Select columns

Prof. Dr. Nicolas Meseth

Select columns

This chapter introduces tools to remove unnecessary columns from the data set. Or, if stated in a positive manner, we learn how to specify the columns we need for our analysis. As with most data transformation operations, we mostly introduce functions from the `{dplyr}` package.

The `select` command

The function `select()` is the designated tool to select columns with `{dplyr}`. By passing different things to the function, we can efficiently define the set of columns in the resulting data frame.

By column names

The easiest and intuitive way to specify the columns we want is by listing their names. We can pass one or more column names to the `select()` function. In case of two or more, we use commas to separate the names:

```
# Just one column name
orders %>%
  select(order_id)
```

```
# A tibble: 2,874 x 1
        order_id
           <dbl>
 1 1130007101519
 2 1130014965839
 3 1130026958927
 4 1130030563407
 5 1130038853711
 6 1130045964367
 7 1130050519119
 8 1130060283983
 9 1130102194255
10 1130106880079
# ... with 2,864 more rows
```

```
#> # A tibble: 2,874 x 1
#>        order_id
#>           <dbl>
#> 1 1130007101519
#> 2 1130014965839
#> 3 1130026958927
#> ...

# A list of column names
orders %>%
  select(order_id, total_price)
```

```
# A tibble: 2,874 x 2
        order_id total_price
           <dbl>       <dbl>
 1 1130007101519        94.7
 2 1130014965839        32.2
 3 1130026958927        30.2
 4 1130030563407        32.2
 5 1130038853711        30.2
 6 1130045964367        30.2
 7 1130050519119        30.2
 8 1130060283983        32.2
 9 1130102194255        96.7
10 1130106880079        32.2
# ... with 2,864 more rows
```

```
#> # A tibble: 2,874 x 2
#>        order_id total_price
#>           <dbl>       <dbl>
#> 1 1130007101519        94.7
#> 2 1130014965839        32.2
#> 3 1130026958927        30.2
#> ...
```

When we only want a few columns, this approach works fine and is usually a good choice. I expect you apply this method in more than 90% of all cases. However, there are cases when you'd wish there was something more flexible. Luckily, there is.

By name patterns

Names starting with a string

Sometimes we want to select columns based on a pattern of their names. Take the orders data set as an example. Here, all variables that contain information about the shipping address have the prefix `shipping`. We leverage this with the helper function `starts_with()`:

```
orders %>%
  select(starts_with("shipping")) %>%
  colnames()
```

```
[1] "shipping_address_city"      "shipping_address_zip"
[3] "shipping_address_country"   "shipping_address_latitude"
[5] "shipping_address_longitude"
```

```
#> [1] "shipping_address_city"      "shipping_address_zip"       "shipping_address_country
#> [4] "shipping_address_latitude"  "shipping_address_longitude"
```

Names ending with a string

Names with a string anywhere

Using regular expressions

By data type

```
orders %>%
  select(where(is.numeric))
```

```
# A tibble: 2,874 x 30
        order_id order_~1 app_id curre~2 curre~3 curre~4 curre~5 total~6 total~7
           <dbl>    <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 1130007101519     1014 580111    94.7    94.7       2       0       2    96.7
 2 1130014965839     1015 580111    32.2    32.2       0       0       0    32.2
 3 1130026958927     1016 580111    30.2    30.2       2       0       2    32.2
 4 1130030563407     1017 580111    32.2    32.2       0       0       0    32.2
 5 1130038853711     1018 580111    30.2    30.2       2       0       2    32.2
 6 1130045964367     1019 580111    30.2    30.2       2       0       2    32.2
 7 1130050519119     1020 580111    30.2    30.2       2       0       2    32.2
 8 1130060283983     1021 580111    32.2    32.2       0       0       0    32.2
 9 1130102194255     1022 580111    96.7    96.7       0       0       0    96.7
10 1130106880079     1023 580111    32.2    32.2       0       0       0    32.2
# ... with 2,864 more rows, 21 more variables: total_outstanding <dbl>,
#   total_price <dbl>, total_tax <dbl>, total_tip_received <dbl>,
#   location_id <dbl>, customer_id <dbl>, customer_accepts_marketing <dbl>,
#   customer_is_hsos <dbl>, customer_orders_count <dbl>,
#   customer_total_spent <dbl>, customer_last_order_id <dbl>,
#   customer_verified_email <dbl>, customer_tax_exempt <dbl>,
#   shipping_address_zip <dbl>, shipping_address_latitude <dbl>, ...
```

```
orders %>%
  select(where(is.logical))
```

```
# A tibble: 2,874 x 3
   test  taxes_included customer_sms_marketing_consent
   <lgl> <lgl>          <lgl>
 1 FALSE TRUE           NA
 2 FALSE TRUE           NA
 3 FALSE TRUE           NA
 4 FALSE TRUE           NA
 5 FALSE TRUE           NA
 6 FALSE TRUE           NA
 7 FALSE TRUE           NA
 8 FALSE TRUE           NA
 9 FALSE TRUE           NA
10 FALSE TRUE           NA
# ... with 2,864 more rows
```

```r
orders %>%
  select(where(is.character))
```

```
# A tibble: 2,874 x 27
   name  discount_~1 finan~2 fulfi~3 sourc~4 landi~5 landi~6 note  tags  proce~7
   <chr> <chr>       <chr>   <chr>   <chr>   <chr>   <chr>   <chr> <chr> <chr>
 1 B1014 DCBPXGJB1J~ paid    fulfil~ web     /passw~ <NA>    <NA>  <NA>  direct
 2 B1015 <NA>        paid    fulfil~ web     /walle~ <NA>    <NA>  <NA>  express
 3 B1016 KYOD5MNEZB~ paid    fulfil~ web     /       <NA>    <NA>  <NA>  express
 4 B1017 <NA>        paid    fulfil~ web     /walle~ <NA>    <NA>  <NA>  express
 5 B1018 DCBPXGJB1J~ paid    fulfil~ web     <NA>    <NA>    <NA>  <NA>  express
 6 B1019 DCBPXGJB1J~ paid    fulfil~ web     <NA>    <NA>    <NA>  <NA>  express
 7 B1020 DCBPXGJB1J~ paid    fulfil~ web     <NA>    <NA>    <NA>  <NA>  express
 8 B1021 <NA>        paid    fulfil~ web     /       <NA>    <NA>  <NA>  express
 9 B1022 <NA>        paid    fulfil~ web     /walle~ <NA>    <NA>  <NA>  express
10 B1023 <NA>        paid    fulfil~ web     <NA>    <NA>    <NA>  <NA>  express
# ... with 2,864 more rows, 17 more variables: payment_details_gateway <chr>,
#   payment_details_credit_card_company <chr>,
#   customer_marketing_opt_in_level <chr>, customer_gender <chr>,
#   customer_state <chr>, customer_note <chr>, customer_tags <chr>,
#   customer_last_order_name <chr>, campaign_tag <chr>,
#   shipping_address_city <chr>, shipping_address_country <chr>,
#   billing_address_city <chr>, billing_address_country <chr>, ...
```

```r
orders %>%
  select(where(is.factor))
```

```
# A tibble: 2,874 x 0
```

```r
orders %>%
  select(where(is.list))
```

```
# A tibble: 2,874 x 0
```

```r
# The package lubridate provides a function to check for date (without time)
orders %>%
  select(where(lubridate::is.Date))
```

```
# A tibble: 2,874 x 0
```

```r
# Select all date/time columns
orders %>%
  select(where(lubridate::is.POSIXct))
```

```
# A tibble: 2,874 x 8
   created_at          updated_at          processed_at
   <dttm>              <dttm>              <dttm>
 1 2019-05-24 12:59:16 2019-06-19 13:23:26 2019-05-24 12:59:15
 2 2019-05-24 13:09:08 2019-06-21 14:40:07 2019-05-24 13:09:07
 3 2019-05-24 13:22:41 2019-06-21 12:35:23 2019-05-24 13:22:40
 4 2019-05-24 13:27:43 2019-06-21 14:27:18 2019-05-24 13:27:42
 5 2019-05-24 13:36:46 2019-06-21 12:11:57 2019-05-24 13:36:45
 6 2019-05-24 13:44:41 2019-06-21 14:37:21 2019-05-24 13:44:41
 7 2019-05-24 13:49:21 2019-06-21 12:25:16 2019-05-24 13:49:20
 8 2019-05-24 13:59:57 2019-06-21 11:49:47 2019-05-24 13:59:57
 9 2019-05-24 14:43:53 2019-06-19 14:12:38 2019-05-24 14:43:53
10 2019-05-24 14:48:16 2019-06-21 15:54:24 2019-05-24 14:48:16
# ... with 2,864 more rows, and 5 more variables:
#   customer_accepts_marketing_updated_at <dttm>, customer_created_at <dttm>,
#   customer_updated_at <dttm>, cancelled_at <dttm>, closed_at <dttm>
```