

# Tidyverse. Dplyr

Dr.Sc. Oleksii Yehorchenkov

Department of Spatial Planning

# Reading data

```
1 library(dplyr)
2 library(readr)
3
4 msleep <- read_csv("../data/msleep.csv", show_col_types = FALSE)
```

# Data. Mammals sleep

Data: The **msleep** (mammals sleep) data set contains the sleeptimes and weights for a set of mammals and is available in the [dagdata](#) repository on github. This data set contains 83 rows and 11 variables.

```
1 glimpse(msleep)
```

```
Rows: 83
```

```
Columns: 11
```

```
$ name      <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Greater  
shor...
```

```
$ genus     <chr> "Acinonyx", "Aotus", "Aplodontia", "Blarina", "Bos",  
"Bra...
```

```
$ vore      <chr> "carni", "omni", "herbi", "omni", "herbi", "herbi",  
"carn...
```

```
$ order     <chr> "Carnivora", "Primates", "Rodentia", "Soricomorpha",  
"Art...
```

```
$ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu", NA,  
"dome...
```

```
$ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4, 8.7, 7.0, 10.1, 3.0,  
5...
```

```
$ sleep_rem  <dbl> NA, 1.0, 0.4, 0.2, 0.7, 0.0, 1.4, NA, 0.0, NA, 0.0, 0.0
```

# The columns of the dataset:

column name	Description
name	common name
genus	taxonomic rank
vore	carnivore, omnivore or herbivore?
order	taxonomic rank
conservation	the conservation status of the mammal
sleep_total	total amount of sleep, in hours
sleep_rem	rem sleep, in hours
sleep_cycle	length of sleep cycle, in hours
awake	amount of time spent awake, in hours
brainwt	brain weight in kilograms
bodywt	body weight in kilograms

# Important dplyr verbs to remember

dplyr verbs	Description
<code>select()</code>	select columns
<code>filter()</code>	filter rows
<code>arrange()</code>	re-order or arrange rows
<code>mutate()</code>	create new columns
<code>summarise()</code>	summarise values
<code>group_by()</code>	allows for group operations in the “split-apply-combine” concept

# dplyr verbs in action

The two most basic functions are `select()` and `filter()` which selects columns and filters rows, respectively.

# Selecting columns using `select()`

Select a set of columns: the *name* and the *sleep\_total* columns.

```
1 sleepData <- select(msleep, name, sleep_total)
2 head(sleepData)
```

```
# A tibble: 6 × 2
  name                sleep_total
  <chr>                <dbl>
1 Cheetah              12.1
2 Owl monkey           17
3 Mountain beaver     14.4
4 Greater short-tailed shrew 14.9
5 Cow                   4
6 Three-toed sloth     14.4
```

To select all the columns except a specific column, use the **"-" (subtraction)** operator (also known as negative indexing)

```
1 head(select(msleep, -c("name", "sleep_total")))
```

```
# A tibble: 6 × 9
  genus      vore  order conservation sleep_rem sleep_cycle awake  brainwt
bodywt
  <chr>      <chr> <chr> <chr>                <dbl>        <dbl> <dbl>      <dbl>
<dbl>
1 Acinonyx  carni  Carn... lc                NA            NA      11.9 NA        50
2 Aotus     omni  Prim... <NA>             1.8          NA       7  0.0155
0.48
3 Aplodon... herbi  Rode... nt                2.4          NA      9.6 NA
1.35
4 Blarina  omni  Sori... lc                2.3          0.133    9.1  0.00029
0.019
5 Bos      herbi  Arti... domesticated    0.7          0.667    20  0.423  600
6 Bradypus herbi  Pilo... <NA>            2.2          0.767    9.6 NA
3.85
```



To select a range of columns by name, use the ":" (colon) operator

```
1 head(select(msleep, name:order))
```

```
# A tibble: 6 × 4
```

	name	genus	vore	order
	<chr>	<chr>	<chr>	<chr>
1	Cheetah	Acinonyx	carni	Carnivora
2	Owl monkey	Aotus	omni	Primates
3	Mountain beaver	Aplodontia	herbi	Rodentia
4	Greater short-tailed shrew	Blarina	omni	Soricomorpha
5	Cow	Bos	herbi	Artiodactyla
6	Three-toed sloth	Bradypus	herbi	Pilosa

To select all columns that start with the character string “sl”, use the function `starts_with()`

```
1 head(select(msleep, starts_with("sl")))
```

```
# A tibble: 6 × 3
  sleep_total sleep_rem sleep_cycle
  <dbl>      <dbl>      <dbl>
1      12.1      NA      NA
2       17       1.8      NA
3      14.4       2.4      NA
4      14.9       2.3    0.133
5        4       0.7    0.667
6      14.4       2.2    0.767
```

Some additional options to select columns based on a specific criteria include

- `ends_with()` = Select columns that end with a character string
- `contains()` = Select columns that contain a character string
- `matches()` = Select columns that match a regular expression
- `one_of()` = Select columns names that are from a group of names

# Selecting rows using `filter()`

Filter the rows for mammals that sleep a total of more than 16 hours.

```
1 filter(msleep, sleep_total >= 16)
```

```
# A tibble: 8 × 11
```

	name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle	awake
	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	
	<dbl>								
1	Owl mo...	Aotus	omni	Prim...	<NA>	17	1.8	NA	7
2	Long-n...	Dasy...	carni	Cing...	lc	17.4	3.1	0.383	
	6.6								
3	North ...	Dide...	omni	Dide...	lc	18	4.9	0.333	6
4	Big br...	Epte...	inse...	Chir...	lc	19.7	3.9	0.117	
	4.3								
5	Thick-...	Lutr...	carni	Dide...	lc	19.4	6.6	NA	
	4.6								
6	Little...	Myot...	inse...	Chir...	<NA>	19.9	2	0.2	
	4.1								
7	...	...	...	...	...	18.1	6.1	NA	

Filter the rows for mammals that sleep a total of more than 16 hours and have a body weight of greater than 1 kilogram.

```
1 filter(msleep, sleep_total >= 16, bodywt >= 1)
```

```
# A tibble: 3 × 11
  name      genus vore  order conservation sleep_total sleep_rem sleep_cycle
<chr>      <chr> <chr> <chr> <chr>              <dbl>      <dbl>      <dbl>
<dbl>
1 Long-n... Dasy... carni Cing... lc              17.4        3.1        0.383
6.6
2 North ... Dide... omni  Dide... lc              18          4.9        0.333    6
3 Giant ... Prio... inse... Cing... en              18.1        6.1        NA
5.9
# i 2 more variables: brainwt <dbl>, bodywt <dbl>
```

# Filter the rows for mammals in the Perissodactyla and Primates taxonomic order

```
1 filter(msleep, order %in% c("Perissodactyla", "Primates"))
```

# A tibble: 15 × 11

	name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle	awake	
	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>		
	<dbl>									
1	Owl m...	Aotus	omni	Prim...	<NA>	17	1.8	NA	7	
2	Grivet	Cerc...	omni	Prim...	lc	10	0.7	NA	14	
3	Horse	Equus	herbi	Peri...	domesticated	2.9	0.6	1		
21.1										
4	Donkey	Equus	herbi	Peri...	domesticated	3.1	0.4	NA		
20.9										
5	Patas...	Eryt...	omni	Prim...	lc	10.9	1.1	NA		
13.1										
6	Galago	Gala...	omni	Prim...	<NA>	9.8	1.1	0.55		
14.2										
7										

You can use the boolean operators (e.g. `>`, `<`, `>=`, `<=`, `!=`, `%in%`) to create the logical tests.

# Pipe operator: `|>`

Before we go any further, let's introduce the pipe operator: `|>`. This operator allows you to pipe the output from one function to the input of another function. Instead of nesting functions (reading from the inside to the outside), the idea of piping is to read the functions from left to right.

**Note.** There is another pipe operator `%>%` from **magrittr** package with the same functionality. The base pipe operator was introduced in R 4.1.0 in 2021 and it is recommended for use in your code.

Here's an example you have seen:

```
1 head(select(msleep, name, sleep_total))
```

Now in this case, we will pipe the `msleep` data frame to the function that will select two columns (*name* and *sleep\_total*) and then pipe the new data frame to the function `head()` which will return the head of the new data frame.

```
1 msleep |>
2   select(name, sleep_total) |>
3   head()
```

```
# A tibble: 6 × 2
  name                sleep_total
  <chr>              <dbl>
1 Cheetah             12.1
2 Owl monkey          17
3 Mountain beaver     14.4
4 Greater short-tailed shrew 14.9
5 Cow                  4
6 Three-toed sloth    14.4
```

# Back to dplyr verbs in action

Now that you know about the **pipe operator** (`|>`), we will use it throughout the rest of this tutorial.

You will soon see how useful the **pipe operator** is when we start to combine many functions.



# Arrange or re-order rows using `arrange()`

To arrange (or re-order) rows by a particular column such as the *taxonomic order*, list the name of the column you want to arrange the rows by

```
1 msleep |>
2   select(name, order, sleep_total) |>
3   arrange(order) |>
4   head()
```

# A tibble: 6 × 3

	name	order	sleep_total
	<chr>	<chr>	<dbl>
1	Tenrec	Afrosoricida	15.6
2	Cow	Artiodactyla	4
3	Roe deer	Artiodactyla	3
4	Goat	Artiodactyla	5.3
5	Giraffe	Artiodactyla	1.9
6	Sheep	Artiodactyla	3.8

Same as above, except here we filter the rows for mammals that sleep for 16 or more hours instead of showing the head of the final data frame

```
1 msleep |>
2   select(name, order, sleep_total) |>
3   arrange(order, sleep_total) |>
4   filter(sleep_total >= 16)
```

# A tibble: 8 × 3

	name <chr>	order <chr>	sleep_total <dbl>
1	Big brown bat	Chiroptera	19.7
2	Little brown bat	Chiroptera	19.9
3	Long-nosed armadillo	Cingulata	17.4
4	Giant armadillo	Cingulata	18.1
5	North American Opossum	Didelphimorphia	18
6	Thick-tailed opossum	Didelphimorphia	19.4
7	Owl monkey	Primates	17
8	Arctic ground squirrel	Rodentia	16.6

Something slightly more complicated: same as above, except arrange the rows in the *sleep\_total* column in a descending order. For this, use the function `desc()`

```
1 msleep |>
2   select(name, order, sleep_total) |>
3   arrange(order, desc(sleep_total)) |>
4   filter(sleep_total >= 16)
```

```
# A tibble: 8 × 3
```

	name	order	sleep_total
	<chr>	<chr>	<dbl>
1	Little brown bat	Chiroptera	19.9
2	Big brown bat	Chiroptera	19.7
3	Giant armadillo	Cingulata	18.1
4	Long-nosed armadillo	Cingulata	17.4
5	Thick-tailed opossum	Didelphimorphia	19.4
6	North American Opossum	Didelphimorphia	18
7	Owl monkey	Primates	17
8	Arctic ground squirrel	Rodentia	16.6

# Create new columns using `mutate()`

The `mutate()` function will add new columns to the data frame. Create a new column called *rem\_proportion* which is the ratio of rem sleep to total amount of sleep.

```
1 msleep |>
2   mutate(rem_proportion = sleep_rem / sleep_total) |>
3   select(name, sleep_rem, sleep_total, rem_proportion) |>
4   head()
```

```
# A tibble: 6 × 4
```

	name <chr>	sleep_rem <dbl>	sleep_total <dbl>	rem_proportion <dbl>
1	Cheetah	NA	12.1	NA
2	Owl monkey	1.8	17	0.106
3	Mountain beaver	2.4	14.4	0.167
4	Greater short-tailed shrew	2.3	14.9	0.154
5	Cow	0.7	4	0.175
6	Three-toed sloth	2.2	14.4	0.153

You can many new columns using `mutate()` (separated by commas). Here we add a second column called *bodywt\_grams* which is the *bodywt* column in grams.

```
1 msleep |>
2   mutate(rem_proportion = sleep_rem / sleep_total,
3           bodywt_grams = bodywt * 1000,
4           wt_ge_50 = if_else(bodywt >= 50, TRUE, FALSE)) |>
5   select(name, sleep_rem, sleep_total,
6           rem_proportion, bodywt, bodywt_grams, wt_ge_50) |>
7   head()
```

```
# A tibble: 6 × 7
```

	name	sleep_rem	sleep_total	rem_proportion	bodywt	bodywt_grams	wt_ge_50
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<lgl>
1	Cheetah	NA	12.1	NA	50	50000	TRUE
2	Owl monkey	1.8	17	0.106	0.48	480	FALSE
3	Mountain b...	2.4	14.4	0.167	1.35	1350	FALSE
4	Greater sh...	2.3	14.9	0.154	0.019	19	FALSE
5	Cow	0.7	4	0.175	600	600000	TRUE
6	Three-toed...	2.2	14.4	0.153	3.85	3850	FALSE

# Create summaries of the data frame using `summarise()`

The `summarise()` function will create summary statistics for a given column in the data frame such as finding the mean. For example, to compute the average number of hours of sleep, apply the `mean()` function to the column *sleep\_total* and call the summary value *avg\_sleep*.

```
1 msleep |>
2   summarise(avg_sleep = mean(sleep_total))

# A tibble: 1 × 1
  avg_sleep
  <dbl>
1      10.4
```

There are many other summary statistics you could consider such `sd()`, `min()`, `max()`, `median()`, `sum()`, `n()` (returns the length of vector), `first()` (returns first value in vector), `last()` (returns last value in vector) and `n_distinct()` (number of distinct values in vector).

```
1 msleep |>
2   summarise(avg_sleep = mean(sleep_total),
3             min_sleep = min(sleep_total),
4             max_sleep = max(sleep_total),
5             total = n())
```

```
# A tibble: 1 × 4
  avg_sleep min_sleep max_sleep total
  <dbl>      <dbl>      <dbl> <int>
1    10.4        1.9       19.9     83
```

# Group operations using `group_by()`

The `group_by()` verb is an important function in dplyr. As we mentioned before it's related to concept of “split-apply-combine”. We literally want to split the data frame by some variable (e.g. taxonomic order), apply a function to the individual data frames and then combine the output.



Let's do that: split the **msleep** data frame by the taxonomic order, then ask for the same summary statistics as above. We expect a set of summary statistics for each taxonomic order.

```
1 msleep |>
2   group_by(order) |>
3   summarise(avg_sleep = mean(sleep_total),
4             min_sleep = min(sleep_total),
5             max_sleep = max(sleep_total),
6             total = n()) |>
7   head()
```

# A tibble: 6 × 5

	order	avg_sleep	min_sleep	max_sleep	total
	<chr>	<dbl>	<dbl>	<dbl>	<int>
1	Afrosoricida	15.6	15.6	15.6	1
2	Artiodactyla	4.52	1.9	9.1	6
3	Carnivora	10.1	3.5	15.8	12
4	Cetacea	4.5	2.7	5.6	3
5	Chiroptera	19.8	19.7	19.9	2
6	Cingulata	17.8	17.4	18.1	2

# References

This tutorial is based on [https://genomicsclass.github.io/book/pages/dplyr\\_tutorial.html](https://genomicsclass.github.io/book/pages/dplyr_tutorial.html)