# Week 6 Module 3: Tables

Exercises

Let's clear the environment:

```
rm( list = ls() )
```

## Week 6 Module 2 Exercises

### Exercise 2.1: Specifying the levels

Suppose we have a very simple vector of character string labels:

```
simple.character.string.vector <-
  c( "blue", "blue", "green", "red", "green", "blue", "red" )
```

Construct a factor that consists of just the values "green" and "blue", and uses the value `NA` when a "red" value occurs in the original character vector.

**Solution**

### Exercise 2.2: Changing the Levels

Using the vector `simple.character.string.vector`, construct a factor with two levels, which have the labels "Green" and "Blue".

Specify the levels using the `levels` option, and specify the labels for the levels using the `labels` option.

**Solution**

### Exercise 2.3: Re-ordering the levels

Re-order the levels of `simple.character.factor` so that "Green" is the first level and "Blue" is the second level.

**Solution**

### Exercise 2.4: Grouping levels

Suppose we have a sequence of cereal box sales stored in a vector:

```
cereal.sales.vector <-
    c( "sbz", "sbz", "kym", "hkt",
       "sbz", "kym", "sb", "ky" )

cereal.sales.factor <-
    factor( cereal.sales.vector )

levels( cereal.sales.factor )
```

```
## [1] "hkt" "ky"  "kym" "sb"  "sbz"
```

We want to represent these sales using our standard 3-letter abbreviation.

Notice that the last 2 entries only use 2 letters for the abbreviation, and R will treat these as distinct levels:

```
cereal.sales.factor
```

```
## [1] sbz sbz kym hkt sbz kym sb  ky
## Levels: hkt ky kym sb sbz
```

Group the levels "sbz" and "sb" together as "sbz". Likewise, group the levels "kym" and "ky" together as "kym".

**Solution**

### Exercise 2.5: Creating factors from numeric vectors

Suppose you have a set of final course scores:

```
final.course.scores <-
    c( 92.1, 85.4, 78.7, 91.3,
       90.0, 93.0, 61.1, 80.0 )
```

Let's use a simplified grading scheme:

- If a student scores 90 or above, the student receives an "A" letter grade.
- If the student scores 80 or above, but less than 90, then the student receives a "B" letter grade.
- If the student scores 70 or above, but less than 90, then the student receives a "C" letter grade.
- If the student scores strictly less than 70, then the student receives an "E" letter grade.

Convert the vector of final course scores to letter grades using the `cut` function. Be careful about how your handle the endpoints.

**Solution**

## Solutions to the Exercises

### Exercise 2.1: Specifying the levels

Suppose we have a very simple vector of character string labels:

```
simple.character.string.vector <-
  c( "blue", "blue", "green", "red", "green", "blue", "red" )
```

Construct a factor that consists of just the values "green" and "blue" by using the `levels` option.

**Solution**

```
simple.factor <-
    factor(
        simple.character.string.vector,
        levels = c("green", "blue")
    )

simple.factor
```

```
## [1] blue  blue  green <NA>  green blue  <NA>
## Levels: green blue
```

## Exercise 2.2: Changing the Levels

Using the vector `simple.character.string.vector`, construct a factor with two levels, which have the labels "Green" and "Blue".

Specify the levels using the `levels` option, and specify the labels for the levels using the `labels` option.

**Solution**

```
simple.factor <-
    factor(
        simple.character.string.vector,
        levels = c("green", "blue"),
        labels = c( "Green", "Blue" )
    )

simple.factor
```

```
## [1] Blue  Blue  Green <NA>  Green Blue  <NA>
## Levels: Green Blue
```

## Exercise 2.3: Re-ordering the levels

Re-order the levels of `simple.character.factor` so that "Blue" is the first level and "Green" is the second level.

**Solution**

```
reordered.simple.factor <-
    factor(
        simple.factor,
        levels = c( "Blue", "Green" )
    )

reordered.simple.factor
```

```
## [1] Blue  Blue  Green <NA>  Green Blue  <NA>
## Levels: Blue Green
```

## Exercise 2.4: Grouping levels

Suppose we have a sequence of cereal box sales stored in a vector:

```
cereal.sales.vector <-
    c( "sbz", "sbz", "kym", "hkt",
       "sbz", "kym", "sb", "ky" )

cereal.sales.factor <-
    factor( cereal.sales.vector )
```

We want to represent these sales using our standard 3-letter abbreviation.

Notice that the last 2 entries only use 2 letters for the abbreviation, and R will treat these as distinct levels:

```
cereal.sales.factor
```

```
## [1] sbz sbz kym hkt sbz kym sb  ky
## Levels: hkt ky kym sb sbz
```

Group the levels "sbz" and "sb" together as "sbz". Likewise, group the levels "kym" and "ky" together as "kym".

**Solution**

```
levels( cereal.sales.factor ) <-
    c( "hkt", "kym", "kym", "sbz", "sbz" )

cereal.sales.factor
```

```
## [1] sbz sbz kym hkt sbz kym sbz kym
## Levels: hkt kym sbz
```

## Exercise 2.5: Creating factors from numeric vectors

Suppose you have a set of final course scores:

```
final.course.scores <-
    c( 92.1, 85.4, 78.7, 91.3,
       90.0, 93.0, 61.1, 80.0 )
```

Let's use a simplified grading scheme:

- If a student scores 90 or above, the student receives an "A" letter grade.
- If the student scores 80 or above, but less than 90, then the student receives a "B" letter grade.
- If the student scores 70 or above, but less than 90, then the student receives a "C" letter grade.

- If the student scores strictly less than 70, then the student receives an "E" letter grade.

Convert the vector of final course scores to letter grades using the `cut` function. Be careful about how your handle the endpoints.

**Solution**

To solve this problem, we need to use the `cut()` function.

First, we'll construct a vector with the cut points:

```
cut.points.vector <-
    c(0, 70, 80, 90, 100)
```

Now let's try using the `cut()` function with the `final.course.scores` vector, using the cutpoints in the `cut.points.vector`:

```
cut(
    final.course.scores,
    breaks = cut.points.vector
)
```

```
## [1] (90,100] (80,90]  (70,80]  (90,100] (80,90]  (90,100] (0,70]   (70,80]
## Levels: (0,70] (70,80] (80,90] (90,100]
```

Now let's put in some labels:

```
cut(
    final.course.scores,
    labels = c( "E", "C", "B", "A" ),
    breaks = cut.points.vector
)
```

```
## [1] A B C A B A E C
## Levels: E C B A
```

But this isn't quite right – the fifth student scored a 90.0, and therefore should receive an "A" grade, but the code assigns a "B" letter grade to this student. Similarly, the eighth student scored an 80.0, and therefore should receive a "B", but in fact was assigned a "C". That's because these grades are right on the cut points, and the default setting is that the intervals for the `cut()` function are open on the left and closed on the right. That means that the interval for a "B" grade is the range strictly greater than 80 and less than or equal to 90. But that's not how we specified the grading scheme! Instead, we want the intervals to be closed on the left and open on the right. To do this, we need to set the `right` option to `FALSE`:

```
cut(
    final.course.scores,
    labels = c( "E", "C", "B", "A" ),
    breaks = cut.points.vector,
    right = FALSE
)
```

```
## [1] A B C A A A E B
## Levels: E C B A
```

Now it works properly!