R-Ladies | R consortium

📅 **Monday, November 28th**

🕐 **16.30**

🏠 **Keilaranta 7, Espoo**

Event sponsor: ROVIO

If you are ready….

Let’s get it started!

For Beginners

r_ladies_helsinki_dplyr - RStudio

Untitled1*

```
1 getwd()
```

1:8 (Top Level) R Script

Console Terminal Background Jobs

R 4.1.3 · ~/Desktop/r_ladies_helsinki_dplyr/

```
> getwd()
[1] "/Users/hazelkavili/Desktop/r_ladies_helsinki_dplyr"
>
```

Environment History Connections Tutorial

Data

game_data 720 obs. of 6 variables

Files Plots Packages Help Viewer Presentation

Home > Desktop > r_ladies_helsinki_dplyr

Name	Size	Modified
..		
.RData	9.1 KB	Nov 5, 2022, 10:46 AM
.Rhistory	2.9 KB	Nov 5, 2022, 10:46 AM
basics.R	128 B	Nov 19, 2022, 6:15 PM
dplyr_data_set.csv	30 KB	Oct 22, 2022, 3:37 PM
dplyr_doc.html	617.6 KB	Nov 5, 2022, 10:29 AM
dplyr_doc.Rmd	1.3 KB	Nov 5, 2022, 10:29 AM
r_ladies_dplyr.R	1.3 KB	Nov 19, 2022, 6:43 PM
r_ladies_ggplot2.R	351 B	Oct 22, 2022, 4:05 PM
r_ladies_helsinki_dplyr.Rproj	205 B	Nov 20, 2022, 2:56 PM

Some basics about R!

- #this is R-Ladies Helsinki
- A <- 10
- a <- 3

-myNumbers <- c(1:10)
-myNumbers
-rep(myNumbers, times = 3)
-twice <- rep(myNumbers, each = 2)

Some basics about R!

R commands;

- are case sensitive
- can be separated either by a semi-colon (‘;’), or by a newline
- #comment

Assignment, Basic Operators

Assignments

- use **<-**

Basic arithmetic operators

- +,-,*,/,^, %%

Logical operators

- <, >, <=, >=, ==, !=, !x, x & y, x | y

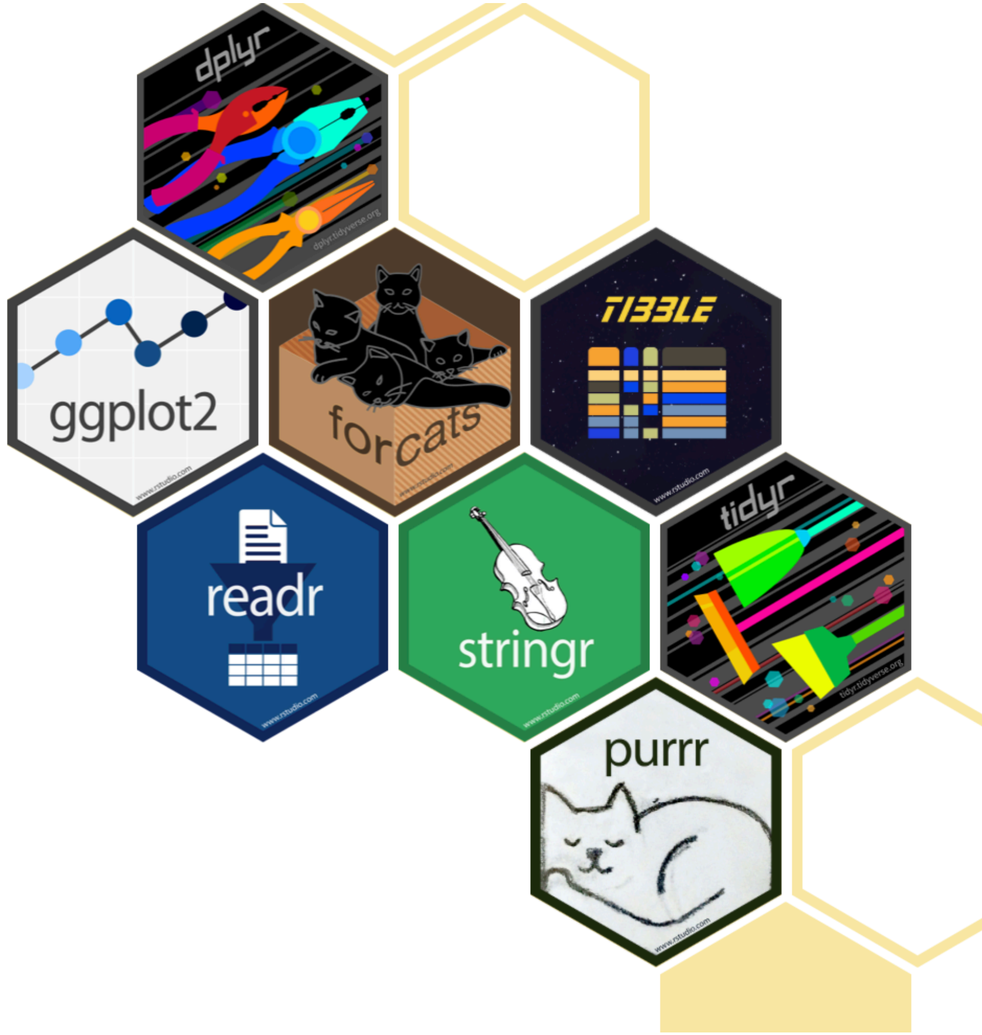
Others

- sum, sqrt, min, max, mean, var, sd, abs, summary

Tidyverse

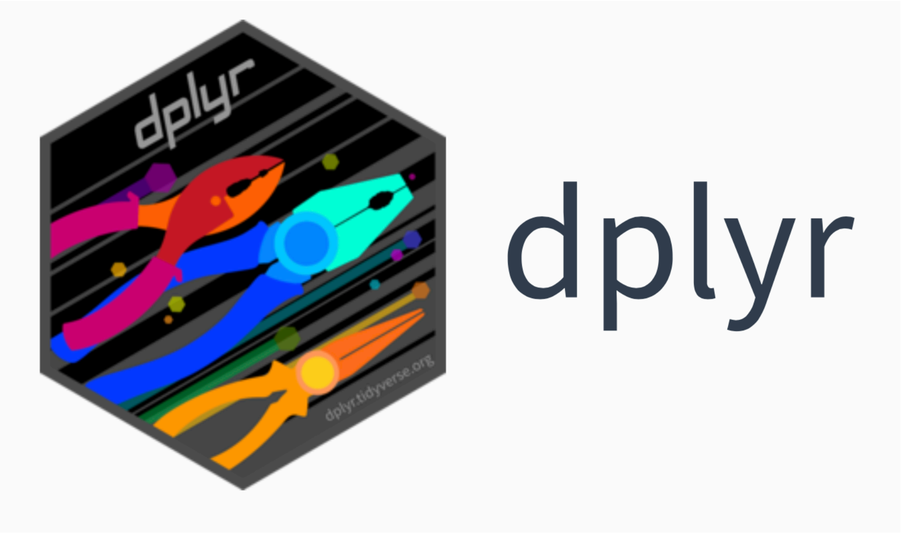
```
install.packages("tidyverse")

library(tidyverse)
```



dplyr

- A package contains set of verbs to do data manipulation.



Check the verbs we will learn today!

- `select()`
- `filter()`
- `mutate()`
- `group_by()`
- `summarise()`

Loading today's data set!

Option 1

```
game_data <- read_csv("https://tinyurl.com/3a5hxnra")
```

Option 2

```
game_data <- read_csv("dplyr_data_set.csv")
```

Get familiar today's data set!

Option 1

```
glimpse(game_data)
```

Option 2

```
game_data %>%
        View()
```

What is that?

Get familiar today's data set!

Option 1

```r
glimpse(game_data)
```

```
> glimpse(game_data)
Rows: 720
Columns: 6
$ date              <date> 2021-01-01, 2021-01-01, 2021-01-01, 2021-01-…
$ game_name         <chr> "game2", "game4", "game3", "game1", "game2", …
$ genre             <chr> "puzzle", "rpg", "rpg", "puzzle", "puzzle", "…
$ daily_active_users <dbl> 169546, 99376, 676, 875956, 181696, 105046, 5…
$ in_app_revenue    <dbl> 409.183, 252.451, 2.109, 1984.495, 420.468, 2…
$ platform          <chr> "iOS", "iOS", "iOS", "iOS", "iOS", "iOS", "iO…
```

Get familiar today's data set!

date	game_name	genre	daily_active_users	in_app_revenue	platform
2021-01-01	game2	puzzle	169546	409.183	iOS
2021-01-01	game4	rpg	99376	252.451	iOS
2021-01-01	game3	rpg	676	2.109	iOS
2021-01-01	game1	puzzle	875956	1984.495	iOS
2021-01-01	game2	puzzle	113168	619.304	Android
2021-01-01	game4	rpg	66388	382.088	Android
2021-01-01	game3	rpg	588	3.192	Android
2021-01-01	game1	puzzle	584108	3003.560	Android
2021-01-02	game2	puzzle	181696	420.468	iOS
2021-01-02	game4	rpg	105046	265.697	iOS
2021-01-02	game3	rpg	586	2.553	iOS
2021-01-02	game1	puzzle	922306	2100.231	iOS
2021-01-02	game2	puzzle	121268	636.384	Android
2021-01-02	game4	rpg	70168	402.136	Android
2021-01-02	game3	rpg	528	3.864	Android
2021-01-02	game1	puzzle	615008	3178.728	Android

```
game_data %>%
    View()
```

Pipe operator %>%

- Chaining multiple functions
- Giving an input to function —> using it's output as another input to another function
- cmd + shift + m
- kntr + shft + m

Pipe operator %>%

```
mtcars %>%
  select(mpg, cyl) %>%
    filter(mpg > 20)
```

```
select(mtcars, mpg, cyl) %>%
  filter(mpg > 20)
```

select()

- Choosing is not losing
- This function returns a modified copy, doesn't change the data.
- `select(dataframe, col1, col2,…)`
- `select(dataframe, 1:4)`

```
24 game_data %>%
25   select(date, game_name, in_app_revenue)
```

24:1 (Top Level)

Console Terminal Background Jobs

R 4.1.3 · ~/Desktop/r_ladies_helsinki_dplyr/

```
> game_data %>%
+   select(date, game_name, in_app_revenue)
# A tibble: 720 × 3
   date       game_name in_app_revenue
   <date>     <chr>             <dbl>
 1 2021-01-01 game2            409.
 2 2021-01-01 game4            252.
 3 2021-01-01 game3              2.11
 4 2021-01-01 game1           1984.
 5 2021-01-02 game2            420.
 6 2021-01-02 game4            266.
 7 2021-01-02 game3              2.55
 8 2021-01-02 game1           2100.
 9 2021-01-03 game2            423.
10 2021-01-03 game4            272.
```

Let's try assignment sign <– with select function

```
• game_data_revenue <-
    game_data %>%
    select(date, game_name,
in_app_revenue)

• View(game_data_revenue)

• glimpse(game_data_revenue)
```

We selected few columns

from original data set

AND

assigned it to a new one

called

game_data_revenue

filter()

- Filter out rows, specific type of **observation.**

- `filter(dataframe, logical_test)`

```
30 #filter
31 game_data %>%
32   filter(date >= '2021-01-01' & date < '2021-04-01')
```

30:1 (Top Level)

Console Terminal Render Background Jobs

R 4.1.3 · ~/Desktop/r_ladies_helsinki_dplyr/

```
> game_data %>%
+   filter(date >= '2021-01-01' & date < '2021-04-01')
# A tibble: 720 × 6
   date       game_name genre  daily_active_users
   <date>     <chr>     <chr>               <dbl>
 1 2021-01-01 game2     puzzle             169546
 2 2021-01-01 game4     rpg                 99376
 3 2021-01-01 game3     rpg                   676
 4 2021-01-01 game1     puzzle             875956
 5 2021-01-02 game2     puzzle             181696
 6 2021-01-02 game4     rpg                105046
 7 2021-01-02 game3     rpg                   586
 8 2021-01-02 game1     puzzle             922306
 9 2021-01-03 game2     puzzle             189286
10 2021-01-03 game4     rpg                 98476
```

mutate()

- Deals with info in your data which is not display

- `mutate(dataframe, new = var1 + var2)`

- `mutate(my_df, x = a + b, y = x + c)`

```
#mutate
game_data %>%
  select(date,in_app_revenue, daily_active_users) %>%
  mutate(iap_per_dau = in_app_revenue / daily_active_users) %>%
  mutate(iap_per_dau_in_euros = iap_per_dau * 1.02)
```

48:1 (Top Level) R Script

Console Terminal Render Background Jobs

R 4.1.3 · ~/Desktop/r_ladies_helsinki_dplyr/

```
+   mutate(iap_per_dau_in_euros = iap_per_dau * 1.02)
# A tibble: 720 × 5
  date       in_app_revenue daily_active_users iap_per_dau
  <date>              <dbl>              <dbl>       <dbl>
1 2021-01-01         409.               169546     0.00241
2 2021-01-01         252.                99376     0.00254
```

Your turn!

- **Can you create a column: which shows only the month of data recorded?**

  - `install.packages("lubridate")`
    `library(lubridate) # a package to use on data time objects.`

  - functions as `day(), week(), month(), year()` can be used on a "date" data.

  - HINT:

    ```
    your_data_frame %>%
    m———(your_variable_name = month(date))
    ```

Answer is here:

- Can you create a column: which shows only the month of data recorded?

```
> game_data %>%
+     select(date) %>%
+     mutate(month_of = month(date))
# A tibble: 720 × 2
   date       month_of
   <date>        <dbl>
 1 2021-01-01        1
 2 2021-01-01        1
 3 2021-01-01        1
 4 2021-01-01        1
```

group_by() & summarise()

- `group_by()` gets the above functions to operate group-by-group rather than on the entire dataset.

- `summarise()` builds a new dataset that contains only the summarising statistics.

- `summarise(dataframe, newColname = expression,. . .)`

- `summarise(dataframe, sum = sum(A), avg = mean(B)...)`

group_by() & summarise()

```
49 game_data %>%
50   select(date, daily_active_users, platform) %>%
51   group_by(platform) %>%
52   summarise(avg_dau = mean(daily_active_users))
```

52:48 (Top Level)

Console | Terminal | Render | Background Jobs

R 4.1.3 · ~/Desktop/r_ladies_helsinki_dplyr/

```
> game_data %>%
+   select(date, daily_active_users, platform) %>%
+   group_by(platform) %>%
+   summarise(avg_dau = mean(daily_active_users))
# A tibble: 2 × 2
  platform avg_dau
  <chr>      <dbl>
1 Android  199798.
2 iOS      299491.
```

Your turn!

- Can you find:

  average IAP revenue by genre for only iOS?

Answer is here:

- Can you find average IAP revenue by genre for only iOS?

```
> game_data %>%
+   select(date, in_app_revenue, platform, genre) %>%
+   filter(platform == 'iOS') %>%
+   group_by(genre) %>%
+   summarise(avg_iap_revenue = mean(in_app_revenue))
# A tibble: 2 × 2
  genre  avg_iap_revenue
  <chr>            <dbl>
1 puzzle           1216.
2 rpg               136.
```

Is there more functions in dplyr?

- `arrange`
- `distinct`
- `rename`
- `count`
- binding two data frames, or joining data frames are possible with `left_join`, `inner_join`, `right_join`, `full_join` or `bind_rows`, `bind_cols`
- subset rows using their position: `slice`