Tracking time

With Clockify and R

Håvard R. Karlsen, NTNU – date: 2026-07-07

Photo by Icons8 Team on Unsplash

Introduction to this project

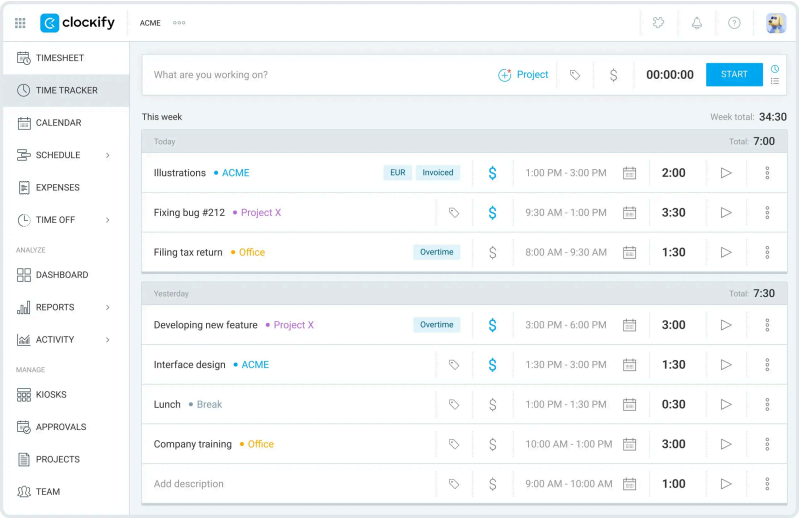
- Associate professors have “special independent position” (read: no fixed workhours) in Norway.
- To avoid burnout: **time tracking**
- To have fun: Create an R project that can help me with time tracking.
    - **I’ll show some of the process and functionality here.**

Two main motivating questions

1. Am I working too much or too little?
2. What am I spending my time on?

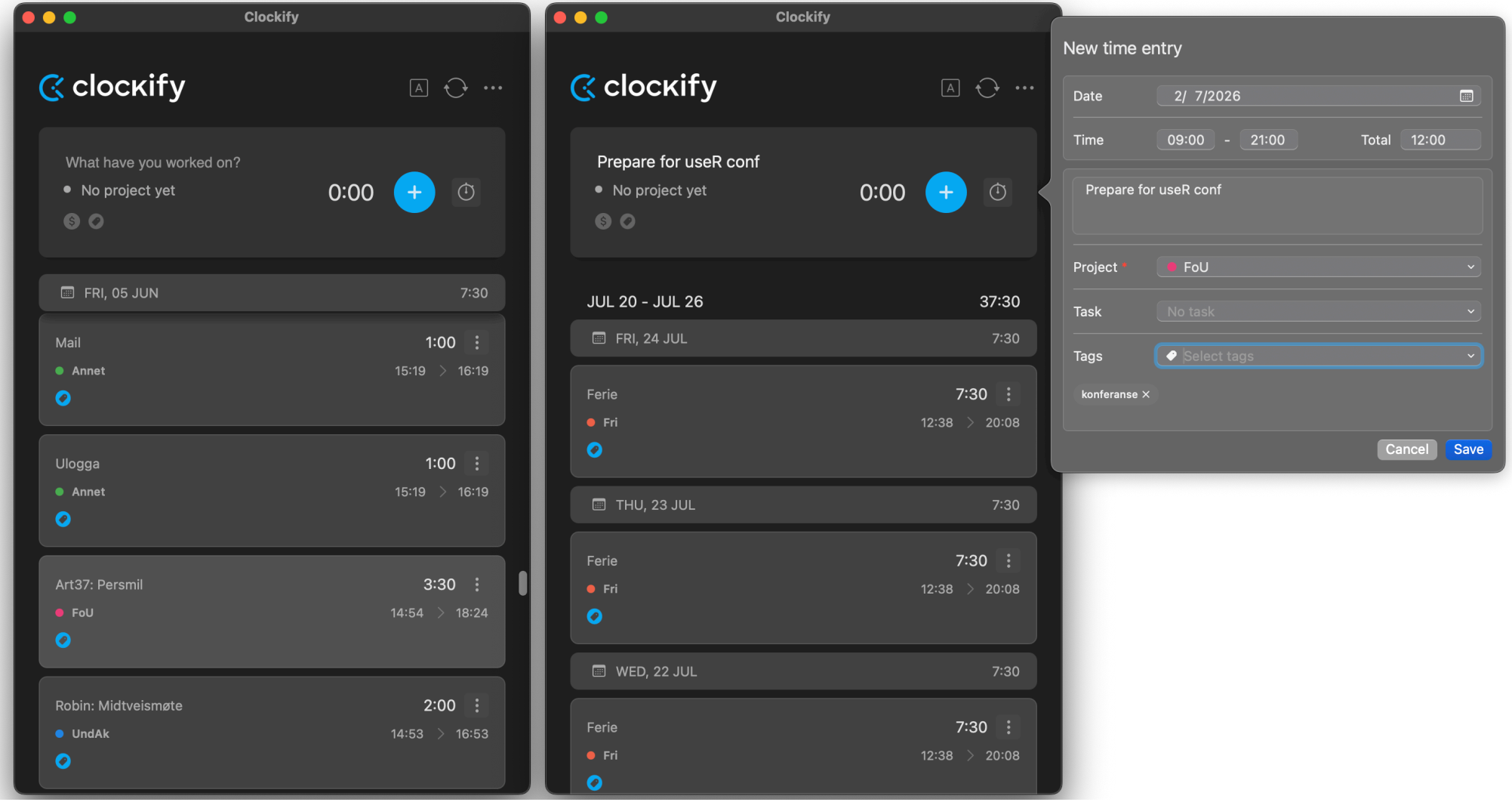
The tools

- Clockify: Freemium
- Clockify R package: Thanks to Andrew B. Collier
- Lubridate package: Thanks to Posit
- Idle time



Logging time in Clockify

Logging is quick and easy. I spend 1 minute at the end of each workday to log what I did that day.



Extracting logged time

```r
library(tidyverse)
library(clockify)

# Set up your API access first (not shown).

timelog <- clockify::reports_detailed("2026-01-01", "2026-01-31")
timelog |> select(-user_id)
```

```
# A tibble: 58 × 24
   id       description start offStart offEnd end   duration timeZone zonedStart
   <chr>    <chr>       <chr>    <int>  <int> <chr>    <int> <chr>    <chr>
 1 6980c30… KIP2: Plan… 2026…     3600   3600 2026…    27000 Europe/… 2026-01-3…
 2 697a26b… Ulogga      2026…     3600   3600 2026…     3600 Europe/… 2026-01-2…
 3 697a26a… Art37: Per… 2026…     3600   3600 2026…     3600 Europe/… 2026-01-2…
 4 697a269… Kollegapra… 2026…     3600   3600 2026…     5400 Europe/… 2026-01-2…
 5 697a269… PEHP-talk   2026…     3600   3600 2026…     3600 Europe/… 2026-01-2…
 6 697a268… KIP2: Plan… 2026…     3600   3600 2026…     7200 Europe/… 2026-01-2…
 7 697a266… Veiledning… 2026…     3600   3600 2026…     3600 Europe/… 2026-01-2…
 8 6978dde… Kollegapra… 2026…     3600   3600 2026…     1800 Europe/… 2026-01-2…
 9 6978dde… KIP2: Plan… 2026…     3600   3600 2026…    18000 Europe/… 2026-01-2…
10 6978ddd… Bach-pers:… 2026…     3600   3600 2026…    10800 Europe/… 2026-01-2…
# ℹ 48 more rows
# ℹ 15 more variables: zonedEnd <chr>, billable <lgl>, project_id <chr>,
#   task_id <chr>, approval_request_id <lgl>, type <chr>, is_locked <lgl>,
#   currency <chr>, user_name <chr>, user_email <chr>, project_name <chr>,
#   client_name <chr>, client_id <chr>, task_name <chr>, tags <list>
```

Cleaning logged time

```
1 timelog |> names()
```

```
 [1] "id"                 "description"        "user_id"
 [4] "start"              "offStart"           "offEnd"
 [7] "end"                "duration"           "timeZone"
[10] "zonedStart"         "zonedEnd"           "billable"
[13] "project_id"         "task_id"            "approval_request_id"
[16] "type"               "is_locked"          "currency"
[19] "user_name"          "user_email"         "project_name"
[22] "client_name"        "client_id"          "task_name"
[25] "tags"
```

More info than I need since Clockify is built for billing hours.

I care about, for each task logged

- Date
- Duration
- Project (broad category of work)
- Tag (specific category of work under project)
- Description (Specific task worked on)

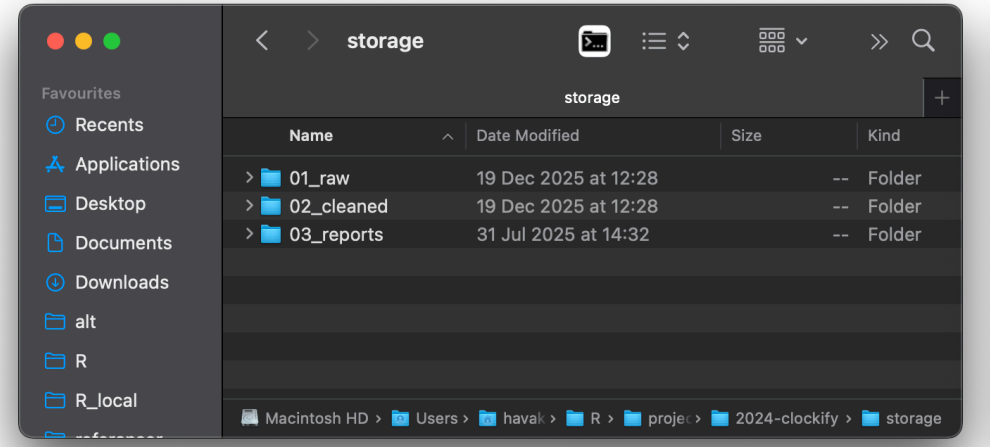
```
  1  timelog_clean |> head()
```

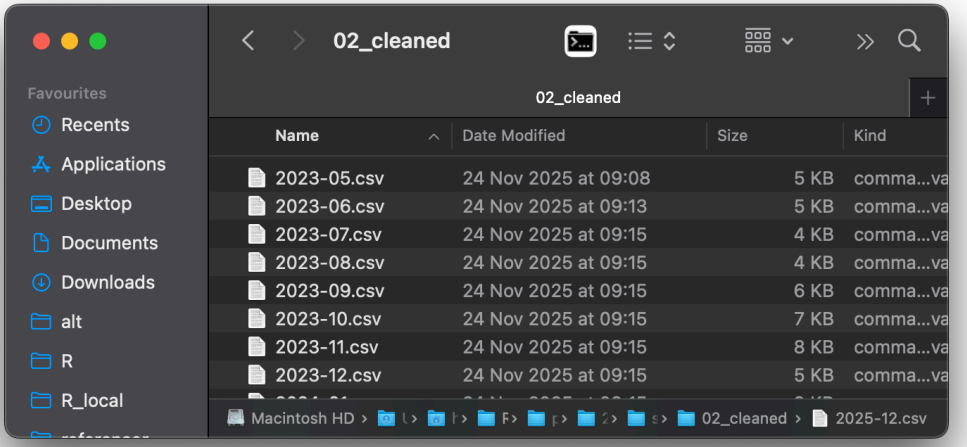
```
# A tibble: 6 × 6
# Rowwise:
  date       hours description       project_name task_name tags
  <date>     <dbl> <chr>             <chr>        <chr>     <chr>
1 2026-01-30   7.5 KIP2: Planlegging UndAk        kip2      emne;688b7a15406427…
2 2026-01-28   1   Ulogga            Annet        <NA>      ulogga;645cd7762b58…
3 2026-01-28   1   Art37: Persmil    FoU          <NA>      artikkel;688b7a2348…
4 2026-01-28   1.5 Kollegaprat/lunsj Annet        <NA>      kollegapause;688b7a…
5 2026-01-28   1   PEHP-talk         FoU          pehp      møter;645cd763d4e13…
6 2026-01-28   2   KIP2: Planlegging UndAk        kip2      emne;688b7a15406427…
```

Note that tags are still unwieldy.

Storing logged time

I store the raw and cleaned timelogs aggregated to each month.





Creating reports

1. Load the timelogs for the months you want from storage
2. Merge them
3. Do date calculations on them and summarise on whatever you want to know

1. Am I working too much or too little?

```
1 give_overtime(all_time)
```

```
For the period: 2023-05-01 UTC--2025-12-31 UTC
Contract hours: 5235
Offhours: 645.5
Should have worked: 4589.5
Worked hours: 4635.83333333333
Overtime status: 46.3333333333339
Note: Positive hours means you worked more than expected, negative means you worked
less.
```

I worked more than I should!

Hooray?

2. What am I spending my time on?

2024

Work allocation

Against intended workload

Project	Total hours	%	Intended %
Admin work	56.50	3.11	6
Research and development	737.75	40.55	47
Teaching	1025.25	56.35	47

Period: 2024-01-01 UTC--2024-12-31 UTC

2025

Work allocation

Against intended workload

Project	Total hours	%	Intended %
Admin work	12.00	0.85	6
Research and development	733.75	51.91	47
Teaching	667.75	47.24	47

Period: 2025-01-01 UTC--2025-10-31 UTC

Concluding thoughts

- Time is difficult
  - How many days are in the interval Jan 1st to Jan 31st? Technically 30 days.

```
timelog <- clockify::reports_detailed("2026-01-01", "2026-01-31")
```

```
timelog$start |> max()
```

```
[1] "2026-01-30T15:29:00Z"
```

```
interval(ymd("2024-01-01"), ymd("2024-01-31")) / days()
```

```
[1] 30
```

- Free tiers of Premium tools are volatile

```
timelog <- clockify::reports_detailed("2026-01-01", "2026-02-01")
```

```
ERROR [2026-07-02 15:55:21] message: Your workspace's subscription plan doesn't
support the selected date range.
```

- These types of small projects are a fun way to learn R.

Thanks for your attention!

Photo by Thomas Lipke on Unspl