r_notebook

Advantages of R Notebooks!

Notebooks are a ***fantastic tool*** when it comes to writing R scripts for research. This is because they make the code ***easy to comprehend and inter-operable,*** making them a good practice of **open science!**

Let’s look at the same R code again, but now with the advantages of a notebook!

1. Demarcate code more clearly with markdown!

1. 1 Load the libraries

```r
suppressPackageStartupMessages({
  library(dplyr)
  library(ggplot2)
})
```

1. 2 Create the data set

```r
# create fitness data
source("R/create_synth_data.R")
```

```

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```r
df_fitness <- create_fitness_data(num_participants = 10, seed = 123)
```

2. Interact with the data as you code!

2.1 *Look at chunks of data within the notebook!*

```r
head(df_fitness)
```

```
  study_number            Datetime week_id is_compliant_day avg_heart_rate
1            1 2021-01-01 12:00:00       1             TRUE           85.6
2            1 2021-01-01 12:01:00       1             TRUE           70.7
3            1 2021-01-01 12:02:00       1             TRUE           71.3
4            1 2021-01-01 12:03:00       1             TRUE           87.1
5            1 2021-01-01 12:04:00       1             TRUE           74.6
6            1 2021-01-01 12:05:00       1             TRUE           57.3
```

2. Make operations on data more visually accessible!

2.1 Consider code chunk below with a single output

```
daily_avg_hr_per_study <- df_fitness |>
  mutate(day = as.Date(Datetime)) |>
  group_by(study_number, day) |>
  summarise(avg_daily_heart_rate = mean(avg_heart_rate, na.rm = TRUE), .groups = 'drop')

head(daily_avg_hr_per_study)
```

```
# A tibble: 6 × 3
  study_number day        avg_daily_heart_rate
         <int> <date>                    <dbl>
1            1 2021-01-01                 70.4
2            1 2021-01-02                 69.2
3            1 2021-01-03                 69.5
4            1 2021-01-04                 67.6
5            1 2021-01-05                 66.1
6            1 2021-01-06                 65.3
```

2.1 Break it into parts!

First step

```
mutated_data <- df_fitness |>
  mutate(day = as.Date(Datetime),.after = study_number)

head(mutated_data)
```

```
  study_number        day            Datetime week_id is_compliant_day
1            1 2021-01-01 2021-01-01 12:00:00       1             TRUE
2            1 2021-01-01 2021-01-01 12:01:00       1             TRUE
3            1 2021-01-01 2021-01-01 12:02:00       1             TRUE
4            1 2021-01-01 2021-01-01 12:03:00       1             TRUE
5            1 2021-01-01 2021-01-01 12:04:00       1             TRUE
6            1 2021-01-01 2021-01-01 12:05:00       1             TRUE
  avg_heart_rate
1           85.6
2           70.7
3           71.3
4           87.1
5           74.6
6           57.3
```

Second step!

```
grouped_data <- mutated_data |>
  group_by(study_number, day)

head(grouped_data)
```

```
# A tibble: 6 × 6
# Groups:   study_number, day [1]
  study_number day        Datetime            week_id is_compliant_day
         <int> <date>     <dttm>                <int> <lgl>
1            1 2021-01-01 2021-01-01 12:00:00       1 TRUE
```

```
2             1 2021-01-01 2021-01-01 12:01:00       1 TRUE
3             1 2021-01-01 2021-01-01 12:02:00       1 TRUE
4             1 2021-01-01 2021-01-01 12:03:00       1 TRUE
5             1 2021-01-01 2021-01-01 12:04:00       1 TRUE
6             1 2021-01-01 2021-01-01 12:05:00       1 TRUE
# ℹ 1 more variable: avg_heart_rate <dbl>
```

Third and final step!

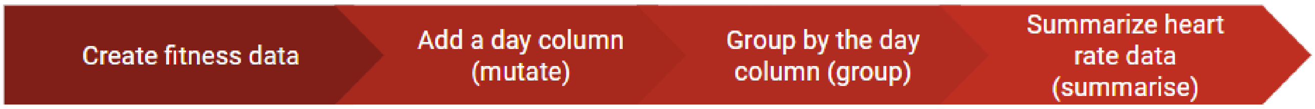
```
summarized_data <- grouped_data |>
  summarise(avg_daily_heart_rate = mean(avg_heart_rate, na.rm = TRUE), .groups = 'drop')

head(summarized_data)
```

```
# A tibble: 6 × 3
  study_number day        avg_daily_heart_rate
         <int> <date>                    <dbl>
1            1 2021-01-01                 70.4
2            1 2021-01-02                 69.2
3            1 2021-01-03                 69.5
4            1 2021-01-04                 67.6
5            1 2021-01-05                 66.1
6            1 2021-01-06                 65.3
```

3. Using additional features!

There are ***many more tools*** that can be used to improve the readability of your code! A non-exhaustive list includes: ***tables, images/figures, links etc.*** Let's revisit the same code chunks with these features!

3. 1 Process Overview



Create fitness data

Add a day column (mutate)

Group by the day column (group)

Summarize heart rate data (summarise)

3. 2 Variables overview

No.	Variable	Meaning
1.	mutated_data	Add a "**day**" column to the fitness data
2.	grouped_data	Group **mutated_data** according to the the **day** column
3.	summarized_data	Summarize the **average heart rate** data

3. 3 Additional resources

- Mutate function: stack overflow
- Grouping and summarizing data: medium article

3. 4 In line code segments

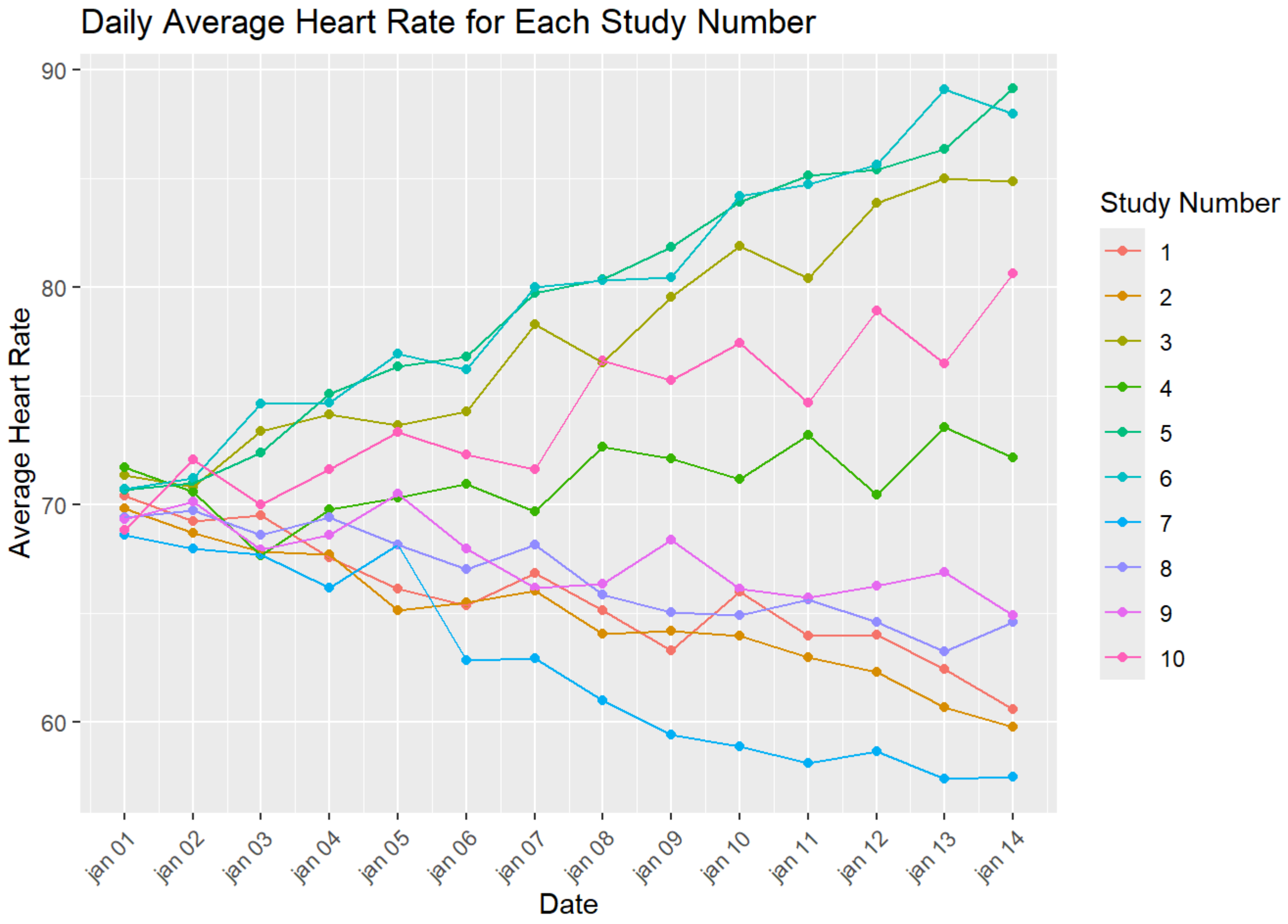
```
# Calculate mean and standard deviation
# Filter the data based on the condition and calculate mean and standard deviation for avg_hea
filtered_data <- df_fitness |>
  filter(study_number == 2)

# Calculate mean and standard deviation for avg_heart_rate
mean_value <- mean(filtered_data$avg_heart_rate, na.rm = TRUE)
std_value <- sd(filtered_data$avg_heart_rate, na.rm = TRUE)
```

The ***mean*** and ***standard deviation*** of ***heart rates*** for ***study 2*** are 64.9245487 and 10.4119959 respectively.

4. Visualize results (more easily)!

With notebooks, you can view multiple visualizations in with different cells!

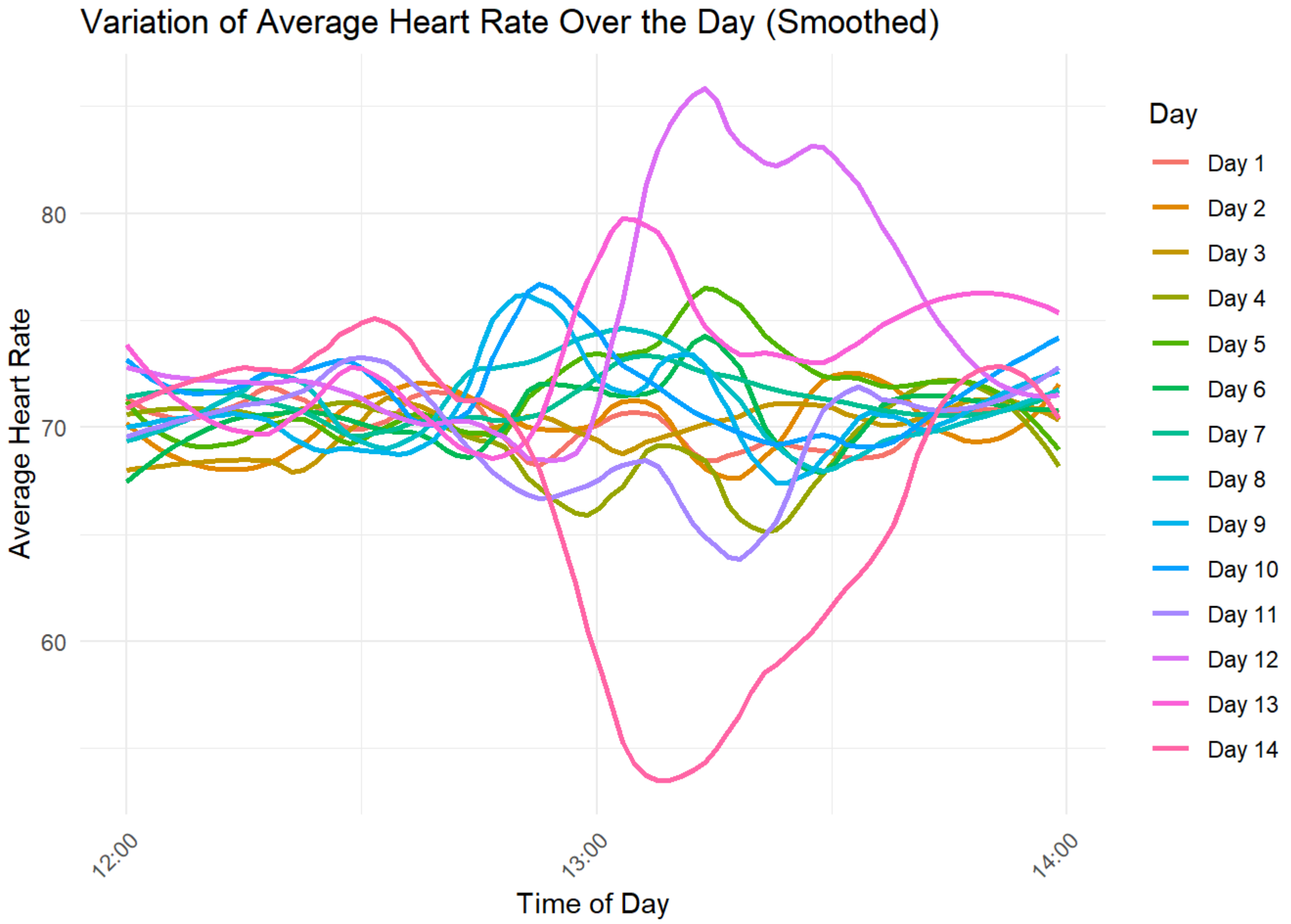
```
daily_avg_hr_per_study <- df_fitness |>
  mutate(day = as.Date(Datetime)) |>
  group_by(study_number, day) |>
  summarise(avg_daily_heart_rate = mean(avg_heart_rate, na.rm = TRUE), .groups = 'drop')

ggplot(daily_avg_hr_per_study, aes(x = day, y = avg_daily_heart_rate, color = as.factor(study_
  geom_point() +
  geom_line() +
  labs(
    title = "Daily Average Heart Rate for Each Study Number",
    x = "Date",
    y = "Average Heart Rate",
    color = "Study Number"
  ) +
  scale_x_date(date_breaks = "1 day", date_labels = "%b %d") +
  scale_color_hue() +
  #theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



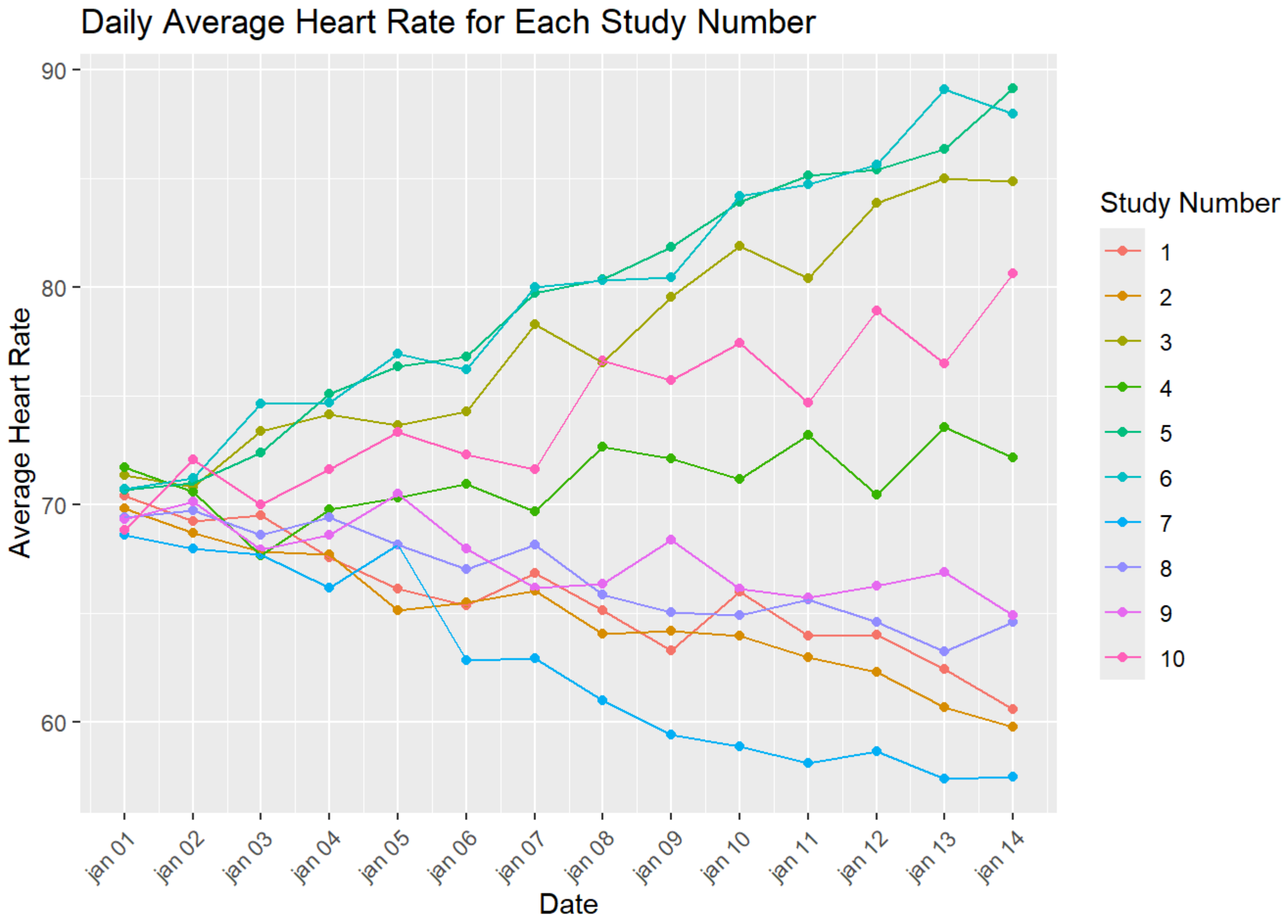
```
hourly_avg_hr <- df_fitness |>
  mutate(
    time_of_day = format(Datetime, "%H:%M"),
    day = as.Date(Datetime),
    day_label = paste("Day", as.integer(day - min(day) + 1))  # Assigns Day 1, Day 2, etc.
  ) |>
  group_by(day, time_of_day, day_label) |>
  summarise(avg_hr_across_studies = mean(avg_heart_rate, na.rm = TRUE), .groups = 'drop')

ggplot(hourly_avg_hr, aes(x = as.POSIXct(time_of_day, format = "%H:%M"), y = avg_hr_across_stu
  geom_smooth(se = FALSE, span = 0.3) +  # Using geom_smooth with a smaller span for smoothing
  labs(
    title = "Variation of Average Heart Rate Over the Day (Smoothed)",
    x = "Time of Day",
    y = "Average Heart Rate",
    color = "Day"
  ) +
  scale_x_datetime(date_breaks = "1 hour", date_labels = "%H:%M") +  # Breaks every hour
  scale_color_hue() +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
Warning: Removed 10 rows containing non-finite outside the scale range
(`stat_smooth()`).
```



```r
daily_avg_hr_per_study <- df_fitness |>
  mutate(day = as.Date(Datetime)) |>
  group_by(study_number, day) |>
  summarise(avg_daily_heart_rate = mean(avg_heart_rate, na.rm = TRUE), .groups = 'drop')

ggplot(daily_avg_hr_per_study, aes(x = day, y = avg_daily_heart_rate, color = as.factor(study_
  geom_point() +
  geom_line() +
  labs(
    title = "Daily Average Heart Rate for Each Study Number",
    x = "Date",
    y = "Average Heart Rate",
    color = "Study Number"
  ) +
  scale_x_date(date_breaks = "1 day", date_labels = "%b %d") +
  scale_color_hue() +
  #theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



5. Export notebooks to Slides/ PDFs!

This makes viewing your notebook on different devices possible without an IDE!