Vectors and Factors Lab

DSC 105, Introduction to data science, Lyon College, Fall 2024

Marcus Birkenkrahe (pledged)

October 11, 2024

Vectors and Factors

1. How can you append `foo` to `bar` if `c(1,2,3) -> foo`, `c(4,5,6) -> bar`?

```
c(1,2,3) -> foo
c(4,5,6) -> bar
foo; bar
c(bar,foo) # solution
```

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

2. How can you turn `c("a","b") -> baz` into a `numeric` vector?

```
c("a","b") -> baz
baz
as.numeric(baz) # solution - but characters cannot become numbers
```

```
[1] "a" "b"
[1] NA NA
Warning message:
NAs introduced by coercion
```

3. How can you turn `c(1,2,3) -> foo` into a `character` vector?

```
c(1,2,3) -> foo
foo
as.character(foo) # solution: numbers can become characters
as.numeric(as.character(foo)) # number characters can become numbers
```

```
[1] 1 2 3
[1] "1" "2" "3"
[1] 1 2 3
```

4. What will `c(1,1,1.0)`, `c(1,1,1.1)`, `c(1,1,1e+6)`, `c(1,1,0.000001e6)` print?

```
c(1,1,1.0)  # coerced to double but displayed as integer
c(1,1,1.1)  # coerced to double
c(1,1,1e+6) # coerced to scientific display
c(1,1,0.000001e6) # coerced to regular display
```

```
[1] 1 1 1
[1] 1.0 1.0 1.1
[1] 1e+00 1e+00 1e+06
[1] 1 1 1
```

5. What R object class are: `Inf`, `NA`, `NaN`, `NULL`, and `c(NA, NaN, Inf)`?

```
class(Inf)
class(NA)  # logical class
class(NaN)
class(NULL) # NULL class
class(c(NA, NaN, Inf)) # numeric vector
```

```
[1] "numeric"
[1] "logical"
[1] "numeric"
[1] "NULL"
[1] "numeric"
```

6. Convert the vector `names` with the elements `"Joe", "Jeff", "Jim", "Jane"` to a factor and store it in `names_f`.

```
names <- c("Joe","Jeff","Jim")
names
factor(names) -> names_f
names_f
```

```
[1] "Joe"  "Jeff" "Jim"
[1] Joe  Jeff Jim
Levels: Jeff Jim Joe
```

7. Extract the levels vector of `names_f` and determine its object class using a pipe!

```
levels(names_f) |> class()
```

```
[1] "character"
```

8. Create a named vector `grades` with the elements A, C, B, B for the `grades` of Jeff, Jim, Jane and Joe, respectively.

```
grades <- c('A','C','B','B')  # just a character vector
grades
grades <- c(Jeff='A',Jim='C',Jane='B',Joe='B')  # a named character vector
grades
```

```
[1] "A" "C" "B" "B"
Jeff  Jim Jane  Joe
 "A"  "C"  "B"  "B"
```

9. Convert `grades` to an ordered factor `grades_f` with the `levels` A, B, C so that A $>$ B $>$ C,

```
factor(grades)
factor(grades,ordered=TRUE)
factor(grades,ordered=TRUE,levels=c('C','B','A')) -> grades_f # solution
grades_f
```

```
Jeff  Jim Jane  Joe
   A    C    B    B
Levels: A B C
Jeff  Jim Jane  Joe
   A    C    B    B
Levels: A < B < C
Jeff  Jim Jane  Joe
   A    C    B    B
Levels: C < B < A
```

10. Store `grades` and `grades_f` in a `data.frame` named `df`, and then apply first `str` and then `summary` to `df`.

```
data.frame(grades,grades_f) -> df
str(df)
summary(df)
```

```
'data.frame': 4 obs. of  2 variables:
 $ grades  : chr  "A" "C" "B" "B"
 $ grades_f: Ord.factor w/ 3 levels "C"<"B"<"A": 3 1 2 2
grades          grades_f
 Length:4           C:1
 Class :character   B:2
 Mode  :character   A:1
```

11. Use `grades_f` to show that Jeff is a better student than Jim.

```
grades_f["Jeff"]
grades_f["Jim"]

grades_f["Jeff"] > grades_f["Jim"] # solution
```

```
Jeff
   A
Levels: C < B < A
Jim
  C
Levels: C < B < A
[1] TRUE
```

12. Add two levels, D and F, to `grades_f`, and then test with `str` if it worked.

```
c('F','D',levels(grades_f)) -> levels(grades_f)
levels(grades_f)
str(grades_f)
```

```
[1] "F" "D" "C" "B" "A"
Ord.factor w/ 5 levels "F"<"D"<"C"<"B"<..: 3 1 2 2
 - attr(*, "names")= chr [1:4] "Jeff" "Jim" "Jane" "Joe"
```