$weight -> surveys_weight
surveys [,"month"] -> surveys_month
surveys$hindfoot_length -> hindfoot_length
str(hindfoot_length)
```

```
##  int [1:35549] 32 33 37 36 35 14 NA 37 34 20 ...
```

```
na.omit(hindfoot_length) %>%
  mean
```

```
## [1] 29.28793
```