My understandment how split works on more than one level:

(relative to week3 of R Programming Language module of Data Science)

Let x, f1, f2 as follows:

```
> x <- 1:10
> f1 <- gl (2, 5)
> f2 <- gl (5, 2)
```

Split of f1 levels it is easy:

```
> f1
 [1]  1  1  1  1  1  2  2  2  2  2
Levels: 1 2
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> str(split (x, f1))
List of 2
 $ 1: int [1:5] 1 2 3 4 5
 $ 2: int [1:5] 6 7 8 9 10
```

As it is split of f2:

```
> f2
 [1]  1  1  2  2  3  3  4  4  5  5
Levels: 1 2 3 4 5
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> str(split (x, f2))
List of 5
 $ 1: int [1:2] 1 2
 $ 2: int [1:2] 3 4
 $ 3: int [1:2] 5 6
 $ 4: int [1:2] 7 8
 $ 5: int [1:2] 9 10
```

Split of 2 levels, f1 and f2, it is an interaction of f1 and f2

```
> f1
 [1]  1  1  1  1  1  2  2  2  2  2
Levels: 1 2
> f2
 [1]  1  1  2  2  3  3  4  4  5  5
Levels: 1 2 3 4 5
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> str(split (x, list(f1,f2)))
List of 10
 $ 1.1: int [1:2] 1 2
 $ 2.1: int(0)
 $ 1.2: int [1:2] 3 4
 $ 2.2: int(0)
 $ 1.3: int 5
 $ 2.3: int 6
 $ 1.4: int(0)
 $ 2.4: int [1:2] 7 8
 $ 1.5: int(0)
 $ 2.5: int [1:2] 9 10
```

In `split(x,list(f1,f2))` there is an implicit call to the function `interaction(f1,f2)`, but there is no respective data for some of results of that interaction.

Because of the implicit call of interaction is that some combinations are nonexistant (they are marked as red).

```
> interaction(f1, f2)
 [1] 1.1 1.1 1.2 1.2 1.3 2.3 2.4 2.4 2.5 2.5
Levels: 1.1 2.1 1.2 2.2 1.3 2.3 1.4 2.4 1.5 2.5
```

The 2 codes are, in fact, the same:

```
str(split (x, list (f1, f2)))
str(split (x, interaction (f1, f2)))
```