rStrava

Hugo B Harrison

06 September, 2021

```
#devtools::install_github('fawda123/rStrava')
library(rStrava)
library(ggmap)
library(lubridate)
library(patchwork)
library(ggthemes)
library(RColorBrewer)
library(tidyverse)
```

1 Setup

rStrava provides a convenient wrappers to retrieve all your Strava activities and information. https://github.com/fawda123/rStrava

1.1 Strava Authentication

Downloading your Strava data requires a personal API and authentication token, both of which can be obtained on your Strava page Log in to Strava to create a personal API in the profile settings. Chose the application, save your client id, and create authentication token to paste below. Additional information about the personal API can be found here: https://developers.strava.com/ Every API retrieval function in the rStrava package requires an authentication token.

```
# name chosen by user
app_name <- 'Hugo Harrison'
# an integer, assigned by Strava
app_client_id  <- '68097'
# an alphanumeric secret, assigned by Strava
app_secret <- '3a9ba85cf988207fcb8ce7d82bff1fbc2c707fe9'

stoken <- httr::config(token = strava_oauth(app_name,
                                            app_client_id,
                                            app_secret,
                                            app_scope="activity:read_all"))

myinfo <- get_athlete(stoken, id = '24919795')
my_acts <- get_activity_list(stoken)
act_data <- compile_activities(my_acts) %>%
  filter(!start_latitude < 0)

mystrava = list(myinfo, my_acts, act_data)

save(my_acts, file = "my_acts.Rdata")
save(act_data, file = "act_data.Rdata")
```

1.2 Google Authentication

To map your activities in Google maps, you must also have a Google API and authentication token. Additional information about Google APIs can be found here: https://developers.google.com/maps/documentation/e

levation/overview#api_key Google will charges by the number of requests per months, so be mindful about sharing your google key.

Add the Google key to your R environment, do this only once. If you mess up (like I did, remove your Google key from the renvironement and recreate it)

```
#save the key, do only once
cat("google_key=whatever-your-google-key-is",
    file=file.path(normalizePath("~/"), ".Renviron"),
    append=TRUE)

#to remove
user_renviron = path.expand(file.path("~", ".Renviron"))
file.edit(user_renviron)
```

```
#retrieve the key, restart R if not found
mykey <- Sys.getenv("google_key")
register_google(mykey)

load(file = "act_data.Rdata")
load(fil = "my_acts.Rdata")
```

2 Strava activities

2.1 Compile all activities

Download all your Strava activities and select any information that is relevant to you.

I simply what to keep track what activities I do, when, and my performance throughout the year. A few things to note here as I wrangle the data: - Dates are a funny thing. - Depending on how you use the data, wrangle it here so that is it passed on to all further analyses.

```
act_sub = act_data %>% select(type, distance,
                              elapsed_time, moving_time,
                              start_date,
                              average_heartrate, max_heartrate,
                              average_speed, max_speed,
                              elev_high, elev_low, total_elevation_gain,
                              name) %>%
        #Convert dates to a useful format
  mutate(Date = as.Date(start_date),
         Year = year(Date),
         Month = month(Date, label = T, abbr = F),
         Week = week(Date),
         #Day of year 1-365 or month 1-31 is useful to keep track of
         #when an activity was done
         day_of_year = yday(start_date),
         day_of_month = mday(start_date),
         #Lubridate considers that the week starts on Sunday, which is ludicrous!
         #If we want the week to start on a Monday, we need to tell R.
         day = wday(Date, label = TRUE, week_start = getOption("lubridate.week.start", 1)),
         #Just some minor wrangling
         moving_time = moving_time / 60,
         average_heartrate = as.numeric(average_heartrate),
         max_heartrate= as.numeric(max_heartrate),
         type = as.factor(type)) %>%
  #I didn't start using Strava until May this year and I have a few odd activities
  #that I want to remove.
  filter(!Year == "2020",
         ! (type == "Ride" &  distance < 2)) %>%
  #A walk, a hike, whatever... as long as you're on 2 feet and bring snacks, it counts.
  mutate(type = str_replace(type, "Walk", "Hike"),
```

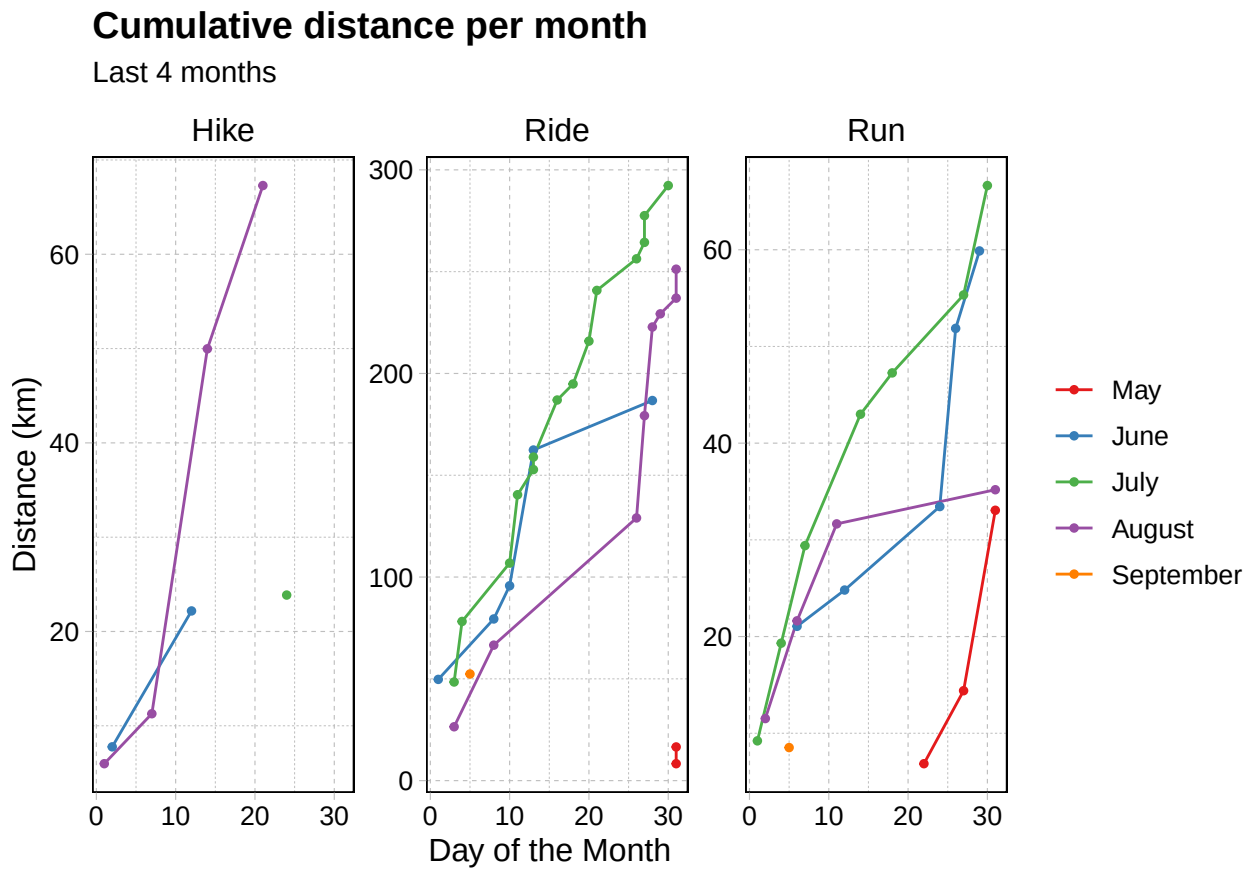
```
        type = str_replace(type, "Kayaking", "Kayak"),
        type = factor(type, levels = c("Hike", "Ride", "Run", "Kayak")))


head(act_sub)
```

```
##   type distance elapsed_time moving_time           start_date average_heartrate
## 1  Run   8.5201         3012    48.25000 2021-09-05T15:09:49Z             157.5
## 2 Ride  52.3885        14526   186.35000 2021-09-05T09:31:21Z              99.2
## 3 Ride  14.2286         1846    30.76667 2021-08-31T17:15:53Z             143.2
## 4  Run   3.5303         1902    28.40000 2021-08-31T16:40:39Z             153.1
## 5 Ride   7.7139         1785    24.80000 2021-08-31T16:10:39Z             133.0
## 6 Ride   6.4632         1537    21.10000 2021-08-29T06:34:40Z                NA
##   max_heartrate average_speed max_speed elev_high elev_low total_elevation_gain
## 1           181       10.5948     19.80      72.9      6.8                 78.7
## 2           136       16.8660     52.92      96.3      4.7                561.7
## 3           165       27.7488     59.76     115.3      6.8                 52.8
## 4           182        7.4592     20.16     317.1    107.2                217.8
## 5           173       18.6624     47.52     128.8      7.7                204.6
## 6            NA       18.3780    110.16      26.0      1.2                 36.9
##             name       Date Year     Month Week day_of_year day_of_month day
## 1  Afternoon Run 2021-09-05 2021 September   36         248            5 Sun
## 2   Morning Ride 2021-09-05 2021 September   36         248            5 Sun
## 3   Evening Ride 2021-08-31 2021    August   35         243           31 Tue
## 4  Afternoon Run 2021-08-31 2021    August   35         243           31 Tue
## 5 Afternoon Ride 2021-08-31 2021    August   35         243           31 Tue
## 6   Morning Ride 2021-08-29 2021    August   35         241           29 Sun
```

2.2 My year in summary

How are we tracking this year? And compared to other Years/Months? We can see our cumulative activities in each months, make sure we're not slacking or craming activities at the end of each month.

```
act_sub %>%
  select(Year, Month, start_date, day_of_month, type, distance) %>%
  filter(type %in% c("Hike", "Run", "Ride")) %>%
  arrange(type, start_date) %>%
  group_by(Year, Month, type) %>%
  mutate(cumulative_distance = cumsum(distance)) %>%

  ggplot(aes(x = day_of_month, y = cumulative_distance, col = as.factor(Month))) +
    geom_line() + geom_point(size = 1) +
    scale_color_brewer(palette = "Set1") +
    facet_wrap(~type, scales = "free_y") +
    labs(title = "Cumulative distance per month",
         subtitle = "Last 4 months",
         y = "Distance (km)",
         x = "Day of the Month") +
    theme_pander() +
    theme(panel.border = element_rect(size = .5, color = "black"),
          legend.title = element_blank())
```



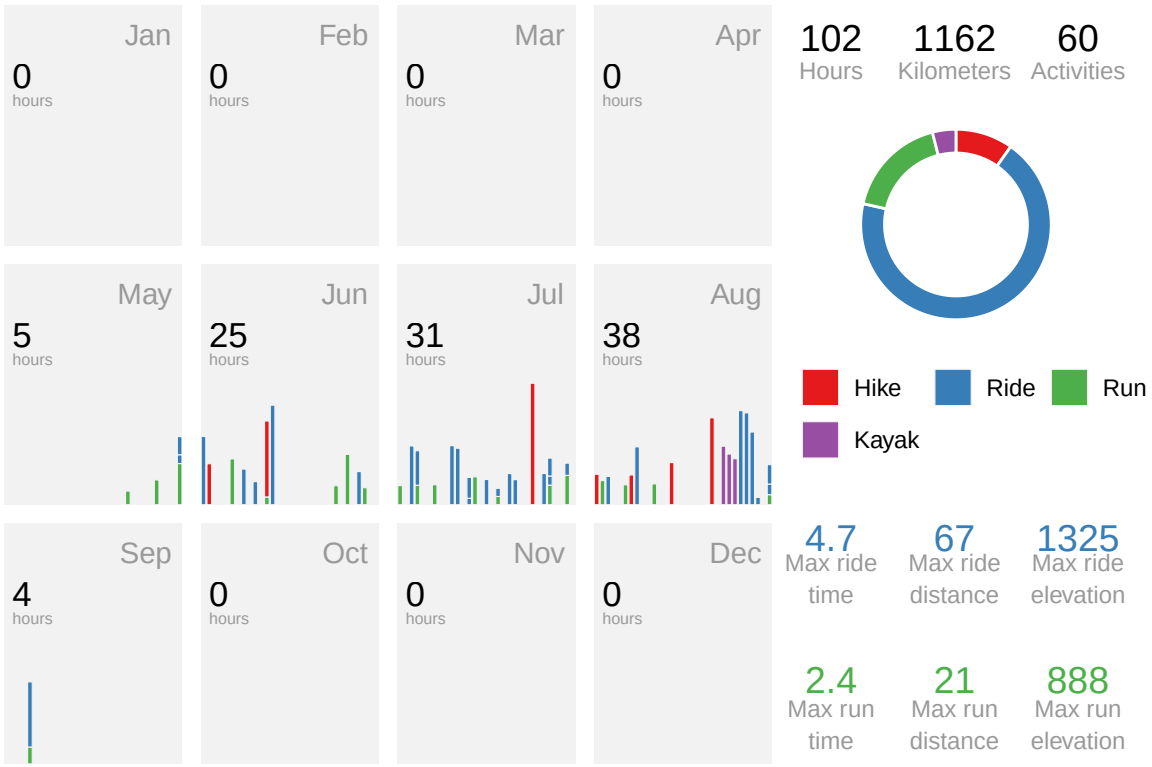
2.3 Activity calendar

A popular way to visualise Strava activities is through an activity calendar. We can plot all our activities throughout the year and summarise all our activity in one panel. There are a few steps to this. First we create a calendar, and one of the tricks to plot a monthly calendar, is of course identifying the day of the week, the week of the month and the month of the year. Lubridate has functiond for most of this but here I created the 'wom' function to identify the week of the month, which will be the rows in each month on the calendar.

```
#Week of the month.
wom <- function(date) {
    #Return the week day of the first day of the month (1-7),
    #with Monday as the first day of the week.
    first <- wday(as.Date(paste(year(date), month(date),1,sep="-")),
                  week_start = getOption("lubridate.week.start", 1))
    #Return the week of the month it belongs to (1-5). %/% is the integral division.
    return((mday(date)+(first-2)) %/% 7+1)
}

#Create the annual calendar
calendar = data.frame(Date = seq(ymd("2021-01-01"), ymd("2021-12-31"), by = "days")) %>%
  mutate(Month = month(Date, label = T, abbr = T),
         day = wday(Date, label = TRUE, week_start = getOption("lubridate.week.start", 1)),
         day_of_month = mday(Date),
         wom = wom(Date)) %>%
  #Add Strava activities
  left_join(act_sub %>%
              group_by(Date, type) %>%
              mutate(cumulative_distance = cumsum(distance),
                     cumulative_time = cumsum(moving_time)),
            by = c("Date", "day", "day_of_month"))

#Get a summary of total active hours each month
monthly_time = calendar %>% ungroup() %>% group_by(Month.x) %>%
```

```r
  summarise(total.time = round(sum(moving_time, na.rm = T)/60, 0))

#1. THE CALENDAR
p1 = ggplot(calendar, aes(x = day_of_month, y = cumulative_time)) +
  geom_col(aes(fill = type), col = "grey95", size = .2, width = .8) +
  geom_text(data = monthly_time, hjust = 0, size = 4.5,
            aes(label = total.time, x = 2,
                y = max(calendar$cumulative_time, na.rm = T)*1.4)) +
   annotate("text", label = "hours", x = 2,
            y = max(calendar$cumulative_time, na.rm = T)*1.2,
            hjust = 0, size = 2, colour = "grey60") +
  scale_x_continuous(expand = c(0,0)) +
  scale_y_continuous(limits = c(0, max(calendar$cumulative_time, na.rm = T)*1.5),
                     expand = c(0,0)) +
  scale_fill_brewer(palette = "Set1") +
  facet_wrap(~Month.x) +
  labs(x = "", y = "") +
  theme_void() +
  theme(strip.text.x = element_text(hjust = .92, size = 11,
                                    margin = margin(.2,0,.2,0, "cm"),
                                    colour = "grey60"),
        panel.background = element_rect(fill = "grey95", colour = "grey95"),
        strip.background = element_rect(fill = "grey95", colour = "grey95"),
        plot.margin = unit(c(1, 1, 1, 1), "lines"),
        panel.spacing = unit(.5, "lines"),
        legend.key.size = unit(.5, 'cm'),
        legend.direction = "horizontal",
        legend.position = "right",
        legend.title.align = 1,
        legend.title = element_blank()) +
  guides(fill=guide_legend(nrow=2, ncol = 3, byrow=TRUE))

#2. TOTAL SUMMARY
totals = act_sub %>% ungroup() %>%
  summarise(Hours = round(sum(moving_time, na.rm = T)/60, 0),
            Kilometers = round(sum(distance, na.rm = T), 0),
            Activities = n()) %>%
  gather()

p2 = ggplot() +
  geom_text(data = totals, aes(label = value, x = c(1,4,7), y = 1.3), size = 5) +
  geom_text(data = totals, aes(label = key, x = c(1,4,7), y = 1),
            size = 3, colour = "grey60") +
  theme_void() + theme(plot.margin = unit(c(1, 1, 1, 1), "lines")) +
  scale_y_continuous(limits = c(-1,1.3))+
  scale_x_continuous(limits = c(0,20))


#3. THE DISTANCE DONUT
donut = act_sub %>% group_by(type) %>%
  summarise(value = sum(distance)) %>%
  mutate(fraction = value / sum(value),
        ymax = cumsum(fraction),
        ymin = c(0, head(ymax, n=-1)))

#https://www.r-graph-gallery.com/128-ring-or-donut-plot.html
p3 = ggplot(donut, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=type)) +
  geom_rect(col = "white") +
  coord_polar(theta="y") +
  scale_fill_brewer(palette = "Set1") +
```

```
  xlim(c(0, 4)) +
  theme_void() +
  theme(legend.position = "none")

#4. MAX RUNS/RIDES
max = act_sub %>% ungroup() %>%
  group_by(type) %>%
  summarise(`Max time` = round(max(moving_time, na.rm = T)/60, 1),
            `Max distance` = round(max(distance, na.rm = T), 0),
            `Max elevation` = round(max(total_elevation_gain))) %>%
  gather(key = key, value = value, - type)

type.colors = brewer.pal(3, "Set1")

p4 = ggplot() +
  geom_text(data = max %>% filter(type == "Ride"),
            aes(label = value, x = c(1,4,7), y = 1.3), size = 5, col = type.colors[2]) +
  geom_text(data = max %>% filter(type == "Ride"),
            aes(label = c("Max ride\ntime", "Max ride\ndistance", "Max ride\nelevation"),
                x = c(1,4,7), y = 1), size = 3, colour = "grey60") +
  theme_void() + theme(plot.margin = unit(c(1, 1, 1, 1), "lines")) +
  scale_y_continuous(limits = c(-1,1.3))+
  scale_x_continuous(limits = c(0,20))

p5 = ggplot() +
  geom_text(data = max %>% filter(type == "Run"),
            aes(label = value, x = c(1,4,7), y = 1.3), size = 5, col = type.colors[3]) +
  geom_text(data = max %>% filter(type == "Run"),
            aes(label = c("Max run\ntime", "Max run\ndistance", "Max run\nelevation"),
                x = c(1,4,7), y = 1), size = 3, colour = "grey60") +
  theme_void() + theme(plot.margin = unit(c(1, 1, 1, 1), "lines")) +
  scale_y_continuous(limits = c(-1,1.3))+
  scale_x_continuous(limits = c(0,20))


#COMBINE IN A SUMMARY FIGURE
p1 + inset_element(p2, .92,.6,2.2,1.11) +
  inset_element(p3, .78,.6,1.7,.94) +
  inset_element(p4, .92,-.2,2.2,.4) +
  inset_element(p5, .92,-.45,2.2,0.2)
```

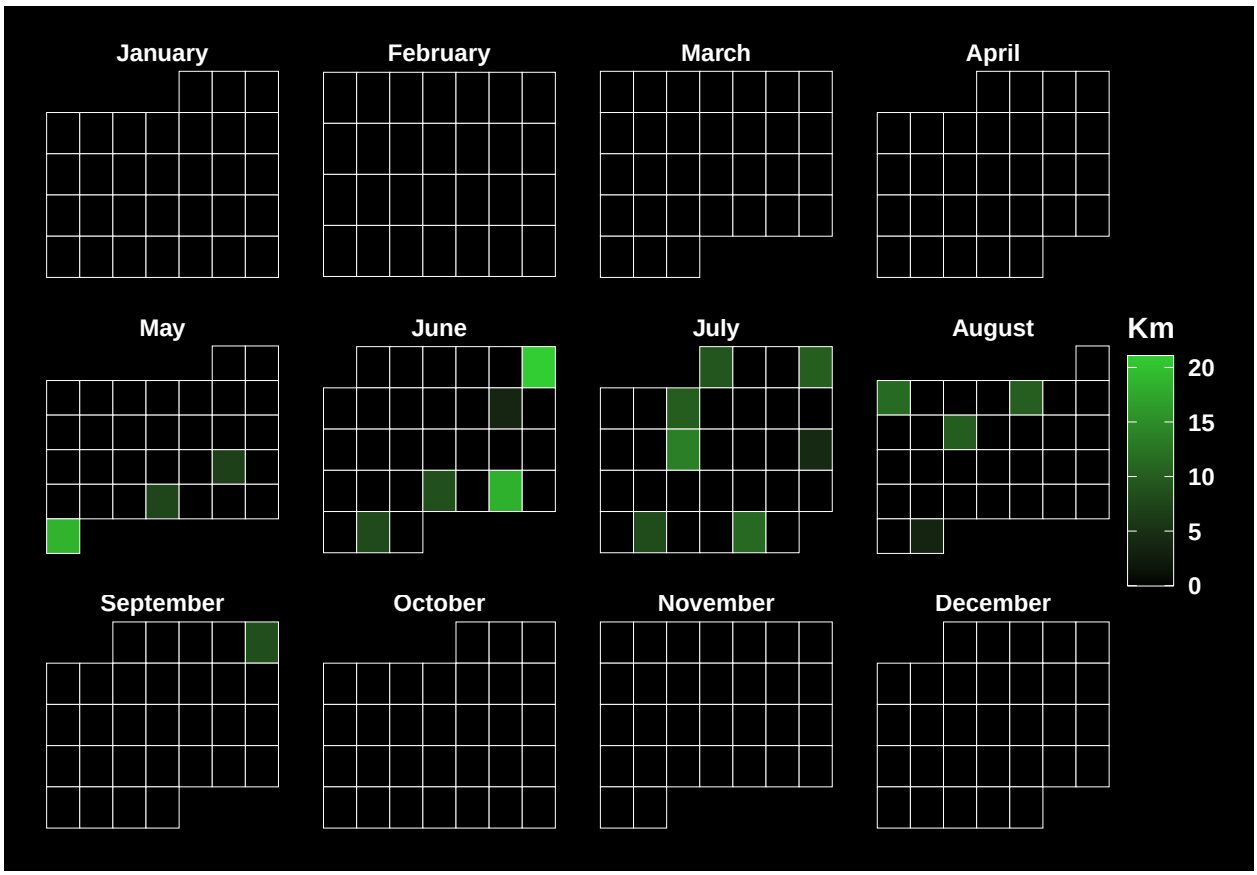



```
#ggsave("Activity calendar.jpeg")
```

2.4 Running calendar

I'm mostly interesting in my running performance. We can use the same approach to make a running calendar.

```
calendar = data.frame(Date = seq(ymd("2021-01-01"), ymd("2021-12-31"), by = "days")) %>%
  mutate(Month = month(Date, label = T, abbr = F),
         day = wday(Date, label = TRUE, week_start = getOption("lubridate.week.start", 1)),
         wom = wom(Date)) %>%
  left_join(act_sub %>% filter(type == "Run") %>%
              group_by(Date, type) %>%
              mutate(cumulative_distance = cumsum(distance)),
            by = c("Date", "Month", "day"))


ggplot(calendar, aes(x = day, y = reorder(wom, - wom))) +
  geom_tile(aes(fill = cumulative_distance), col = "white") +
  scale_fill_continuous(low = "black", high = "limegreen", na.value="black", lim = c(0, NA),
                        guide = guide_colourbar(title = "Km", frame.colour = "white")) +
  facet_wrap(~Month, scales = "free") +
  labs(x = "", y = "") +
  theme_void() +
  theme(panel.background = element_rect(fill = "black"),
        plot.background = element_rect(fill = "black"),
        text = element_text(colour = "white", face = "bold"),
        strip.text = element_text(vjust = 1.25),
        plot.margin = unit(c(1, 1, 1, 1), "lines"),
        panel.spacing = unit(1, "lines"))
```



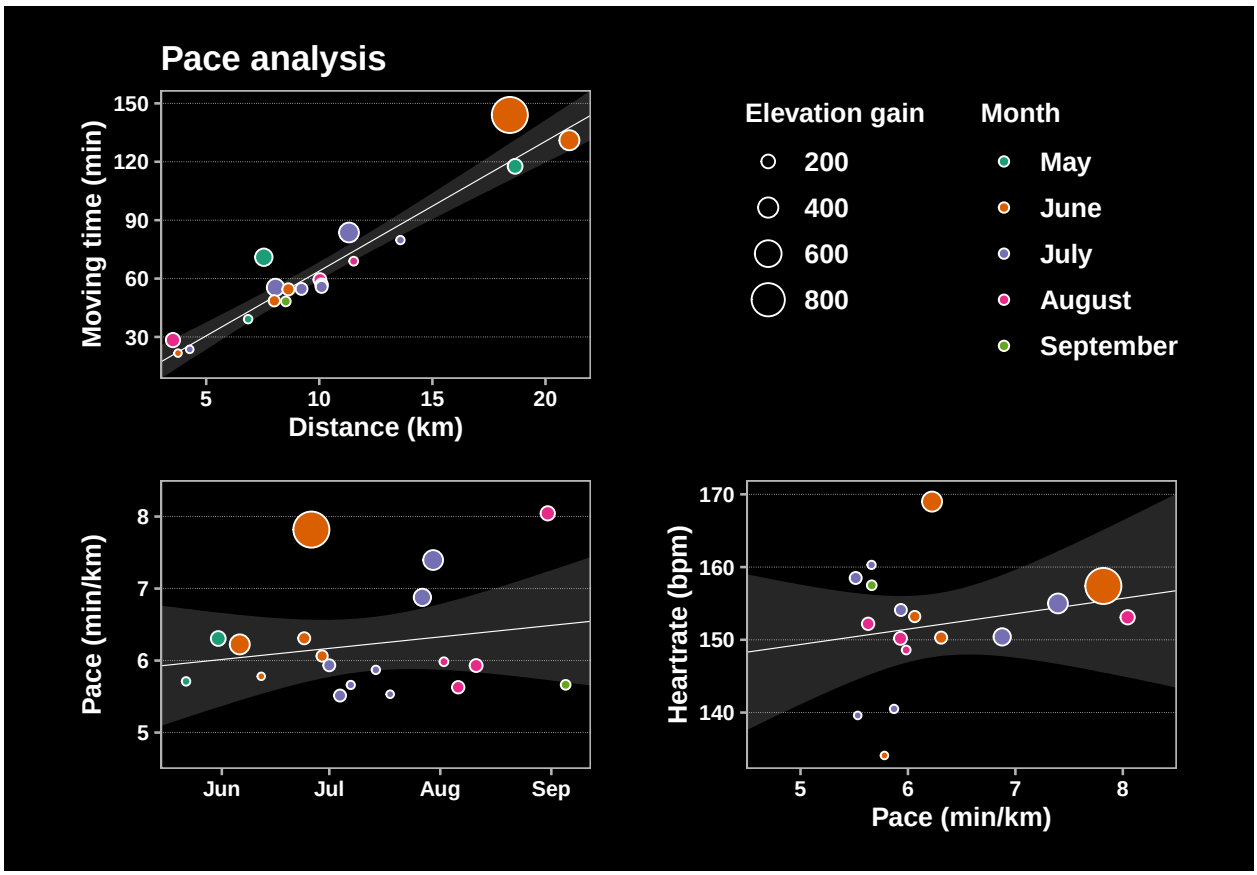

```
#ggsave("Running calendar.jpeg", width = 15, height = 10, units = "cm")
```

2.5 Pace analysis

How's my running performance tracking? We can look at pace, hearrate and overall performance of this period.

```
Run = act_sub %>% filter(type == "Run")

#moving time vs distance with elevation
Pace1 = ggplot(Run, aes(y = moving_time,
                        x = distance,
                        size = total_elevation_gain)) +
  stat_smooth(method = "lm", col = "white", fill = "grey",
              alpha = .2, size = .2, fullrange = TRUE) +
  geom_point(aes(fill = factor(Month)), shape = 21, col = "white", alpha = 1) +
  scale_radius(name = "Elevation gain") +
  scale_fill_brewer(type = "qual", palette = "Dark2", name = "Month") +
  scale_y_continuous(name = "Moving time (min)", breaks = c(30,60,90,120,150),
                     expand = c(0,0)) +
  scale_x_continuous(name = "Distance (km)", limits = c(3,22), expand = c(0,0)) +
  ggtitle("Pace analysis") +
  theme_calc()

#Pace through time with elevation
Pace2 = ggplot(Run, aes(y = moving_time / distance,
                        x = Date,
                        size = total_elevation_gain)) +
  stat_smooth(method = "lm", col = "white", fill = "grey",
              alpha = .2, size = .2, fullrange = TRUE) +
  geom_point(aes(fill = factor(Month)), shape = 21, col = "white", alpha = 1) +
  scale_radius(name = "Elevation gain") +
  scale_fill_brewer(type = "qual", palette = "Dark2", name = "Month") +
  scale_y_continuous(name = "Pace (min/km)", breaks = c(5,6,7,8), limits = c(4.5,8.5),
```

```
                    expand = c(0,0)) +
  scale_x_date(name = "", expand = c(0,0), limits = c(min(Run$Date) - 7, max(Run$Date+7))) +
  theme_calc()

#Pace vs heartrate... am I struggling?
Pace3 = ggplot(Run, aes(x = moving_time / distance,
                        y = average_heartrate,
                        size = total_elevation_gain)) +
  stat_smooth(method = "lm", col = "white", fill = "grey",
              alpha = .2, size = .2, fullrange = TRUE) +
  geom_point(aes(fill = factor(Month)), shape = 21, col = "white", alpha = 1) +
  scale_radius(name = "Elevation gain") +
  scale_fill_brewer(type = "qual", palette = "Dark2", name = "Month") +
  scale_y_continuous(name = "Heartrate (bpm)") +
  scale_x_continuous(name = "Pace (min/km)", breaks = c(5,6,7,8), limits = c(4.5,8.5),
                     expand = c(0,0)) +
  theme_calc()

#Collect all plots
pace_analysis = Pace1 + guide_area() + Pace2 + Pace3 +
  plot_layout(guides = 'collect')  &
  theme(legend.direction = "vertical",
        legend.position = "right",
        legend.box = "horizontal",
        panel.background = element_rect(fill = "black"),
        panel.grid.major.y = element_line(colour = "white",linetype = "dotted", size = .1),
        plot.margin = unit(c(0.5, 1, 0.5, 1), "lines"),
        plot.background = element_rect(color = "black", fill = "black"),
        text = element_text(colour = "white", face = "bold"),
        legend.background = element_rect(fill = "transparent"),
        legend.key = element_rect(fill = "transparent"))

pace_analysis
```



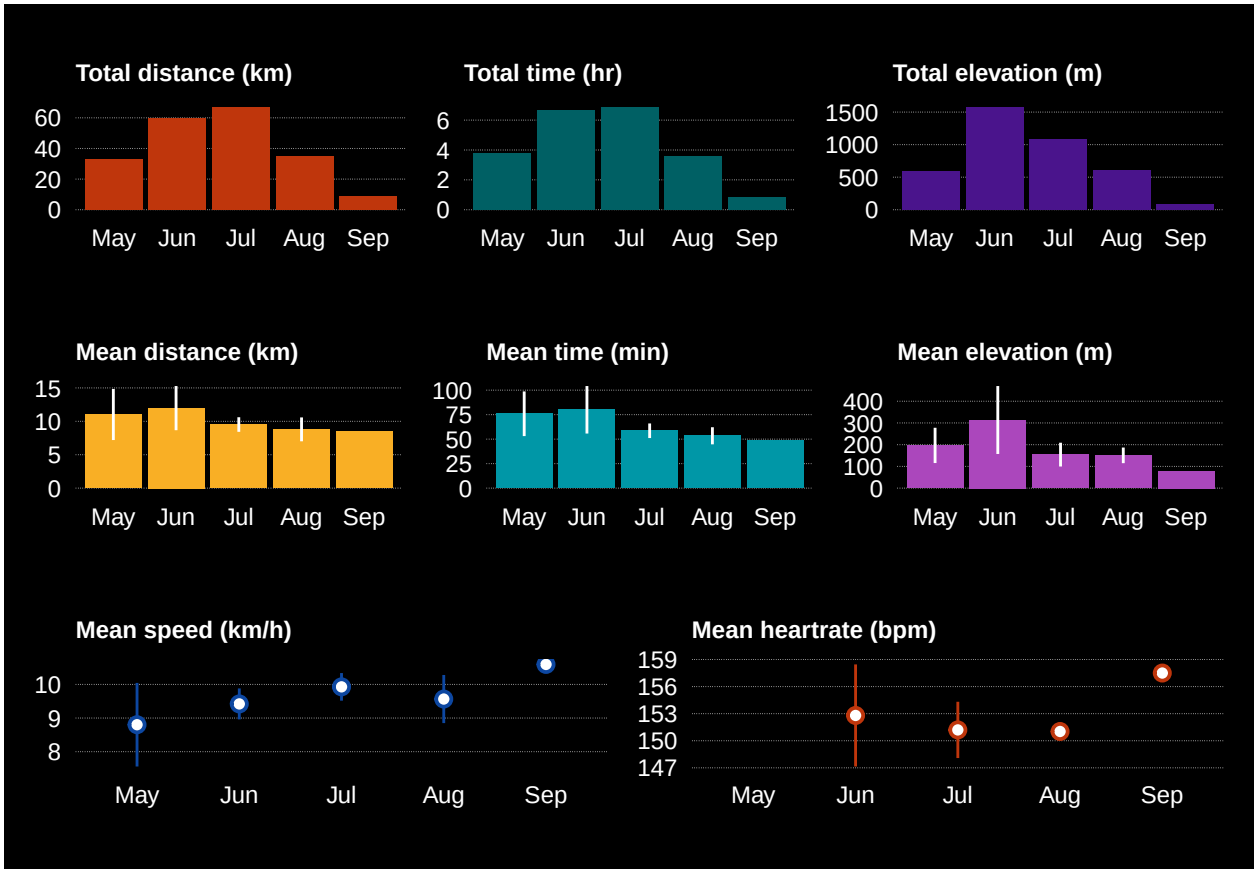

```
#ggsave("Pace analysis.jpeg", width = 16, height = 15, units = "cm")
```

2.6 Run summary and overview

```
Run.summary = Run %>%
  mutate(Month = month(Date, label = T, abbr = T)) %>%
  select(distance, moving_time, total_elevation_gain,
         average_heartrate, average_speed, Year, Month) %>%
  group_by(Year, Month) %>%
  summarise_all(list(n = ~ n(),
                     mean = ~mean(., na.rm = T),
                     sum = ~sum(., na.rm = T),
                     se = ~sd(., na.rm = T) / sqrt(n()))) %>%
#  select(- c(total_elevation_gain_n, average_heartrate_n,
#               moving_time_n, average_speed_n,
#             average_heartrate_sum,average_speed_sum)) %>%
  pivot_longer(cols = c(-Year, -Month),
               names_to = c("variable", ".value"),
               names_pattern = "(.+)_(.+)")

my_theme = theme_clean() +
  theme(plot.title = element_text(size = 9, face = "bold"),
        axis.ticks = element_blank(),
        axis.line.x = element_blank(),
        axis.line.y = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        panel.border = element_blank(),
        plot.background = element_blank(),
        plot.margin = margin(.6,.2,.6,.2, "cm"))

#Distance
```

```r
p1 = ggplot(Run.summary %>% filter(variable == "distance"),
            aes(y = mean, x = Month)) +
  geom_col(fill = "#F9AF25") + labs(title = "Mean distance (km)") +
  geom_linerange(aes(ymin = mean - se, ymax = mean + se), col = "white") +
  my_theme

p2 = ggplot(Run.summary %>% filter(variable == "distance"),
            aes(y = sum, x = Month)) +
  geom_col(fill = "#BF360C") + labs(title = "Total distance (km)") +
  my_theme

#Time
p3 = ggplot(Run.summary %>% filter(variable == "moving_time"),
            aes(y = mean, x = Month)) +
  geom_col(fill = "#0097A7") + labs(title = "Mean time (min)") +
  geom_linerange(aes(ymin = mean - se, ymax = mean + se), col = "white") +
  my_theme

p4 = ggplot(Run.summary %>% filter(variable == "moving_time"),
            aes(y = sum / 60 , x = Month)) +
  geom_col(fill = "#006064") + labs(title = "Total time (hr)") +
  my_theme

#Elevation
p5 = ggplot(Run.summary %>% filter(variable == "total_elevation_gain"),
            aes(y = mean, x = Month)) +
  geom_col(fill = "#AB47BC") + labs(title = "Mean elevation (m)") +
  geom_linerange(aes(ymin = mean - se, ymax = mean + se), col = "white") +
  my_theme

p6 = ggplot(Run.summary %>% filter(variable == "total_elevation_gain"),
            aes(y = sum, x = Month)) +
  geom_col(fill = "#4A148C") + labs(title = "Total elevation (m)") +
  my_theme

#Speed
p7 = ggplot(Run.summary %>% filter(variable == "average_speed"),
            aes(y = mean, x = Month)) +
  labs(title = "Mean speed (km/h)") +
  geom_pointrange(aes(ymin = mean - se, ymax = mean + se),
                  shape = 21, fill = "white", col = "#0D47A1") +
  my_theme

#heartrate
p8 = ggplot(Run.summary %>% filter(variable == "average_heartrate"),
            aes(y = mean, x = Month)) +
  labs(title = "Mean heartrate (bpm)") +
  geom_pointrange(aes(ymin = mean - se, ymax = mean + se),
                  shape = 21, fill = "white", col = "#BF360C") +
  my_theme


#Combine all plots
(p2 + p4 + p6) /
(p1 + p3 + p5) /
  (p7 + p8) &
    theme(panel.background = element_rect(fill = "black"),
        panel.grid.major.y = element_line(colour = "white", linetype = "dotted", size = .1),
        plot.background = element_rect(color = "black", fill = "black"),
        text = element_text(colour = "white"),
```

```
        axis.text = element_text(colour = "white"))
```




```
#ggsave("~/Desktop/myStrava2.jpeg", width = 15, height = 15, units = "cm")
```

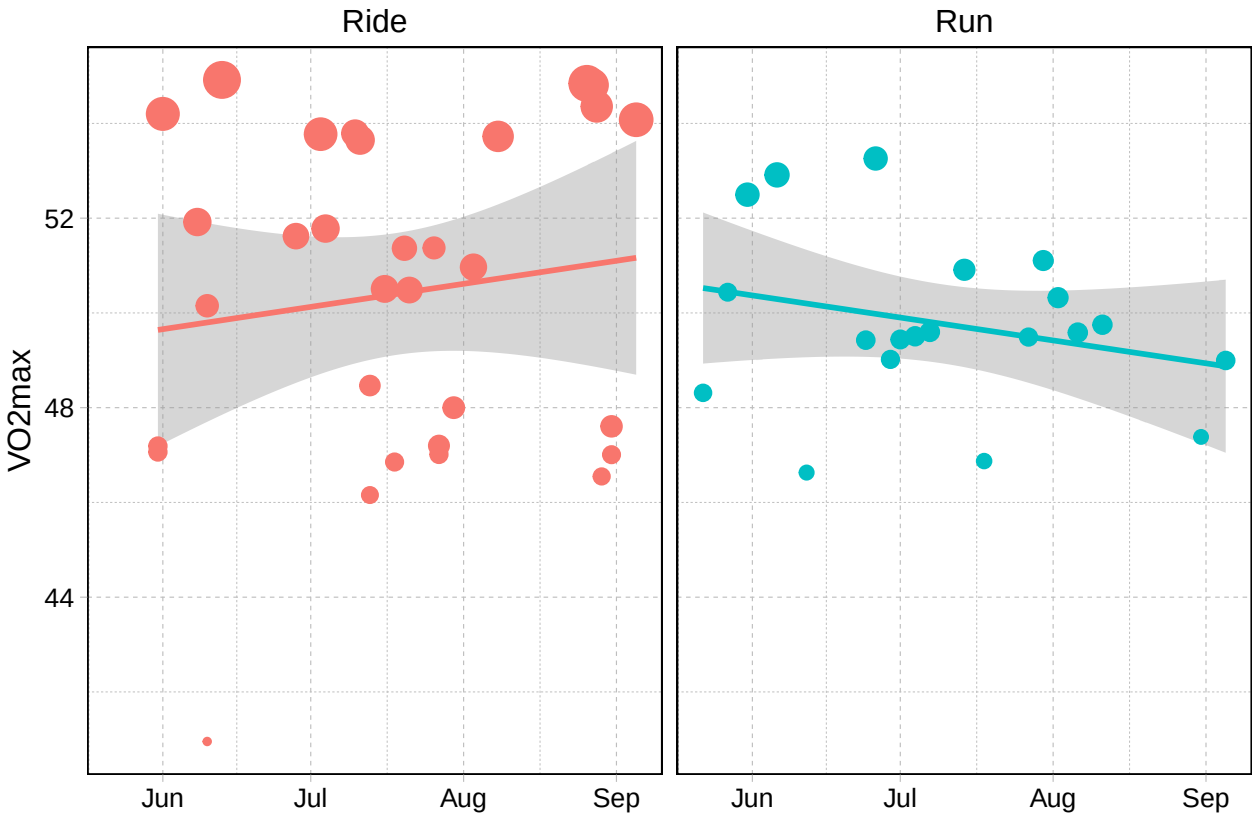
2.7 VO2 max

VO2 max can be estimated from your pace and heartrate. It's crude but it's a good indication of how much effort goes in to each activity. This is a work in progress. Now we can look at each activity individually and measure changes in pace and elevation throughout the activity using the get_activity_streams() function.

```
max_bpm = max(Run$max_heartrate, na.rm = T)
resting_bpm = 48 #Apple Watch (June 2021)
reserve_bpm = max_bpm - resting_bpm
```

Is my VO2 max improving? Work in progress.

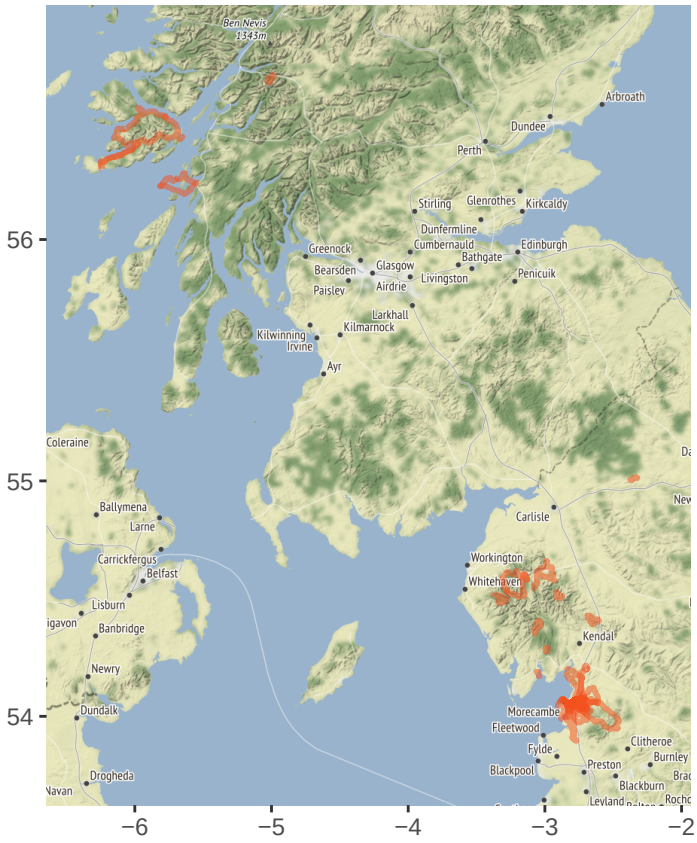
```
act_data %>% filter(type %in% c("Ride", "Run")) %>%
  mutate(average_heartrate = as.numeric(average_heartrate),
    p.reserve = (average_heartrate - resting_bpm) / reserve_bpm,
    mtr.min = mean(average_speed, na.rm = T) * 1000 / 60, #metres per minute
         p.max = 0.8 + 0.1894393 * exp(-0.012778 * moving_time/60) +
                       0.2989558 * exp(-0.1932605 * moving_time/60),
         VO2 = -4.60 + 0.182258 * mtr.min + 0.000104 * mtr.min^2, #Oxygen cost
         VO2max = VO2 / p.max) %>%
  mutate(date = as.Date(start_date)) %>%

  ggplot(aes(y = VO2max, x = date, col = type)) +
    geom_smooth(method = "lm") +
    geom_point(aes(size = distance)) +
    facet_wrap(~type) +
    scale_x_date(breaks = "1 month", date_labels =  "%b", name = "") +
    theme_pander() +
    theme(panel.border = element_rect(size = .5, color = "black"),
        legend.position = "none")
```



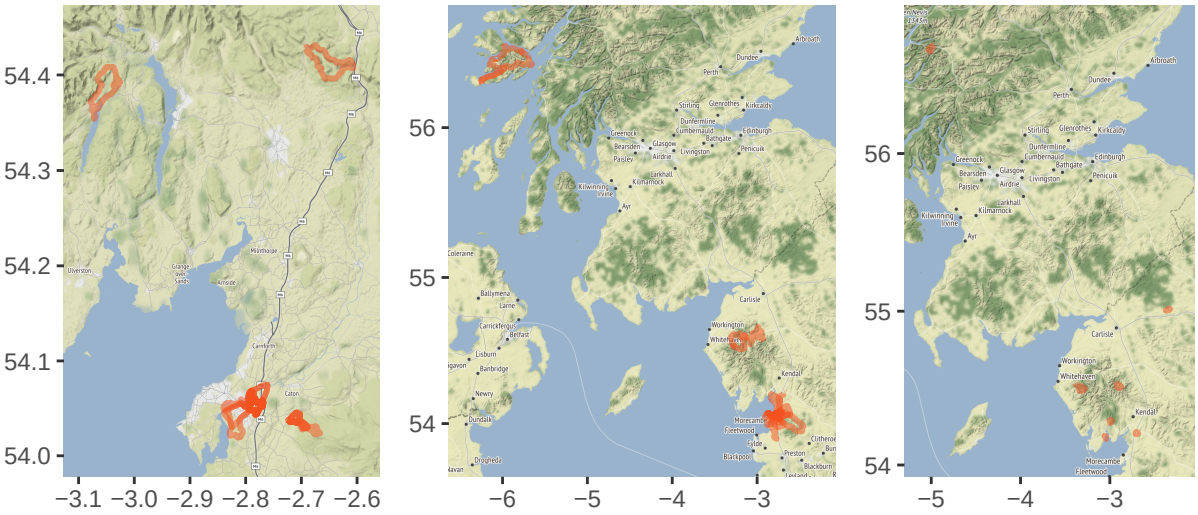
3 Mapping Strava activities

3.1 Map all your activities in Google maps

```
get_heat_map(act_data = act_data, key = mykey,
             col = '#F4511E', size = 1, distlab = F, f = 0.1, expand = 1)
```

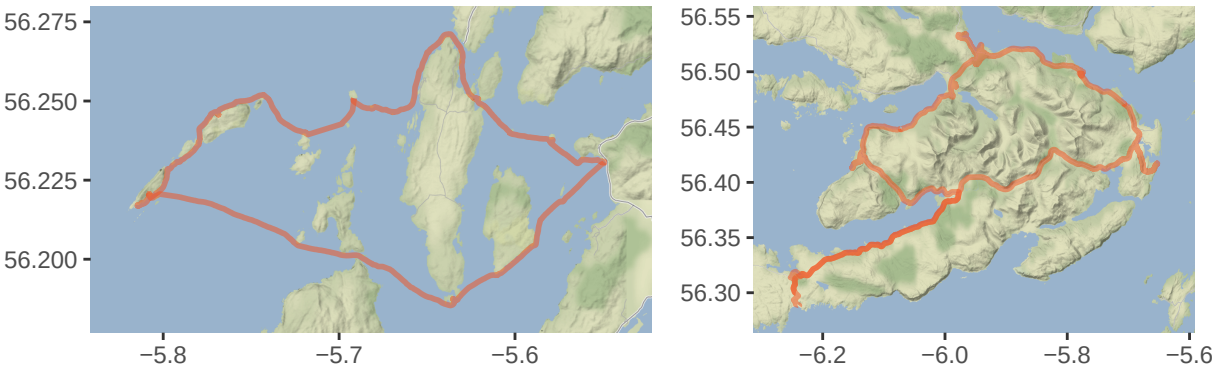


```
runs = get_heat_map(act_data = act_data %>% filter(type == "Run"),
                    key = mykey,  col = '#F4511E', size = 1,
                    distlab = F, f = 0.1, expand = 1)

rides = get_heat_map(act_data = act_data %>% filter(type == "Ride"),
                     key = mykey,  col = '#F4511E', size = 1,
                     distlab = F, f = 0.1, expand = 1)

hikes = get_heat_map(act_data = act_data %>% filter(type == "Hike"),
                     key = mykey,  col = '#F4511E', size = 1,
                     distlab = F, f = 0.1, expand = 1)

runs + rides + hikes
```



3.2 Map group of activities

```
#Scotland holidays 2021
kayak = get_heat_map(act_data = act_data %>% filter(type == "Kayaking"),
                     key = mykey,  col = '#F4511E', size = 1,
                     distlab = F, f = 0.1, expand = 1)

Mull = get_heat_map(
  act_data = act_data %>%
    filter(type == "Ride" & start_latitude >56 & start_longitude < -5.5),
                     key = mykey,  col = '#F4511E', size = 1,
                     distlab = F, f = 0.1, expand = 1)

kayak + Mull
```



3.3 Map specific activities

```
# actitivy id
id <- 5569683844

#Plotting elevation and grade for a single ride:
p1 = get_heat_map(my_acts, id = id, alpha = 1, add_elev = T, f = 0.3, distlab = F, key = mykey, size =
# plot % gradient along a single ride
p2 = get_heat_map(my_acts, id = id, alpha = 1, add_elev = T, f = 0.3,  distlab = F, as_grad = T, key =
p3 = get_elev_prof(my_acts, id = id, key = mykey, units = 'metric')
p4 = plot_spdsplits(my_acts, stoken, id = id, units = 'metric')
p1 + (p3 / p4)
```

#END