Creating sequences automatically

Why would we want to create vectors or data tables on our own?

No available data can be due to many reasons. For example, the lack of time, e.g. for conducting tests, or the lack of resources like money.

- Privacy reasons, i.e. to be able to share data that looks like real data but is made up for sharing.
- To create a toy dataset for asking questions on Stackoverflow.
- Padding of time series, e.g. create regular spaced data out of irregular transactional data.

Sequences for equal spacing (1/2)

1. Basic sequences

```
1:12
```

```
1  2  3  4  5  6  7  8  9  10  11  12
```

2. Specific intervals

```
seq(from=1 to=12, by=3)
```

```
1  4  7  10
```

e.g. the first month of each quarter of a year.

```
seq(from=1, to=6, by=0.25)
```

```
1.25 1.5 … 5.75 6.00
```

e.g. the Swiss grading scale.

Sequences for equal spacing (1/2)

1. **Basic sequences**

```
1:12
```

```
1  2  3  4  5  6  7  8  9  10  11  12
```

2. **Specific intervals**

```
seq(from=1 to=12, by=3)
```

```
1  4  7  10
```

e.g. the first month of each quarter of a year.

```
seq(from=1, to=6, by=0.25)
```

```
1.25 1.5 … 5.75 6.00
```

e.g. the Swiss grading scale.

Sidenote: Unnamed arguments

You can both specify arguments by name or unnamed.

```
seq(from=1, to=12, by=3)
```

```
seq(1,12,3)
```

```
1, 4, 7, 10
```

However, you can only change the order of arguments when they are explicitly named!

(look in the help file to see the correct order)

```
seq(1,3,12)
```

Now you create a sequence from 1 to 3, by steps of 12, since the arguments are not properly ordered.

```
1
```

```
seq(from=1, by=3, to=12)
```

```
1, 4, 7, 10
```

Sidenote: Unnamed arguments

You can both specify arguments by name or unnamed.

```
seq(from=1, to=12, by=3)
```

```
seq(1,12,3)
```

```
1, 4, 7, 10
```

However, you can only change the order of arguments when they are explicitly named!

(look in the help file to see the correct order)

```
seq(1,3,12)
```

Now you create a sequence from 1 to 3, by steps of 12, since the arguments are not properly ordered.

```
1
```

```
seq(from=1, by=3, to=12)
```

```
1, 4, 7, 10
```

Sequences for equal spacing (2/2)

3. Characters

```
LETTERS[seq(from=1, to=12, by=3)]
"A" "D" "G" "J"
```

Upper case letters can be generated with `LETTERS[]`, lower case letters with `letters[]` (1)

4. Dates

The same can be done with dates by `"day"`, `"week"`, `"month"`, or `"year"` (2)

```
seq(from=ymd("2011-01-01"), to=ymd("2011-03-31"), by="month")
"2011-01-01" "2011-02-01" "2011-03-01"
```

Sequences for equal spacing (2/2)

3. **Characters**

```
LETTERS[seq(from=1, to=12, by=3)]
"A" "D" "G" "J"
```

Upper case letters can be generated with `LETTERS[]`, lower case letters with `letters[]` (1)

4. **Dates**

The same can be done with dates by `"day"`, `"week"`, `"month"`, or `"year"` (2)

```
seq(from=ymd("2011-01-01"), to=ymd("2011-03-31"), by="month")
"2011-01-01" "2011-02-01" "2011-03-01"
```

Replicate a number or vector

`rep()` replicates the numbers of a vector:

- **Continuous replications**

`c()` specifies the vector to be repeated (1)

`times` indicates how many times to repeat the vector (2)

```
rep(c(1:12), times=3)
```

```
1 2 ... 11 12 1 2 ... 11 12 1 2 ... 11 12
```

e.g. the months for three years

- **Consecutive repetitions**

`each` indicates how many times to repeat each element of the vector (3)

```
rep(c(1:12), each=3)
```

```
1, 1, 1, 2, 2, 2, … 11, 11, 11, 12, 12, 12
```

e.g. grouped month for three years:

Replicate a number or vector

`rep()` replicates the numbers of a vector:

- **Continuous replications**

`c()` specifies the vector to be repeated (1)

`times` indicates how many times to repeat the vector (2)

```
rep(c(1:12), times=3)
```

```
1 2 ... 11 12 1 2 ... 11 12 1 2 ... 11 12
```

e.g. the months for three years

- **Consecutive repetitions**

`each` indicates how many times to repeat each element of the vector (3)

```
rep(c(1:12), each=3)
```

```
1, 1, 1, 2, 2, 2, … 11, 11, 11, 12, 12, 12
```

e.g. grouped month for three years:

Create a table with all possible permutations of the given variables

`expand.grid()` creates a data.frame from all combinations of the supplied vectors or factors.

```
Example <- expand.grid(Cust_Id=165:167,
Day=1:2,Month=1,Year=2017)
```

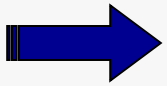
Cust_Id	Day	Month	Year
165	1	1	2017
166	1	1	2017
167	1	1	2017
165	2	1	2017
166	2	1	2017
167	2	1	2017

The output is a `data.frame` object with 4 variables and 6 observations

Sidenote: Concatenate strings

Example: the dates of transactions are saved by day, month, and year in 3 separate columns. It would be more useful to create one column with the complete date.

Cust_Id	Day	Month	Year
165	1	1	2017
166	1	1	2017
167	1	1	2017
165	2	1	2017
166	2	1	2017
167	2	1	2017



Cust_Id	...	Date
165	...	01-01-1017
166	...	01-01-1017
167	...	01-01-1017
165	...	02-01-1017
166	...	02-01-1017
167	...	02-01-1017

```
CustData[,Date := dmy(paste(Day,Month,Year,sep="-"))]
```

`sep` will combine the elements for each vector by a given symbol and will return single elements

Sidenote: Finding unique elements

- **Return the unique entries of a data table**

```
unique(CustData)
```

`unique()` applied to a data table removes all duplicated rows ❶

- **Return the unique elements of a vector**

```
unique(CustData[,Store])
…: 1 2
```

`unique()` applied to a vector returns all unique values ❷

Cust_Id	Store	Store_Cust
165	1	1_165
166	1	1_166
167	1	1_167
165	2	2_165
166	2	2_166
167	2	2_167

Table1: CustData

➢ To count the number of unique entries / elements, combine the command with the `length()` command.