Selecting columns

Abhjit Dasgupta

You can select columns by name or position, of course.

```
weather1 %>% select(year, element)
```

```
## # A tibble: 3 x 2
##    year element
##   <dbl> <chr>  
## 1  2010 tmax   
## 2  2010 tmin   
## 3  2010 tmax   
```

```
weather1 %>% select(2, 4, 5)
```

```
## # A tibble: 3 x 3
##    year element    d1
##   <dbl> <chr>   <dbl>
## 1  2010 tmax       NA
## 2  2010 tmin       NA
## 3  2010 tmax       NA
```

> `weather1` is just a data.frame with the first 3 rows of the weather dataset

You can select consecutive columns

```
weather1 %>% select(d1:d14)
```

```
## # A tibble: 3 x 14
##      d1    d2    d3    d4    d5    d6    d7    d8 d9      d10   d11 d12     d13   d14
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl> <dbl> <dbl> <lgl> <dbl> <dbl>
## 1    NA  NA    NA      NA    NA    NA    NA    NA NA       NA  NA   NA       NA    NA
## 2    NA  NA    NA      NA    NA    NA    NA    NA NA       NA  NA   NA       NA    NA
## 3    NA  27.3  24.1    NA    NA    NA    NA    NA NA       NA  29.7 NA       NA    NA
```

```
weather1 %>% select(3:7)
```

```
## # A tibble: 3 x 5
##   month element    d1    d2    d3
##   <dbl> <chr>   <dbl> <dbl> <dbl>
## 1     1 tmax       NA  NA    NA  
## 2     1 tmin       NA  NA    NA  
## 3     2 tmax       NA  27.3  24.1
```

You can also select columns based on some criteria in the column names.

```
weather1 %>% select(starts_with("d"))
```

```
## # A tibble: 3 x 31
##      d1    d2    d3    d4    d5    d6    d7    d8 d9      d10   d11 d12     d13   d14   d15   d16
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl> <dbl> <dbl> <lgl> <dbl> <dbl> <dbl> <dbl>
## 1    NA  NA    NA      NA    NA    NA    NA    NA NA       NA  NA   NA       NA    NA    NA    NA
## 2    NA  NA    NA      NA    NA    NA    NA    NA NA       NA  NA   NA       NA    NA    NA    NA
## 3    NA  27.3  24.1    NA    NA    NA    NA    NA NA       NA  29.7 NA       NA    NA    NA    NA
## # … with 15 more variables: d17 <dbl>, d18 <lgl>, d19 <lgl>, d20 <lgl>, d21 <lgl>, d22 <lgl>,
## #   d23 <dbl>, d24 <lgl>, d25 <dbl>, d26 <dbl>, d27 <dbl>, d28 <dbl>, d29 <dbl>, d30 <dbl>,
## #   d31 <dbl>
```

```
weather1 %>% select(ends_with('2'))
```

```
## # A tibble: 3 x 3
##      d2 d12   d22
##   <dbl> <lgl> <lgl>
## 1  NA   NA    NA
## 2  NA   NA    NA
## 3  27.3 NA    NA
```

> Whatever is in these selector functions `starts_with`, `ends_with`, etc., has to be quoted

You can select by column type

```
weather1 %>% select(where(is.character))
```

```
## # A tibble: 3 x 2
##   id      element
##   <chr>   <chr>  
## 1 MX17004 tmax   
## 2 MX17004 tmin   
## 3 MX17004 tmax   
```

```
weather1 %>% select(!where(is.numeric)) # Not numeric
```

```
## # A tibble: 3 x 10
##   id      element d9    d12   d18   d19   d20   d21   d22   d24  
##   <chr>   <chr>   <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl>
## 1 MX17004 tmax    NA    NA    NA    NA    NA    NA    NA    NA   
## 2 MX17004 tmin    NA    NA    NA    NA    NA    NA    NA    NA   
## 3 MX17004 tmax    NA    NA    NA    NA    NA    NA    NA    NA   
```

There are several other options:

```
?tidyselect::language
```

`:` for selecting a range of consecutive variables.

`!` for taking the complement of a set of variables.

`&` and `|` for selecting the intersection or the union of two sets of variables.

`c()` for combining selections.

In addition, you can use selection helpers. Some helpers select specific columns:

`everything()`: Matches all variables.

`last_col()`: Select last variable, possibly with an offset.

These helpers select variables by matching patterns in their names:

`starts_with()`: Starts with a prefix.

`ends_with()`: Ends with a suffix.

`contains()`: Contains a literal string.

`matches()`: Matches a regular expression.

`num_range()`: Matches a numerical range like x01, x02, x03.

```
?tidyselect::language
```

These helpers select variables from a character vector:

`all_of()`: Matches variable names in a character vector. All names must be present, otherwise an out-of-bounds error is thrown.

`any_of()`: Same as all_of(), except that no error is thrown for names that don't exist.

This helper selects variables with a function:

`where()`: Applies a function to all variables and selects those for which the function returns TRUE.