Working with data frames

STAT221

Package dplyr

We will use the R add-on package dplyr that provides a set of tools for common data manipulation tasks, i.e. filtering rows or calculating summary statistics.

If the package dplyr is not available on your computer, you first have to install it once. The easiest way of installing a package is through the RStudio package interface.

You can make all dplyr functionality available by

```
library(dplyr)
```

We will use the starwars dataset as an example, which is directly available in dplyr, and contains information about 87 Star Wars characters from the Star Wars API (https://swapi.dev/).

```
data(starwars)
starwars
```



```
## # A tibble: 87 x 14
##    name  height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke…    172    77 blond      fair       blue            19   male  mascu…
##  2 C-3PO    167    75 <NA>       gold       yellow         112   none  mascu…
##  3 R2-D2     96    32 <NA>       white, bl… red             33   none  mascu…
##  4 Dart…    202   136 none       white      yellow          41.9 male  mascu…
##  5 Leia…    150    49 brown      light      brown           19   fema… femin…
##  6 Owen…    178   120 brown, gr… light      blue            52   male  mascu…
##  7 Beru…    165    75 brown      light      blue            47   fema… femin…
##  8 R5-D4     97    32 <NA>       white, red red             NA   none  mascu…
##  9 Bigg…    183    84 black      light      brown           24   male  mascu…
## 10 Obi-…    182    77 auburn, w… fair       blue-gray       57   male  mascu…
## # … with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```


The data is stored as a `tibble`, which is quite similar to the `data.frame` class.

The pipe operator

Let's assume that we want to filter a subset of rows that contain all the droids (information in the species column) and only return the columns name, height, and mass. dplyr provides a convenient function `filter()` to select a subset of rows, that requires an expression, which returns a logical value. All the rows are kept for which the condition evaluates to `TRUE`. The analogous function to select columns is called `select()`.

Hence, we could first filter rows and then select columns:

```
select(filter(starwars, species == "Droid"), c("name", "height", "mass"))
```

```
## # A tibble: 6 x 3
##   name   height  mass
##   <chr>   <int> <dbl>
## 1 C-3PO     167    75
## 2 R2-D2      96    32
## 3 R5-D4      97    32
## 4 IG-88     200   140
## 5 R4-P17     96    NA
## 6 BB8        NA    NA
```

But when we want to use several operators on the same dataset, the code becomes quickly unclear and messy. An alternative is the pipe command `%>%` that allows to stack several function calls, using the previous output as the new input.

```
starwars %>%
  filter(species == "Droid") %>%
  select(c("name", "height", "mass"))
```



```
## # A tibble: 6 x 3
##   name   height  mass
##   <chr>   <int> <dbl>
## 1 C-3PO     167    75
## 2 R2-D2      96    32
## 3 R5-D4      97    32
## 4 IG-88     200   140
## 5 R4-P17     96    NA
## 6 BB8        NA    NA
```


Additional functions are available, i.e. selecting the smallest or largest value with `slice_min()` or `slice_max()`, or selecting a random sample of rows with `slice_sample()`.

Add columns

We can also add a new column with transformed variables using function `mutate()`.

For example, we can calculate the Body Mass Index for the droids:

```
starwars %>%
  filter(species == "Droid") %>%
  select(c("name", "height", "mass")) %>%
  mutate(bmi = mass / (height/100)^2)
```

```
## # A tibble: 6 x 4
##   name   height  mass   bmi
##   <chr>   <int> <dbl> <dbl>
## 1 C-3PO     167    75  26.9
## 2 R2-D2      96    32  34.7
## 3 R5-D4      97    32  34.0
## 4 IG-88     200   140  35
## 5 R4-P17     96    NA  NA
## 6 BB8        NA    NA  NA
```

C-3PO seems to be slightly overweight and all other droids are classified as obese (BMI > 30).

Group and Summarise

The function `summarise()` is letting us to define summary statistics, i.e. calculating the average height and mass for all characters in the dataset.

We can also use the convenient function `n()` to count rows.

```
starwars %>%
  summarise(N = n(), aveheight = mean(height, na.rm=TRUE), avemass = mean(mass, na.rm=TRUE))
```



```
## # A tibble: 1 x 3
##       N aveheight avemass
##   <int>     <dbl>   <dbl>
## 1    87      174.    97.3
```


Instead of calculating a single summary statistic for all the data, we might want to group by gender and species.


```
starwars %>%
  group_by(species, gender) %>%
  summarise(N = n(), aveheight = mean(height, na.rm=TRUE), avemass = mean(mass, na.rm=TRUE)) %>%
  print(n=42)
```

```
## `summarise()` regrouping output by 'species' (override with `.groups` argument)
```

```
## # A tibble: 42 x 5
## # Groups:   species [38]
##    species        gender        N aveheight avemass
##    <chr>          <chr>     <int>     <dbl>   <dbl>
##  1 Aleena         masculine     1        79     15
##  2 Besalisk       masculine     1       198    102
##  3 Cerean         masculine     1       198     82
##  4 Chagrian       masculine     1       196    NaN
##  5 Clawdite       feminine      1       168     55
##  6 Droid          feminine      1        96    NaN
##  7 Droid          masculine     5       140     69.8
##  8 Dug            masculine     1       112     40
##  9 Ewok           masculine     1        88     20
## 10 Geonosian      masculine     1       183     80
## 11 Gungan         masculine     3       209.    74
## 12 Human          feminine      9       160.    56.3
## 13 Human          masculine    26       182.    87.0
## 14 Hutt           masculine     1       175   1358
## 15 Iktotchi       masculine     1       188    NaN
## 16 Kaleesh        masculine     1       216    159
## 17 Kaminoan       feminine      1       213    NaN
## 18 Kaminoan       masculine     1       229     88
## 19 Kel Dor        masculine     1       188     80
## 20 Mirialan       feminine      2       168     53.1
## 21 Mon Calamari   masculine     1       180     83
## 22 Muun           masculine     1       191    NaN
## 23 Nautolan       masculine     1       196     87
## 24 Neimodian      masculine     1       191     90
## 25 Pau'an         masculine     1       206     80
## 26 Quermian       masculine     1       264    NaN
## 27 Rodian         masculine     1       173     74
## 28 Skakoan        masculine     1       193     48
## 29 Sullustan      masculine     1       160     68
## 30 Tholothian     feminine      1       184     50
## 31 Togruta        feminine      1       178     57
## 32 Toong          masculine     1       163     65
## 33 Toydarian      masculine     1       137    NaN
## 34 Trandoshan     masculine     1       190    113
## 35 Twi'lek        feminine      1       178     55
## 36 Twi'lek        masculine     1       180    NaN
## 37 Vulptereen     masculine     1        94     45
## 38 Wookiee        masculine     2       231    124
## 39 Xexto          masculine     1       122    NaN
## 40 Yoda's species masculine     1        66     17
## 41 Zabrak         masculine     2       173     80
## 42 <NA>           <NA>          4       181.    48
```