ps1_q3

Jingxian(Derrick) Chen

Sep 20th, 2019

I have tried to apply the data from Stats506_F19 repo on Github to check my solutions against the sample measures.

I will display my functions and results in the order of the questions and show the formatted table for the metrics for the test trajectories at last.

Get sample data

First, I need to process the sample data in order to use in my functions.
Here I choose part of the train_trajectories data giving subject_nr is 1 and count_trial is 2.

subset will be an n*3 matrix representng the trajectory (x, y, t) giving subject_nr is 1 and count_trial is 2.

Relevant code:

```
subset <- as.matrix(train_trajectories[train_trajectories$subject_nr == 1
        & train_trajectories$count_trial == 2, 3:5])
head(subset,3)
```

```
##      xpos ypos     tm
## [1,]  -10  414 103611
## [2,]  -10  414 103622
## [3,]  -10  414 103632
```

Solution (a)

Functon description: to solve question 3a, translate input data to begin with time zero at the origin.

Input: an n * 3 matrix representing the trajectory (x, y, t).

Output: an n * 3 matrix representing the trajectory (x, y, t) beginning with time zero at the origin.

Function code:

```
translate <- function(trajectories_data) {
  translate_data <- trajectories_data
  translate_data[, 1] <- trajectories_data[, 1] - trajectories_data[1, 1]
  translate_data[, 2] <- trajectories_data[, 2] - trajectories_data[1, 2]
  translate_data[, 3] <- trajectories_data[, 3] - trajectories_data[1, 3]
  translate_data
}
trans_subset <- translate(subset)
head(trans_subset,3)
```

```
##      xpos ypos tm
## [1,]    0    0  0
## [2,]    0    0 11
## [3,]    0    0 21
```

We could see that the sample data was successfully translated to begin with time zero at the origin.

Solution (b)

Function description: to compute the angle θ formed by the secant line connecting the origin and the final position in the trajectory.

Input: the translated n * 3 matrix representing the trajectory (x, y, t).

Output: a double digit representing the angle between $[-\pi, \pi]$.

Function code

```
angle_compute <- function(trajectories_data) {
  x_final <- trajectories_data[nrow(trajectories_data), 1]
  y_final <- trajectories_data[nrow(trajectories_data), 2]
  angle <- as.double(atan2(y_final, x_final))
  angle
}
theta <- angle_compute(trans_subset)
theta
```

```
## [1] -0.9383606
```

So I have successfully calculated that the angle for the sample data.

Solution (c)

Function description: to rotate the (x, y) coordinates of a trajectory so that the final point lies along the positive x-axis.

Input:
#1: the translated n * 3 matrix representing the trajectory (x, y, t).
#2: the angle computed by the angle_compute function.

Output: an n * 3 matrix representing the trajectory (x, y, t) whose final point lies along the positive x-axis.

Function code

```
rotate <- function(trajectories_data, theta) {
# A represents the rotate matrix.
  new_data <- trajectories_data
  A <- matrix(c(cos(theta), -sin(theta), sin(theta), cos(theta)), nrow = 2)
  new_data[, 1:2] <- t(A %*% t(trajectories_data[, 1:2]))
  new_data
}
Rotate_subset <- rotate(trans_subset,theta)
head(Rotate_subset,3)
```

```
##      xpos ypos tm
## [1,]    0    0  0
## [2,]    0    0 11
## [3,]    0    0 21
```

```
tail(Rotate_subset,3)
```

```
##            xpos ypos   tm
##  [99,] 1030.263    0  981
## [100,] 1030.263    0  991
## [101,] 1030.263    0 1000
```

So I have successfully rotated the sample data to begin at the origin and let its final point lies along the positive x-axis.

Solution (d)

Function description: to combine the functions above that normalizes an n * 3 trajectory matrix to begin at the origin and end on the positive x-axis.

Input: the origin n * 3 matrix representing the trajectory (x, y, t)

Output: an n * 3 trajectory matrix to begin at the origin and end on the positive x-axis.

Function code

```
normalize <- function(trajectories_data) {
  translate_data <- translate(trajectories_data)
  theta <- angle_compute(translate_data)
  normalize_data <- rotate(translate_data, theta)
  normalize_data
}
normalize_subset <- normalize(subset)
head(normalize_subset,3)
```

```
##      xpos ypos tm
## [1,]    0    0  0
## [2,]    0    0 11
## [3,]    0    0 21
```

```
tail(normalize_subset,3)
```

```
##            xpos ypos   tm
##  [99,] 1030.263    0  981
## [100,] 1030.263    0  991
## [101,] 1030.263    0 1000
```

So, I have successfully got the same result from normalize function and the first three functions.

Solution (e)

Function description: to compute the values of question e giving specific subject_nr and count_trial.

Input: the normalized trajectory matrix.

Output: a 1 * 4 vector consists of total (Euclidean) distance traveled, maximum absolute deviation, average absolute deviation, absolute area.

Reference formulas:

(1)

$$tot\_dist = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

(2)

$$max\_abs\_dev = max_i \left| \frac{(y_n * x_i - x_n * y_i)}{\sqrt{y_n^2 + x_n^2}} \right|$$

(3)

$$avg\_abs\_dev = \frac{1}{n} * \sum_{i=1}^{n} \left| \frac{(y_n * x_i - x_n * y_i)}{\sqrt{y_n^2 + x_n^2}} \right|$$

(4)

$$AUC = \sum_{i=1}^{n-1} (x_{i+1} - x_i) * \frac{|y_i| + |y_{i+1}|}{2}$$

Function code

```
compute <- function(trajectories_data) {
  data <- normalize(trajectories_data)
  tot_dist <- 0
  max_abs_dev <- 0
# temp_abs_dev is for computing the max_abs_dev.
  temp_abs_dev <- 0
  avg_abs_dev <- 0
  AUC <- 0
  t_final <- data[nrow(data), 3]
  x_final <- data[nrow(data), 1]
  y_final <- data[nrow(data), 2]
# d2 is for computing the temporary absolute deviation.
  d2 <- (y_final)^2 + (x_final)^2
  for (i in 1:nrow(data)) {
    temp_abs_dev <- abs(y_final * data[i, 1] - x_final * data[i, 2]) / sqrt(d2)
    avg_abs_dev <- avg_abs_dev + temp_abs_dev
    if (i != nrow(data)) {
# d1 is for computing the total distance.
      d1 <- (data[i, 1] - data[i + 1, 1])^2 + (data[i, 2] - data[i + 1, 2])^2
      tot_dist <- tot_dist + sqrt(d1)
      AUC <- AUC + diff(data[, 1])[i] * (abs(data[i + 1, 2]) + abs(data[i, 2])) / 2
    }
    if (temp_abs_dev >= max_abs_dev) {
      max_abs_dev <- temp_abs_dev
    }
  }
  avg_abs_dev <- avg_abs_dev / nrow(data)
  c("tot_dist" = as.double(tot_dist), "max_abs_dev" = as.double(max_abs_dev),
    "avg_abs_dev" = as.double(avg_abs_dev), "AUC" = as.double(AUC))
}
```

```r
result <- compute(subset)
result
```

```
##     tot_dist  max_abs_dev  avg_abs_dev          AUC
##  1063.002271    85.082153     6.032171 46527.000000
```

```r
as.matrix(train_measures[1,])
```

```
##      subject_nr count_trial tot_dist max_abs_dev avg_abs_dev   AUC
## [1,]          1           2 1063.002    85.08215    6.032171 46527
```

So I have got the same measures with the sample data successfully.

Solution (f)

Check my solutions against all the data from train_trajectories.

Relevant code

```r
newdatax <- matrix(0, nrow = nrow(train_measures), ncol = 6)
for (i in 1:nrow(train_measures)) {
    s_nr <- as.double(train_measures[i, 1])
    c_nr <- as.double(train_measures[i, 2])
    subset <- as.matrix(train_trajectories[train_trajectories$subject_nr == s_nr
                                           & train_trajectories$count_trial == c_nr, 3:5])
    S <- compute(subset)
    newdatax[i, ] <- c(c("subject_nr" = s_nr, "count_trial" = c_nr), S)
}
head(newdatax,5)
```

```
##      [,1] [,2]     [,3]      [,4]      [,5]      [,6]
## [1,]    1    2 1063.002  85.08215  6.032171  46527.00
## [2,]    1    3 1032.680  65.41896 14.552720  16242.50
## [3,]    1    4 1153.464  99.18373 18.172340  54516.01
## [4,]    1    5 1049.805  64.34041 20.684094  32053.50
## [5,]    1    6 1961.695 623.55022 92.456128 228192.85
```

```r
head(train_measures,5)
```

```
## # A tibble: 5 x 6
##   subject_nr count_trial tot_dist max_abs_dev avg_abs_dev     AUC
##        <dbl>       <dbl>    <dbl>       <dbl>       <dbl>   <dbl>
## 1          1           2    1063.        85.1        6.03  46527.
## 2          1           3    1033.        65.4       14.6   16242.
## 3          1           4    1153.        99.2       18.2   54516.
## 4          1           5    1050.        64.3       20.7   32054.
## 5          1           6    1962.       624.        92.5  228193.
```

So I verified my solutions against all the train samples.

Compute for the test_trajectories

Relevant code

```
test_measure <- matrix(0, nrow = 5, ncol = 6)
for (i in 1:5) {
    s_nr <- 5 + i
    c_nr <- 1
    subset <- as.matrix(test_trajectories[test_trajectories$subject_nr == s_nr, 3:5])
    result_i <- compute(subset)
    test_measure[i, ] <- c(c(s_nr, c_nr), result_i)
}
# reformat the matrix into data.frame
colnames(test_measure) <- c("subject_nr", "count_trial", "tot_dist", "max_abs_dev",
                            "avg_abs_dev", "AUC")
test_measure <- data.frame(test_measure)
test_measure
```

```
##   subject_nr count_trial tot_dist max_abs_dev avg_abs_dev       AUC
## 1          6           1 1650.769   464.89910   90.387825 275254.35
## 2          7           1 1252.550    35.46823    4.723562  19981.20
## 3          8           1 1069.158    18.41130    1.757015  10133.99
## 4          9           1 1092.076    74.20550    7.302945  36134.40
## 5         10           1 1086.835    85.33933   12.487715  51446.32
```

Reformat as a table

I need to report my results in a nicely formatted table:

Table 1: The test measure results

subject_nr	count_trial	tot_dist	max_abs_dev	avg_abs_dev	AUC
6	1	1650.769	464.89910	90.387826	275254.35
7	1	1252.550	35.46823	4.723562	19981.20
8	1	1069.158	18.41130	1.757015	10133.99
9	1	1092.076	74.20550	7.302945	36134.40
10	1	1086.835	85.33933	12.487715	51446.32