Отчёт по заданию «Анализ траектории»

Выполнено: Рыжковой А.Ю.

R.Script – 1.0 (предоперации)

```
# ---------------- Предоперации ------------------
library(sf)
library(ggplot2)
library(adehabitatHR)
library(mapview)
library(dplyr)
library(leaflet)
library(plotly)
library(units) #Работа с физическими единицами измерения
library(geosphere) #Альтернатива расчета расстояния и скорости
library(htmlwidgets) #Сохранение как HTML-файл (интерактивная карта)
library(webshot) #Создание статичных png- и jpeg-изображений через
HTML (для презентации)
webshot::install_phantomjs()

# Чтение GPX как пространственного объекта
dir <- "C:/GIS/3 homework"
st_layers(file.path(dir, "2022-01-09_14-00_Sun.gpx"))
ski <- st_read(file.path(dir, "2022-01-09_14-00_Sun.gpx"), layer =
"track_points")
ski_track <- st_read(file.path(dir, "2022-01-09_14-00_Sun.gpx"), layer
= "tracks")

# Проверяем структуру данных
print(st_geometry_type(ski))  # Тип геометрии (POINT)
print(colnames(ski))          # Показывает доступные столбцы
sf::st_crs(ski)$epsg

print(st_geometry_type(ski_track))  # Тип геометрии (MULTILINESTRING),
                                   #может пригодится для QGIS, сейчас
будет
                                   #использован тип POINT
print(colnames(ski_track))          # Показывает доступные столбцы
sf::st_crs(ski_track)$epsg
```

```
# Проверяем часовой пояс
attr(ski_track$time, "tzone") #не задано, значит время локальное
# Преобразуем UTC в московское время (UTC+3)
ski$time <- as.POSIXct(ski$time, tz = "UTC")  # Сначала явно
указываем UTC
attr(ski$time, "tzone") <- "Europe/Moscow"    # Затем меняем часовой
пояс,
#поэтому неправильно интерпретировалось у ребят (в Москве UTC+3)
#принудительно ставим Москву, чтобы не запутаться на картах в
дальнейшем
head(ski$time)

# Проверяем текущую CRS (должна быть WGS84, EPSG:4326)
st_crs(ski)
st_crs(ski_track)
head(st_coordinates(ski))
head(st_coordinates(ski_track))

# Трансформируем в EPSG:32637 (UTM Zone 37N)
ski_37N <- st_transform(ski, 32637)
ski_track_37N <- st_transform(ski_track, 32637)

# Проверяем результат
st_crs(ski_37N)
sf::st_crs(ski_37N)$epsg
head(st_coordinates(ski_37N)) # Координаты теперь в метрах
```

R.Script – 1.1

```r
# Преобразование в обычную таблицу и извлечение координат + сокращение столбцов
ski_table <- ski_37N %>%
 mutate(
   lon = st_coordinates(.)[,1],
   lat = st_coordinates(.)[,2]
) %>%
 as.data.frame() %>%
 select(track_seg_point_id, ele, time, hdop, lon, lat)

# Для линейного трека (ski_track) — извлекаем координаты (дополнительные
# операции, если первое недоступно, отмечаю - #)
#ski_track_coords <- st_coordinates(ski_track) %>%
 #as.data.frame() %>%
 #rename(lon = X, lat = Y, line_id = L1)

# Просмотр структуры
View(ski_table)
head(ski_table)
nrow(ski_table) #835 строк
print(ski_table$time[1]) #записываем начало движения - 14:00:18
tail(ski_table, n = 10)
print(ski_table$time[835]) #записываем конец движения - 15:16:08

# Удаляем полные дубликаты точек (дубли всех стобцов)
nrow_original <- nrow(ski_table)
ski_table <- ski_table %>%
  distinct(lon, ele, lat, hdop, .keep_all = TRUE)
nrow(ski_table) #829 строк
removed <- nrow_original - nrow(ski_table)
message("Удалено дубликатов: ", removed)

# Экспорт в CSV (для просмотра данных отдельно)
write.csv(ski_table, "C:/GIS/3 homework/ski_track.csv", row.names = FALSE, fileEncoding = "UTF-8")
```

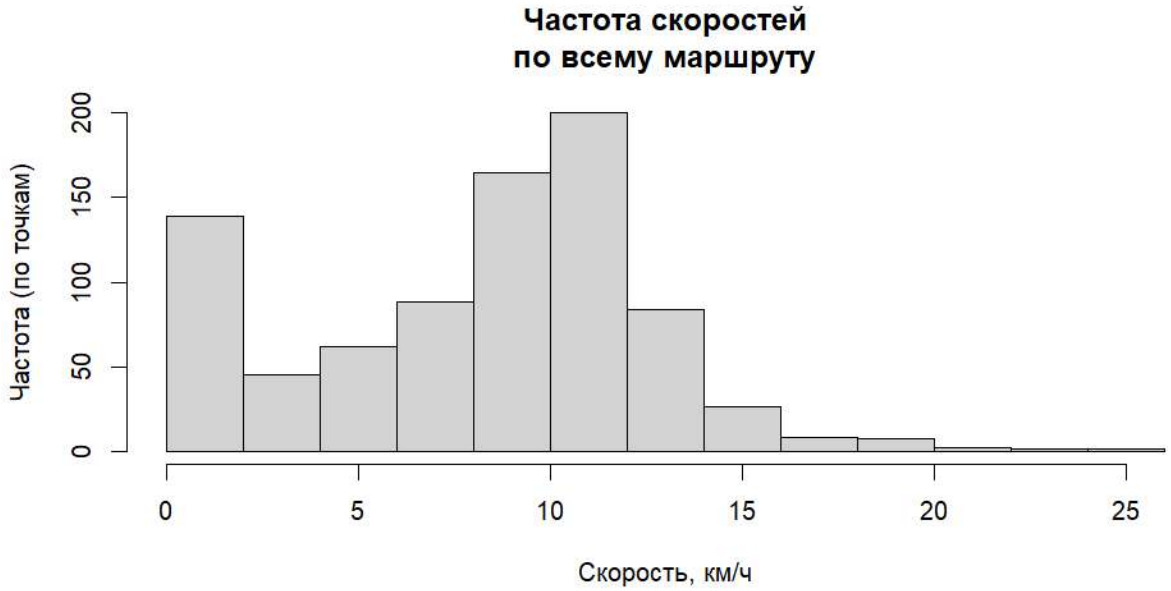
```r
#write.csv(ski_track_coords, "C:/GIS/3 homework/ski_track_line.csv", row.names = FALSE, fileEncoding = "UTF-8")
getwd() #просмотр рабочей директории R
file.exists("ski_track.csv") #файл существует?
#file.exists("ski_track_line.csv") #файл существует?
# Чтение файла (если что-то меняли в csv)
#ski_data <- read.csv(file.path(dir, "ski_track.csv"), encoding = "UTF-8",
                     #row.names = 1)
#head(ski_data)
#ski_data1 <- read.csv(file.path(dir, "ski_track_line.csv"), encoding = "UTF-8",
                     #row.names = NULL)
#head(ski_data1)

# Быстрая первоначальная визуализация
#leaflet не работает ни с чем, кроме 4326, пришлось помучаться
#используем первоначальныt ski и ski_track для визуализации, а ski_37N/ski_table и ski_track_37N для расчётов

# Проверяем, совпадает ли система координат (CRS) с EPSG:4326 (WGS84))
if(st_crs(ski) != st_crs(4326)) {
 # Если нет - преобразуем оба объекта в WGS84
 ski <- st_transform(ski, 4326)  # Преобразуем основной объект
 ski_track <- st_transform(ski_track, 4326)  # Преобразуем трек
}

# Рассчитываем общее пройденное расстояние
total_distance <- sum(as.numeric(st_length(ski_track)))

# Выводим общее расстояние
print(paste("Общее пройденное расстояние:", total_distance, "метров")) #9719 метров
```

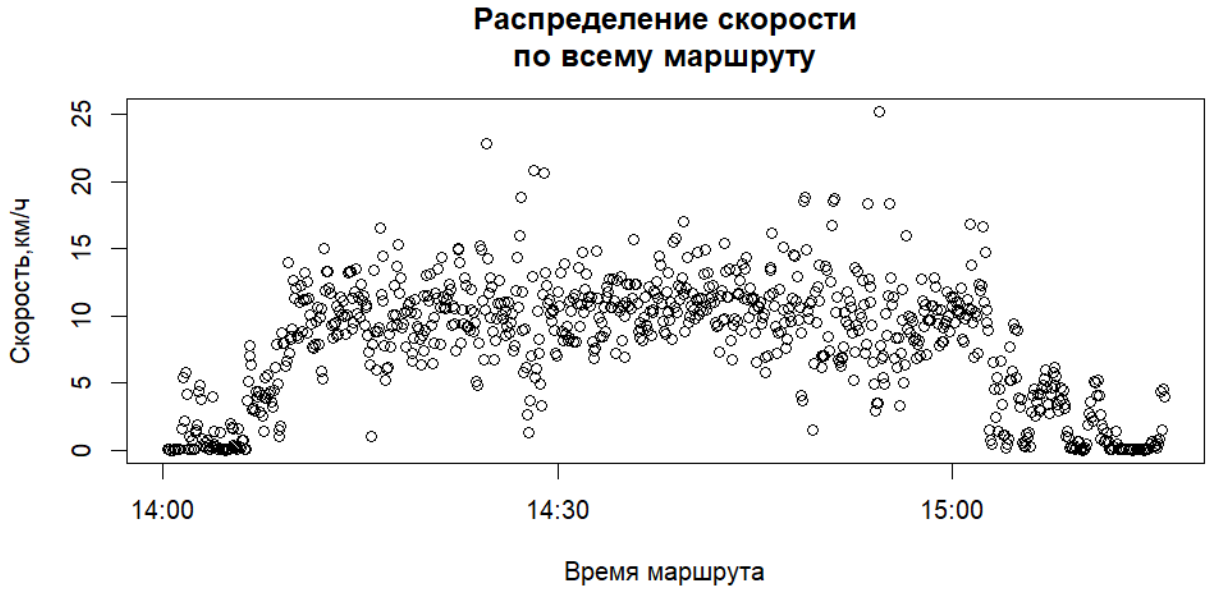
Визуализация в R – 3.0

```
# Создаем первоначальную карту с основными параметрами в точках
map01 <- leaflet() %>%
 addTiles() %>%
 addPolylines(
   data = st_coordinates(ski_track) %>% as.data.frame(),
   lng = ~X,
   lat = ~Y,
   color = "red",
   weight = 3,
   popup = paste("Длина:", round(as.numeric(st_length(ski_track)), 1), "м")
) %>%
     addCircleMarkers(
       data = ski %>% cbind(st_coordinates(ski) %>% as.data.frame()) %>%
                                                  st_drop_geometry(),
       lng = ~X,
       lat = ~Y,
       radius = 3,
       color = "blue",
       popup = ~paste(
         "Время:", format(time, "%H:%M:%S MSK"), "<br>",
         "Высота:", round(ele, 1), "м<br>",
         "HDOP:", hdop
       )
     ) %>%
 addMarkers(
   lng = (ski %>% st_coordinates() %>% as.data.frame())$X[1],
   lat = (ski %>% st_coordinates() %>% as.data.frame())$Y[1],
   popup = "Начало"
) %>%
 addMarkers(
   lng = (ski %>% st_coordinates() %>% as.data.frame())$X[nrow(ski)],
   lat = (ski %>% st_coordinates() %>% as.data.frame())$Y[nrow(ski)],
   popup = "Конец"
) %>%
 addControl(
   position = "topright",
   html = paste0("<div style='background: white; padding: 5px;'>",
                 "<b>Общее расстояние:</b> ",
                 round(total_distance), " м</div>")
)

# Выводим карту
print(map01)

# Создаем HTML и скриншот для презентации
htmlwidgets::saveWidget(map01, file.path(dir, "ski_track_map01.html"), selfcontained = TRUE)
webshot::webshot(file.path(dir, "ski_track_map01.html"), file = file.path(dir,
"ski_track_map01.png"),
                 vwidth = 1000, vheight = 800, delay = 5)
```

R.Script – 2.0 (Анализ)

```
# Расчет расстояния и скорости (в метрической системе)
ski_table <- ski_table %>%
 mutate(
   # Расстояние векторов (dist) в метрах
   dist_v = sqrt((lon - lag(lon))^2 + (lat - lag(lat))^2),
   # Разница высот (delta_h) в метрах
   delta_h = ele - lag(ele),
   # Расстояние (dist) с учётом высоты в метрах
   dist = sqrt(dist_v^2 + delta_h^2),
   # Временной интервал в секундах
   time_diff_sec= as.numeric(difftime(time, lag(time), units =
"secs")),
   # Скорость в м/с
   speed_ms = dist / time_diff_sec,
   # Временной интервал в часах
   time_diff_hour = as.numeric(difftime(time, lag(time), units =
"hours")),
   # Скорость в км/ч
   speed_kmh = (dist/1000) / time_diff_hour,
  )

# Проверяем результат
glimpse(ski_table)
summary(ski_table$dist)  # Статистика по расстояниям
summary(ski_table$speed_ms)   # Статистика по скоростям
summary(ski_table$speed_kmh)
summary(ski_table)

hist(ski_table$speed_kmh, main = "Частота скоростей\nпо всему
маршруту", xlab = "Скорость, км/ч", ylab = "Частота (по точкам)")
plot(ski_table$time, ski_table$speed_kmh, main = "Распределение
скорости\nпо всему маршруту", xlab = "Время маршрута", ylab =
"Скорость,км/ч")
```





R.Script – 2.1 (Общая статистика)

```
> nrow(ski_table)
[1] 829
> print(ski_table$time[1])
[1] "2022-01-09 14:00:18 MSK"
> print(ski_table$time[829])
[1] "2022-01-09 15:16:08 MSK"

> # Рассчитываем общее пройденное расстояние
> total_distance <- sum(as.numeric(st_length(ski_track)))
>
> # Выводим общее расстояние
> print(paste("Общее пройденное расстояние:", total_distance, "метров")) #9719 метров
[1] "Общее пройденное расстояние: 9718.95148594696 метров"

> summary(ski_table$dist)  # Статистика по расстояниям
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
 0.00612  6.77226 13.54127 12.02305 16.63252 41.92391        1
> summary(ski_table$speed_ms)   # Статистика по скоростям
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
0.001113 1.271632 2.516010 2.228896 3.096840 6.987319        1
> summary(ski_table$speed_kmh)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
 0.004008  4.577874  9.057637  8.024025 11.148625 25.154347         1
```

R.Script – 2.2 (Общая статистика)

```
> summary(ski_table)
 track_seg_point_id      ele             time
 Min.   :  0.0      Min.   :119.1   Min.   :2022-01-09 14:00:18.00
 1st Qu.:208.0      1st Qu.:131.3   1st Qu.:2022-01-09 14:19:16.00
 Median :415.0      Median :137.2   Median :2022-01-09 14:37:55.00
 Mean   :415.2      Mean   :135.8   Mean   :2022-01-09 14:37:58.56
 3rd Qu.:622.0      3rd Qu.:139.1   3rd Qu.:2022-01-09 14:56:38.00
 Max.   :834.0      Max.   :156.9   Max.   :2022-01-09 15:16:08.00

      hdop            lon              lat             dist_v
 Min.   : 3.40   Min.   :388550   Min.   :6289495   Min.   : 0.00612
 1st Qu.:13.10   1st Qu.:388727   1st Qu.:6289725   1st Qu.: 6.61526
 Median :28.60   Median :388886   Median :6289831   Median :13.35901
 Mean   :27.41   Mean   :388861   Mean   :6289875   Mean   :11.76066
 3rd Qu.:41.30   3rd Qu.:388994   3rd Qu.:6290053   3rd Qu.:16.40127
 Max.   :50.00   Max.   :389094   Max.   :6290205   Max.   :41.91807
                                                    NA's   :1
    delta_h               dist          time_diff_sec       speed_ms
 Min.   :-16.000000   Min.   : 0.00612   Min.   : 5.000   Min.   :0.001113
 1st Qu.: -0.900000   1st Qu.: 6.77226   1st Qu.: 5.000   1st Qu.:1.271632
 Median :  0.000000   Median :13.54127   Median : 5.000   Median :2.516010
 Mean   : -0.002536   Mean   :12.02305   Mean   : 5.495   Mean   :2.228896
 3rd Qu.:  0.900000   3rd Qu.:16.63252   3rd Qu.: 6.000   3rd Qu.:3.096840
 Max.   : 17.800000   Max.   :41.92391   Max.   :15.000   Max.   :6.987319
 NA's   :1            NA's   :1          NA's   :1        NA's   :1
 time_diff_hour       speed_kmh
 Min.   :0.001389   Min.   : 0.004008
 1st Qu.:0.001389   1st Qu.: 4.577874
 Median :0.001389   Median : 9.057637
 Mean   :0.001526   Mean   : 8.024025
 3rd Qu.:0.001667   3rd Qu.:11.148625
 Max.   :0.004167   Max.   :25.154347
 NA's   :1          NA's   :1
```

R.Script – 2.3

```r
# Ищем точку начала движения на лыжах-----------------------------------------
print(ski_table$speed_kmh[1:110])

# Предлагаю решение в двух вариантах
# 1. Найдем точку резкого возрастания скорости (начало катания), исходя из средней скорости движения
walking_threshold <- mean(ski_table$speed_kmh, na.rm = TRUE) # км/ч
cat("Автоматически рассчитанный порог скорости:", walking_threshold, "км/ч\n")

# Найдем первую точку, где скорость превышает порог и остается высокой
start_skiing_index <- which(ski_table$speed_kmh > walking_threshold)[1]
start_skiing_time <- ski_table$time[start_skiing_index]
start_skiing_point <- ski_table[start_skiing_index, ]
start_skiing_index #точка 94. Запоминаем пока что.
start_skiing_time

# 2. Уточненный анализ с определением порога по устойчивому превышению скорости (без случайных вбросов)
# Определим порог скорости для пешего хода (обычно до 5 км/ч)
# Параметры анализа
min_high_speed_points <- 5  # Минимальное количество точек с высокой скоростью подряд
speed_threshold <- 5        # Порог скорости в км/ч для определения начала катания

# Найдем все последовательности, где скорость превышает порог
high_speed_segments <- rle(ski_table$speed_kmh > speed_threshold)

# Отберем только последовательности, где достаточно точек подряд
valid_segments <- which(high_speed_segments$values & high_speed_segments$lengths >= min_high_speed_points)
# Найдем первый такой сегмент (начало катания)
if (length(valid_segments) > 0) {
 start_pos <- sum(high_speed_segments$lengths[1:(valid_segments[1]-1)]) + 1
 start_skiing_index <- start_pos
 start_skiing_time <- ski_table$time[start_skiing_index]
 start_skiing_point <- ski_table[start_skiing_index, ]

 cat("Начало катания определено в точке:", start_skiing_index,
     "\nВремя:", format(start_skiing_time, "%H:%M:%S"),
     "\nСкорость:", round(ski_table$speed_kmh[start_skiing_index], 1), "км/ч\n")
} else {
 stop("Не найдено подходящих сегментов с высокой скоростью")}
```

```r
# Сравниваем результаты двух методов
cat("Сравнение методов определения точки начала катания:\n")
cat("По среднему значению:", c(start_skiing_index, "точка - "),
format(start_skiing_time, "%H:%M:%S"), "\n")
cat("По устойчивому ускорению:", c(start_skiing_index, "точка - "),
format(start_skiing_time, "%H:%M:%S"))

#Итого: и первый, и второй способ дали одинаковые результаты!
#Начало первого круга - точка 94. Время: 14:09:03.

# Ищем конец кругов :) Лыжник мог ехать просто медленно устав, поэтому без среднего порога # Ищем сразу порог через стабильное снижение скорости
# Уточненный анализ с устойчивым замедлением (без случайных вбросов и с устойчивым замедления движения)
min_low_speed_points <- 5  # Минимальное количество точек с низкой скоростью подряд
speed_threshold <- 5       # Тот же порог скорости

# Найдем все последовательности после начала катания, где скорость ниже порога
low_speed_segments <- rle(ski_table$speed_kmh[start_skiing_index:nrow(ski_table)] < speed_threshold)

# Отберем только последовательности, где достаточно точек подряд
valid_slow_segments <- which(low_speed_segments$values & low_speed_segments$lengths >= min_low_speed_points)

# Найдем первый такой сегмент после начала катания (окончание катания)
if (length(valid_slow_segments) > 0) {
 end_pos <- sum(low_speed_segments$lengths[1:(valid_slow_segments[1]-1)]) + start_skiing_index
 end_skiing_index <- end_pos
 end_skiing_time <- ski_table$time[end_skiing_index]
 end_skiing_point <- ski_table[end_skiing_index, ]

 cat("Окончание катания определено в точке:", end_skiing_index,
     "\nВремя:", format(end_skiing_time, "%H:%M:%S"),
     "\nСкорость:", round(ski_table$speed_kmh[end_skiing_index], 1), "км/ч\n")
} else {
 stop("Не найдено подходящих сегментов с низкой скоростью")}

# Записываем время конца 2 круга (точка 701) - 15:03:48. Позже сравним визуально (вдруг подгон - наш лучший друг).
ski_table$speed_kmh[650:730]
```

R.Script – 2.4

```r
# Поиск точки перехода-------------------------------------------------------
# Предлагаю находить точку перехода кругов, не просто деля пополам, а через геометрию (то есть точки наиближайшие к точкам начала и конца катания)
# Окончательный алгоритм поиска точки перехода между кругами (чисто геометрический)
find_circle_transition <- function(ski_table, start_index, end_index) {
 # Преобразуем в пространственный объект
 coords <- st_as_sf(ski_table, coords = c("lon", "lat"), crs = 32637)

 # 1. Разделяем маршрут на две равные части по времени
 time_diff <- as.numeric(ski_table$time[end_index] - ski_table$time[start_index])
 mid_time <- ski_table$time[start_index] + time_diff/2
 mid_index <- which.min(abs(as.numeric(ski_table$time - mid_time)))

 # 2. Определяем первый и второй круги
 first_circle <- coords[start_index:mid_index, ]
 second_circle <- coords[mid_index:end_index, ]

 # 3. Находим точку в конце первого круга, ближайшую к началу маршрута
 dist_to_start <- st_distance(first_circle, coords[start_index, ])
 end_first_circle_idx <- which.min(dist_to_start) + start_index - 1

 # 4. Находим точку в начале второго круга, ближайшую к концу маршрута
 dist_to_end <- st_distance(second_circle, coords[end_index, ])
 start_second_circle_idx <- which.min(dist_to_end) + mid_index - 1

 # 5. Точка перехода - середина между этими точками
 transition_index <- round((end_first_circle_idx + start_second_circle_idx)/2)

 # 6. Проверяем, чтобы точка была в разумных пределах
 min_valid_idx <- start_index + 0.2*(end_index - start_index)
 max_valid_idx <- start_index + 0.8*(end_index - start_index)

 if (transition_index < min_valid_idx | transition_index > max_valid_idx) {
   transition_index <- round((start_index + end_index)/2)
}

 # Возвращаем результат
 list(
   point_id = ski_table$track_seg_point_id[transition_index],
   index = transition_index,
   time = ski_table$time[transition_index],
   lon = ski_table$lon[transition_index],
   lat = ski_table$lat[transition_index],
   dist_to_start = as.numeric(st_distance(coords[transition_index,], coords[start_index,])),
   dist_to_end = as.numeric(st_distance(coords[transition_index,], coords[end_index,]))
)
}
```

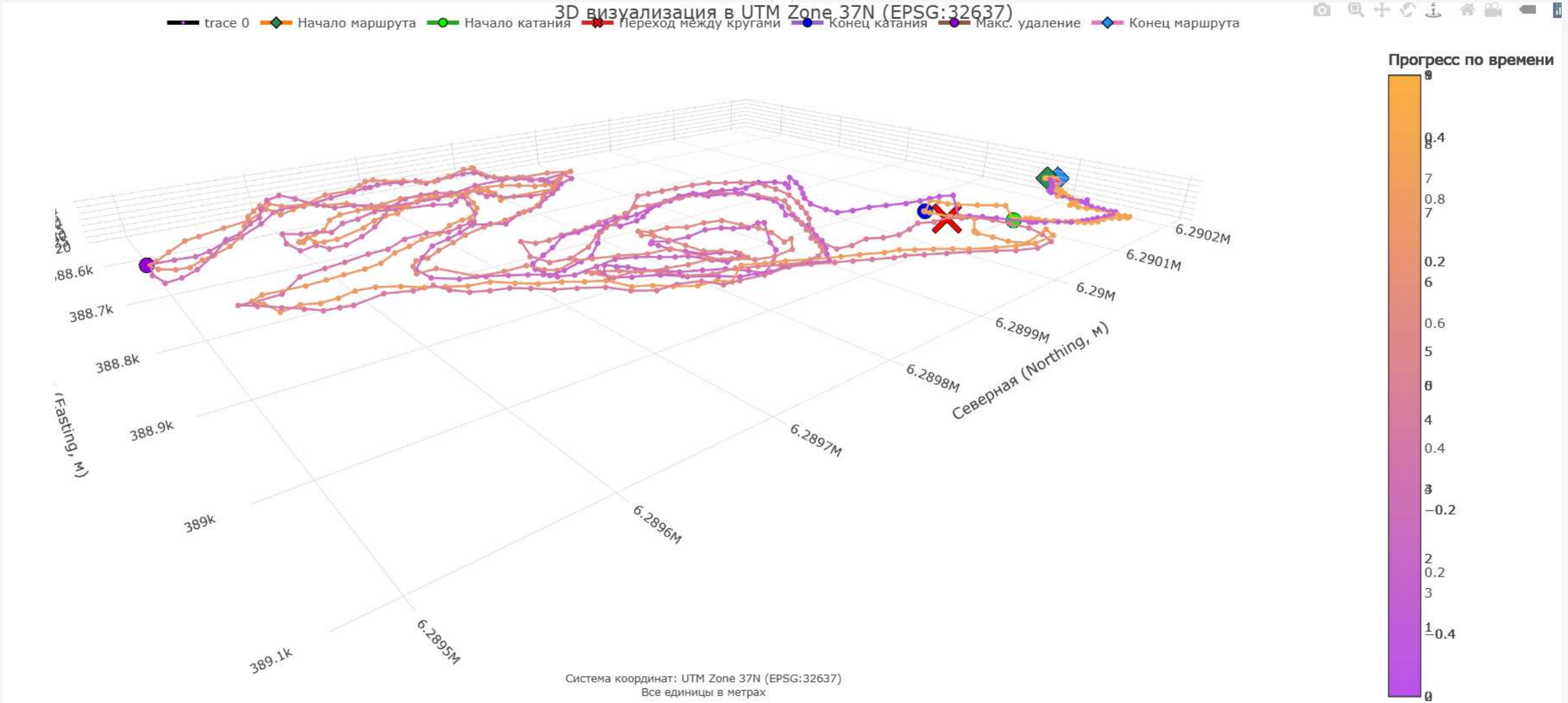
```r
# Поиск точки максимального удаления от базы-----------------------------------
find_max_distance_from_base <- function(ski_table) {
 # Преобразуем данные в пространственный объект
 coords <- st_as_sf(ski_table, coords = c("lon", "lat"), crs = 32637)

 # Базовая точка (первая точка маршрута)
 base_point <- coords[1, ]
 # Вычисляем расстояния от всех точек до базы
 distances <- st_distance(coords, base_point)
 # Находим индекс точки с максимальным расстоянием
 max_dist_index <- which.max(distances)

 list(
   point_id = ski_table$track_seg_point_id[max_dist_index],
   index = max_dist_index,
   time = ski_table$time[max_dist_index],
   lon = ski_table$lon[max_dist_index],
   lat = ski_table$lat[max_dist_index],
   distance_from_base = as.numeric(distances[max_dist_index]),
   speed = ski_table$speed_kmh[max_dist_index]
)
}

max_dist_point <- find_max_distance_from_base(ski_table)

# Вывод результатов
cat("\nТОЧКА МАКСИМАЛЬНОГО УДАЛЕНИЯ ОТ ПЕРВОЙ ТОЧКИ МАРШРУТА\n")
cat(sprintf("%-25s: %d\n", "Номер точки", max_dist_point$point_id))
cat(sprintf("%-25s: %s\n", "Время", format(max_dist_point$time, "%H:%M:%S")))
cat(sprintf("%-25s: %.1f км/ч\n", "Скорость", max_dist_point$speed))
cat(sprintf("%-25s: %.2f м\n", "Расстояние от базы", max_dist_point$distance_from_base))
cat(sprintf("%-25s: %.5f\n", "Долгота", max_dist_point$lon))
cat(sprintf("%-25s: %.5f\n", "Широта", max_dist_point$lat))

#Записываем максимальное удаление от базы: 764.38 м

# Определяем минимальную и максимальную высоту-------------------
min_elevation <- min(ski_table$ele, na.rm = TRUE) #min - 191.1 м
max_elevation <- max(ski_table$ele, na.rm = TRUE) #max - 159.9 м

# Находим точки с экстремальными высотами
min_elev_point <- ski_table[which.min(ski_table$ele), ]
max_elev_point <- ski_table[which.max(ski_table$ele), ]
```

Искомые данные

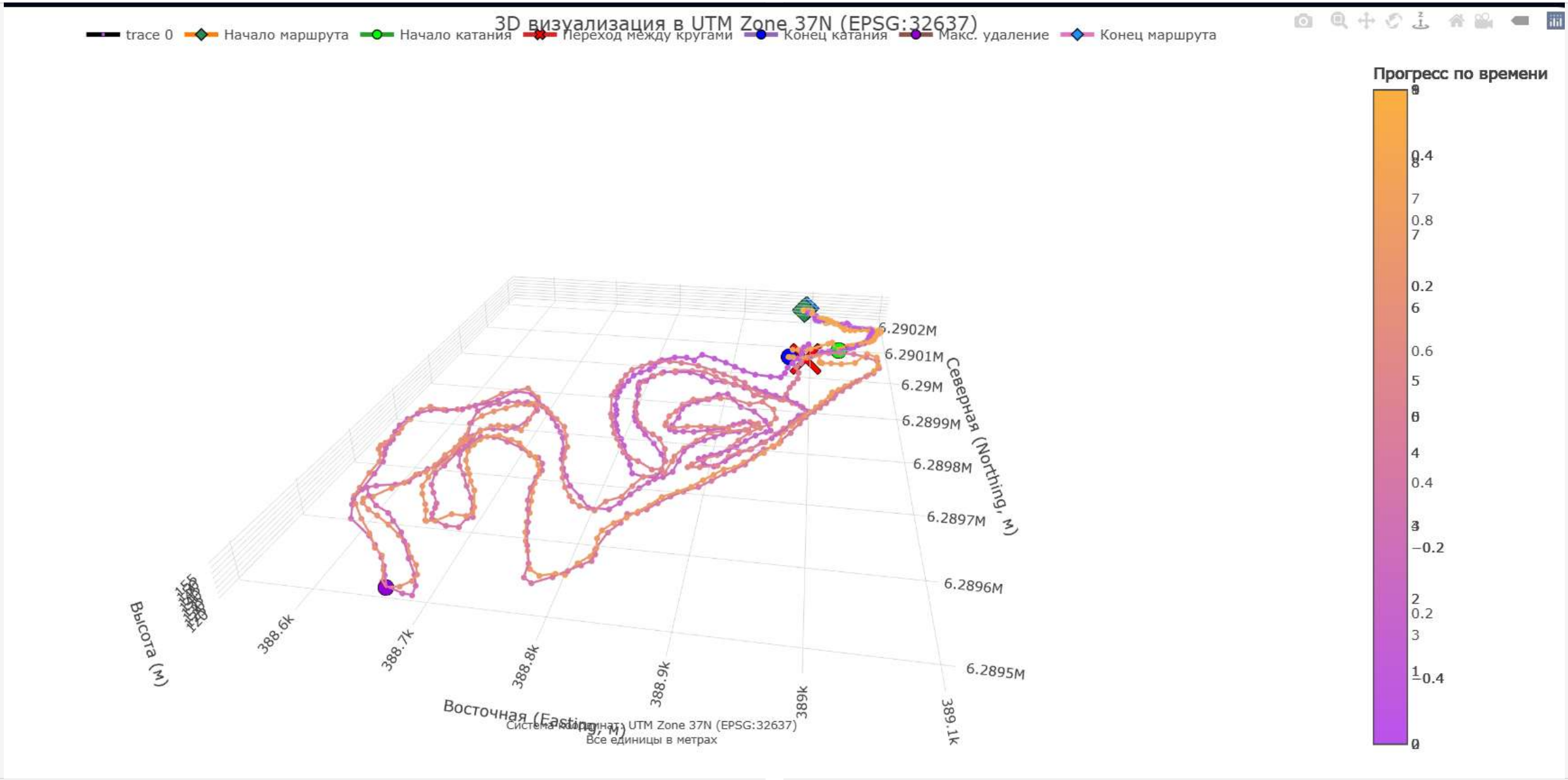
	R
Общее пройденное расстояние, м	9718.95
Длина круга, м	4541.87
Средняя скорость прохождения круга, км/ч	10.18
Средняя скорость подхода/ухода к/от кругу(а), км/ч	2.08
Время начала движения, ЧЧ:ММ:СС	14:00:18
Время начала 1 круга, ЧЧ:ММ:СС	14:09:03
Время окончания 1 круга и начала 2 круга, ЧЧ:ММ:СС	14:36:24
Время окончания 2 круга, ЧЧ:ММ:СС	15:03:48
Время окончания прогулки, ЧЧ:ММ:СС	15:16:08
Высота min, м	119.1
Высота max, м	156.9
Максимальное удаление от базы, м	764.38
CRS	EPSG: 32637

…Остальной код находится в отдельном файле R.script. Далее только визуал…

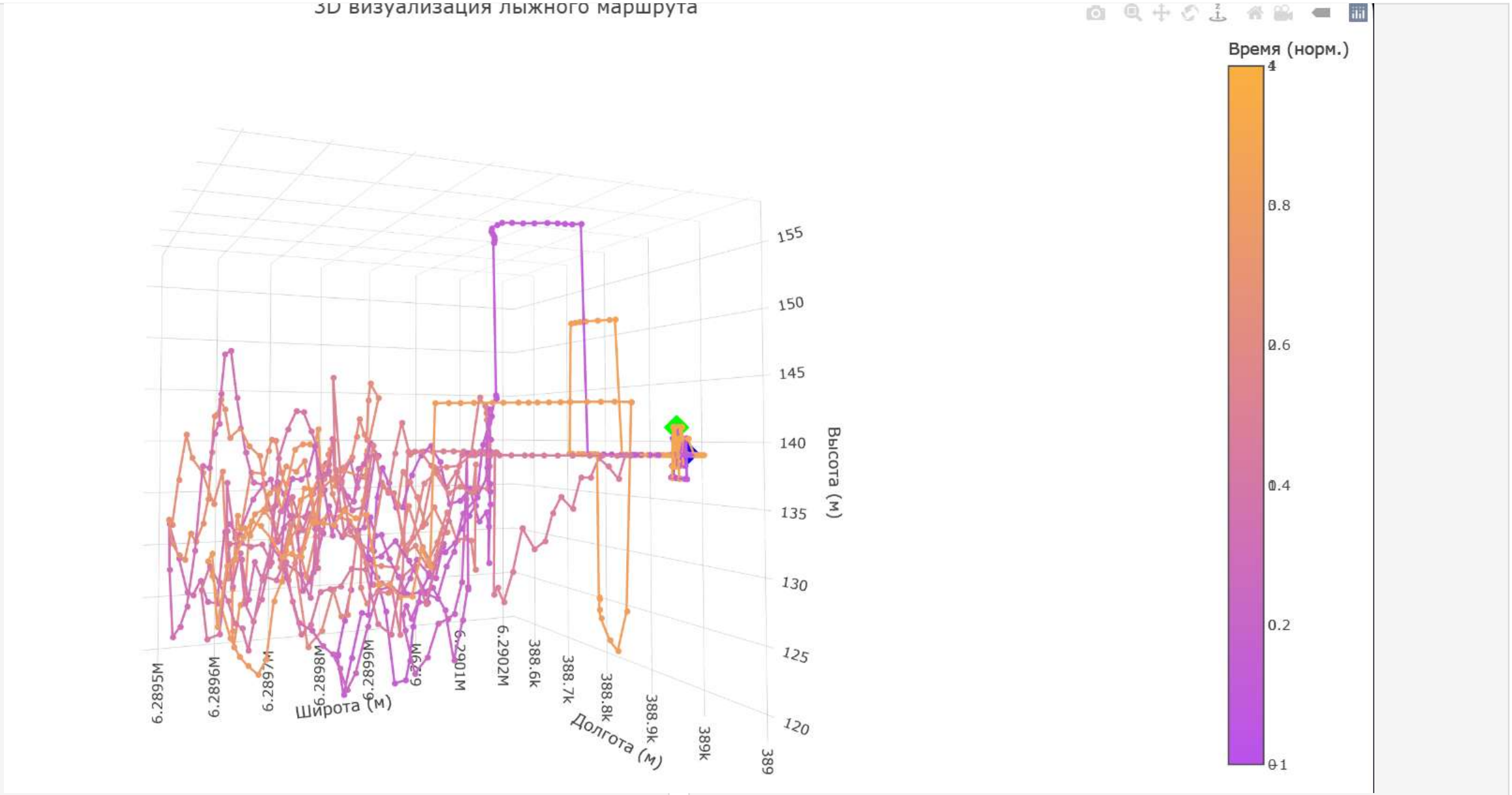
R.Script – 3 (Визуализация)



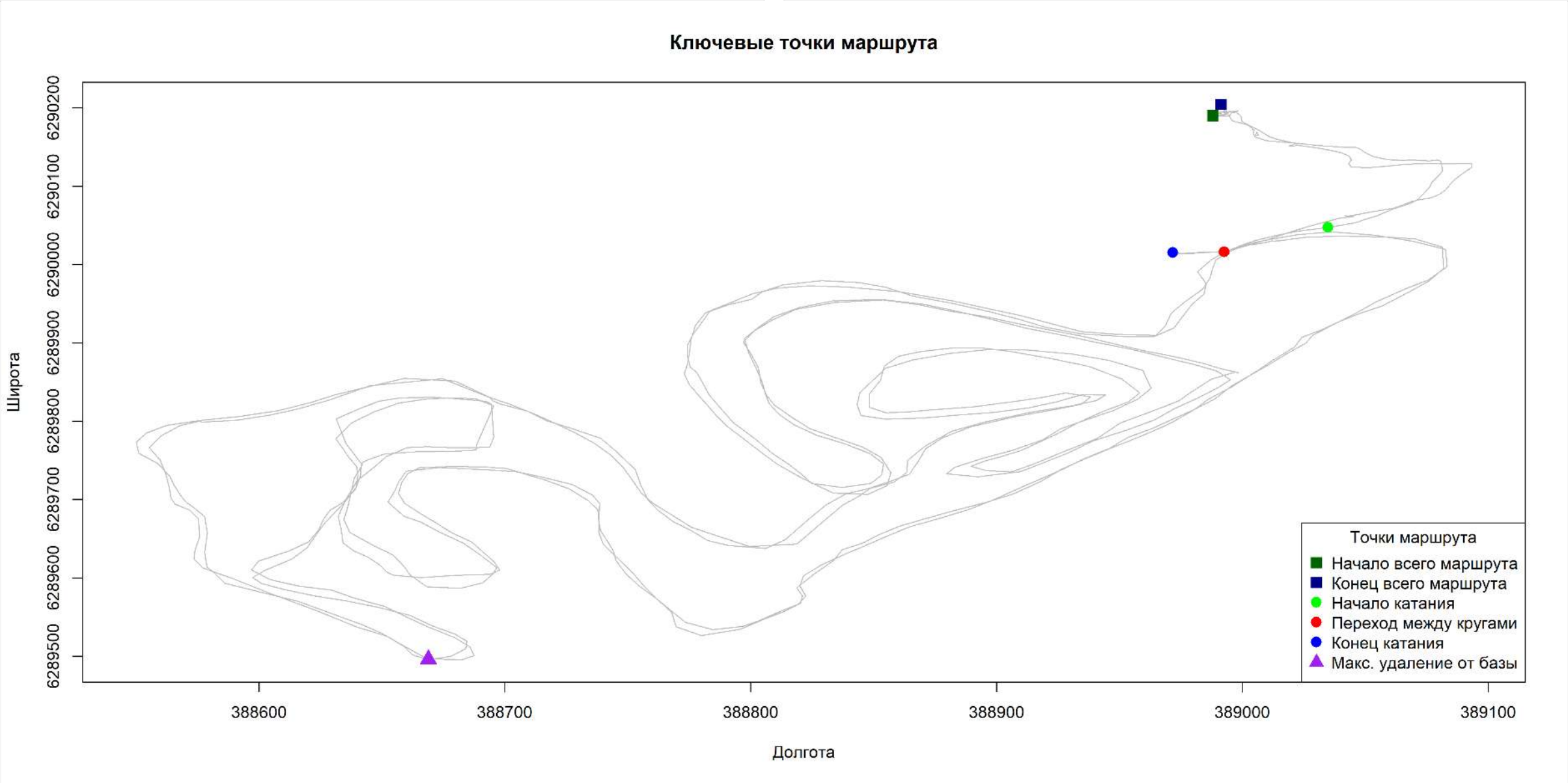
R.Script – 3


3D визуализация в UTM Zone 37N (EPSG:32637)
trace 0
Начало маршрута
Начало катания
Переход между кругами
Конец катания
Макс. удаление
Конец маршрута
Прогресс по времени
Высота (м)
Восточная (Easting, м)
Северная (Northing, м)
388.6k
388.7k
388.8k
388.9k
389k
389.1k
6.2902M
6.2901M
6.29M
6.2899M
6.2898M
6.2897M
6.2896M
6.2895M
Система координат: UTM Zone 37N (EPSG:32637)
Все единицы в метрах

R.Script – 3


3D визуализация лыжного маршрута
Время (норм.)
1
0.8
0.6
0.4
0.2
0
Высота (м)
155
150
145
140
135
130
125
120
Широта (м)
6.2895M
6.2896M
6.2897M
6.2898M
6.2899M
6.29M
6.2901M
6.2902M
Долгота (м)
388.6k
388.7k
388.8k
388.9k
389k
389

R.Script – 3


Ключевые точки маршрута
Широта
6289500
6289600
6289700
6289800
6289900
6290000
6290100
6290200
388600
388700
388800
388900
389000
389100
Долгота
Точки маршрута
Начало всего маршрута
Конец всего маршрута
Начало катания
Переход между кругами
Конец катания
Макс. удаление от базы

R.Script – 3


Профиль высот с кругами катания
Средняя длина круга: 4542 м
Начало ка
Перехо
Конец кат
150
140
130
120
Высота (м)
14:00
14:30
15:00
Время

R.Script – 3


Московская улица
Дачная улица
Черная речка
Лесная улица
Объединенный
институт
ядерных
25
20
15
10
5
0

*Bonus

Колонка hdop (Horizontal Dilution of Precision) в GPX-файлах — это показатель точности GPS-координат по горизонтали, который отражает влияние геометрии спутников на погрешность позиционирования.