Commuting and Straying Behaviour of Cyclists in Copenhagen

Natalie Brandenberg[1], Laura Markwalder[1]

GEO 880 Computational Movement Analysis (UZH)
Patterns and Trends in Environmental Data (ZHAW)
[1]Department of Geography, University of Zurich
Email: natalie.brandenberg@geo.uzh.ch
lauralynn.markwalder@geo.uzh.ch

Date: 30.06.2019
Equally shared responsibilities within our team, same grade requested.

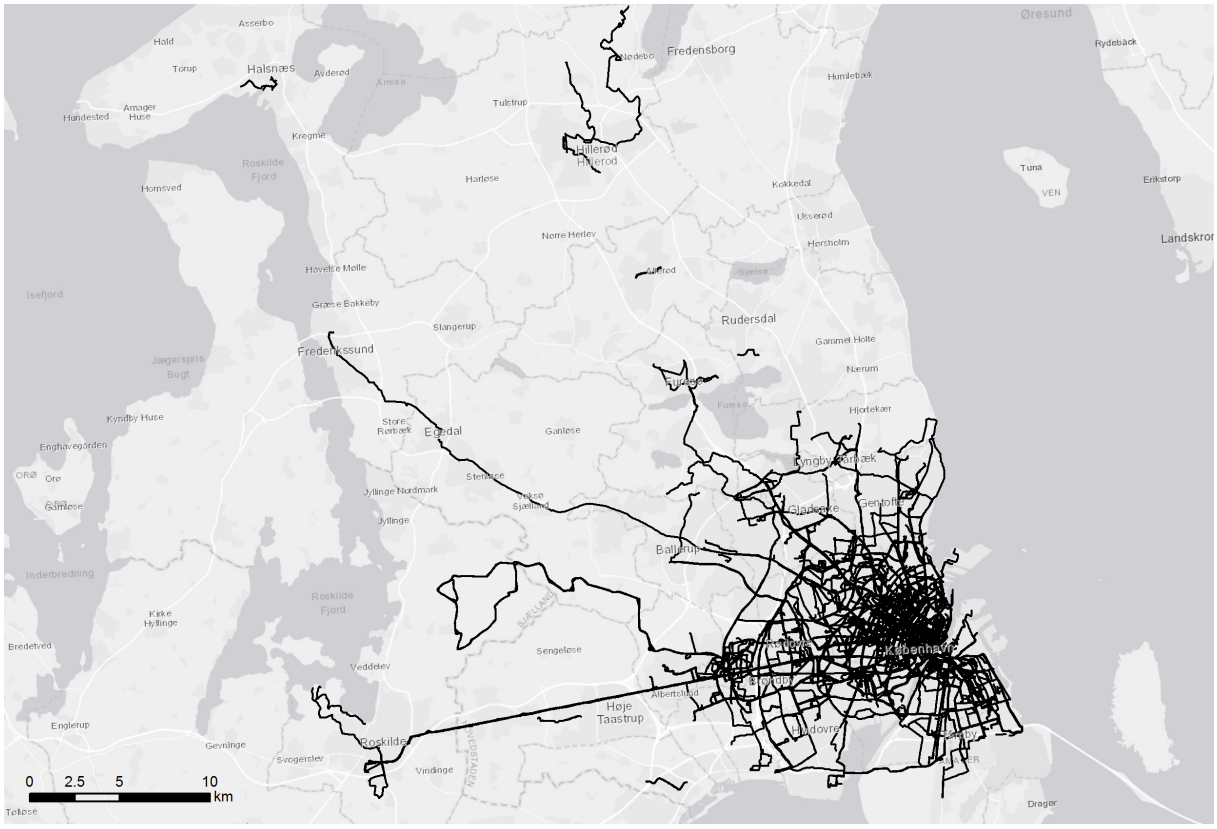

Figure 1: Overview of all trips in the city of Copenhagen included in this study.

1 Background and Research Goals

Cities nowadays provide a unique economic and social landscape. Main services of cities are the creation of jobs, basic services for a dense living population including affordable housing, transportation and transport safety, recreation areas, culture, education and innovation. In this century the number of people in cities will increase to around 70–80% (Bannister 2007). This means that city infrastructures are challenged by the growth of the population. Bicycle riding as a form of mobility in a carbon-neutral way of moving can dramatically improve the evermore heavily utilised transportation infrastructure of cities (Collier et al. 2013).

Biking as a form of transportation is especially widespread in Denmark and of course, in the capital city Copenhagen as well. Bicycles are used for different modes of transportation: e.g. to commute, for leisure and sport activities. Bike lanes make this possible and are heavily

used to the point of congestion during rush hours (Ministry of Foreign Affairs of Denmark n.d.).

Using GPS data from 185 cyclists in Copenhagen, we evaluate their individual trips as well as their transport behaviour over-all in terms of commuting and stray cycling. Additionally, we analyse general temporal and spatial patterns in bike usage over the study period of three weeks.

Our research questions are:

RQ.1 Using two methods, how can trips of individual cyclist be categorised into commute and stray types?

RQ.2 How can the dwellings of different types of cyclists be found using the GPS data from their trips?

Our research is therefore split into a trip-based analysis in research question 1 and a cyclist-based analysis in research question 2.

2 Data and Methods

2.1 Dataset

Hans Skov-Petersen from University of Copenhagen provides our dataset. The GPS data describing cyclist movements in the city of Copenhagen and surrounding areas are sampled over 3 weeks. Each fix point in the dataset contains the following attributes:

- ObjectID
- RespID (= Cyclist)
- TripID (= Trajectory)
- X-Coordinate
- Y-Coordinate
- Datetime

The characteristics of the bicycle data enable or disable certain analytical methods. Therefore, taking a closer look at the dimensions of representing movement based on the literature is necessary. Table 1 shows a summary of the important characteristics.

Table 1: Dimensions of representing movement (based on Laube (2017) and Laube (2014)).

Properties of the movement spaces	Properties of the movement	Perspective of observation	Semantic level of interest
- Continuous movement spaces, 2D - Conceptual models: • Entity based: Trips • Field based: Heat map - Data structures: • Raster: Heat map • Vectors: Fixes and trips	- Limited (to streets and paths) - Intermittent, Periodic - Active	- Lagrangian (GPS) - Event-based - Active tracking - Off-line	- Fix, move, segment

To start the data exploration a summary of all trips' characteristics is shown in Table 2.

Table 2: Summary of bike data in Copenhagen.

Time period	3 weeks in 2011 02.–08.05 / 23.–29.05. / 13.–19.06.
Number of trips	1488
Number of respondents	185
Mean extension (trip length in time and distance, speed)	41 min, 9182 m, 13.5 km/h
Min extension (trip length in time and distance, speed)	5.05 min, 67.72 m, 0.1 km/h
Max extension (trip length in time and distance, speed)	328 min, 74600.96 m, 28.1 km/h
Mean trip count per cyclist	11.4
Min trip count per cyclist	1
Max trip count per cyclist	30

Figure 2 shows the temporal distribution of all trips. For a better comparison the number of rides are normalized by the maximum trip count per day. It is clearly visible that temporal differences in number of trips over the days and between weekdays occur. Overall, in the first two study weeks the highest number of trips are recorded in the morning (7-8) and afternoon 15-18) on weekdays and at noon (12-15) on weekends. The third week shows a similar temporal pattern except for Monday. As it was the day of Whitsun and therefore a holiday, the pattern corresponds with the temporal distribution of the weekend.

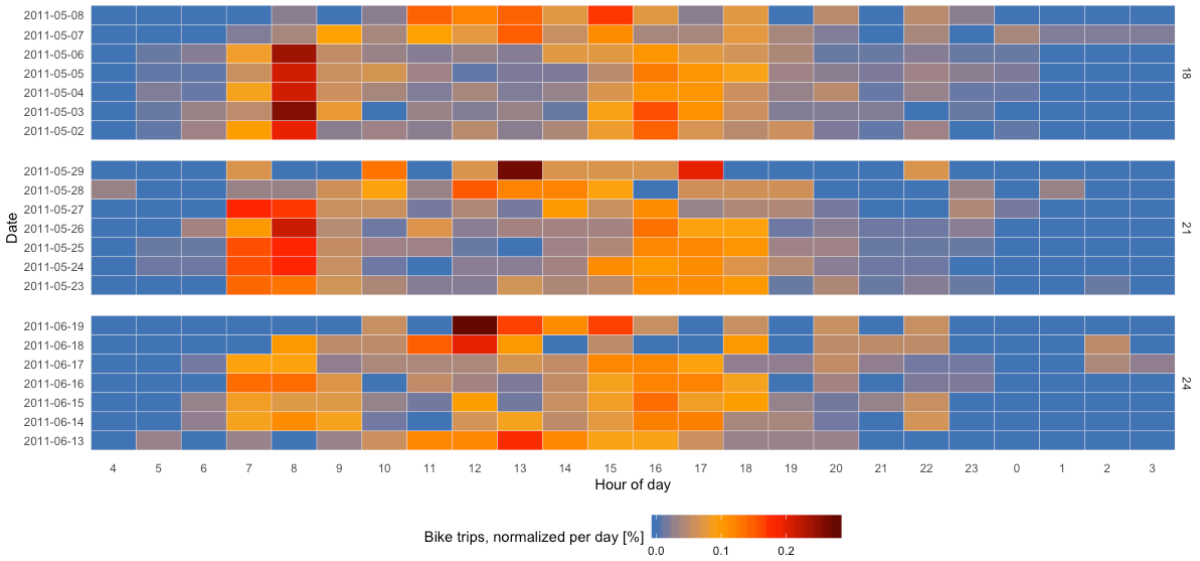

Figure 2: Number of trips per hour (normalised) for the three study weeks (week 18, 21 and 24 of 2011).

Spatially and temporally, an accumulation of trips is detected in the city centre as shown in figure 3. For better readability, the image section is focused on the area around the city. Therefore, a few short trips to the north are cropped.

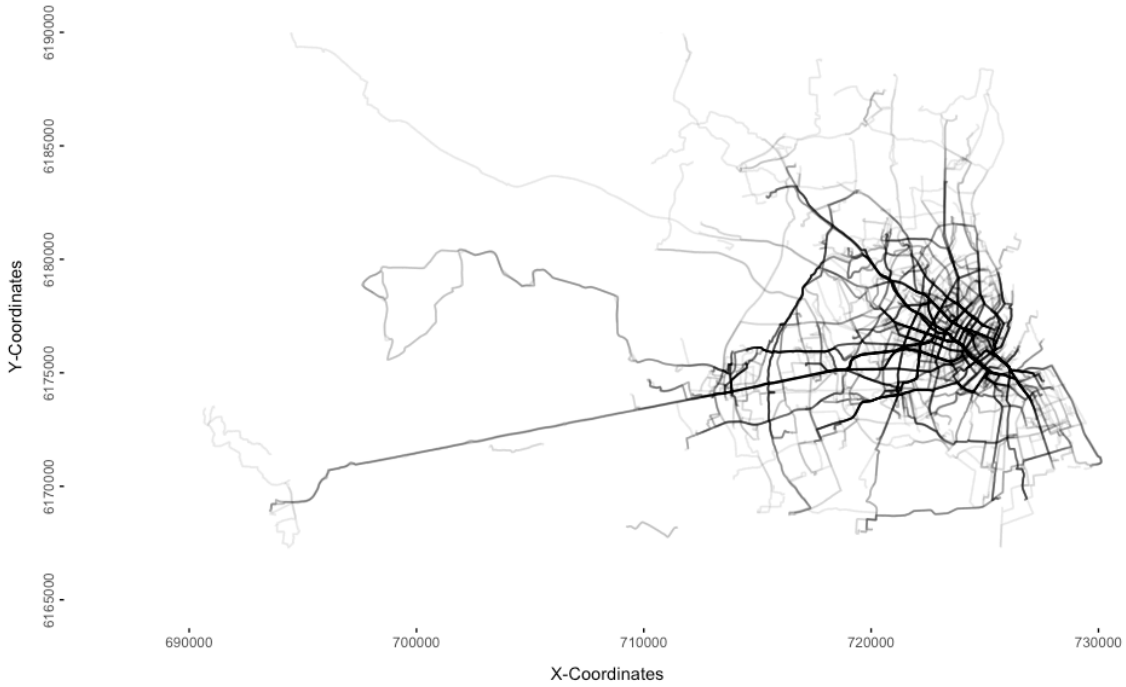

Figure 3: Overview of the trips around the city center in Copenhagen. The saturation reflects the number of trips.

2.2 Pre-processing

The initial dataset had issues with the trip segmentation. Trip IDs changed seemingly within trips, i.e. where there was neither a gap in time for more than a minute or a big spatial difference over a few hundred metres. A new segmentation of trips was made based on a time difference of more than 30 minutes or a spatial difference of more than 1500 metres between two fixes. To get the final dataset, only certain trips are selected. Trips need more than 10 fixes, because the high temporal sampling rate (an average of 18 seconds between fixes) means that fewer fixes result in spatially very small trips (only a few metres across), which we do not consider as trips. An example of this can be seen in figure 4 where the trip number 3 of respondent 58093101 with only 13 fixes is recorded. The low spatial extent (28 metres in direction x and 30 metres in direction y) and the wild zigzagging suggest that this might have been a respondent standing still and it recorded the GPS error.

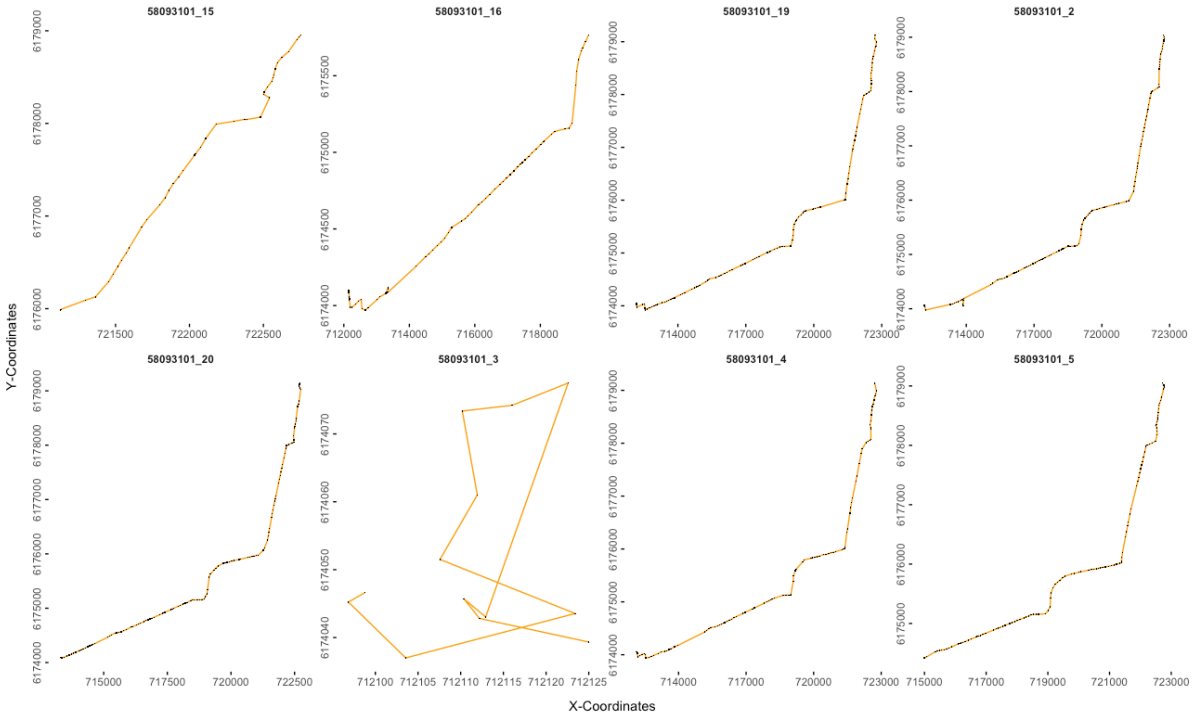

Figure 4: A selection of trips of respondent 58093101, a typical commuter.

Trips with a mean speed of faster than 120 km/h are also not selected, since this is also most likely a GPS error. Additionally, trips need to be more than 0 metres long and must have taken more than 5 minutes to be considered in our study.

2.3 Methods

Different methods were explored to answer RQ.1, the trip-based analysis of the data. The methods depend on the focus we choose to analyse data under. In the assignment by Skov-Petersen we are asked to assess the extent to which Trip IDs spatially overlap with all other trips performed by this respondent. We should also divide trips into categories of commute and stray proportions. The focus can thus either be on how much trips of one respondent overlap with each other, or it can be more on the issue of commuting. Commuting we define as cycling from places to places, which are visited more than once by the same respondent, i.e. considering the start and end fixes of a trajectory.

A similarity measure of trajectories seems to fit both requirements. Dynamic time warping (DTW), for example, would be suitable because it allows for varying sampling rates and works with Euclidean distance (Toohey & Duckham 2015). The resulting values per trip would allow us to estimate how near it is to other trips which models the spatial overlap. Also the more similar trips are the more we can assume that start and end destination were similar as well. However, the very high numbers of points to compare (136’338 fixes) proved to be too much for the computer to calculate in a feasible amount of time. Preliminary results from just one respondent were difficult to interpret as well as high numbers did not seem to correspond with visually dissimilar trips.

In the end, two other methods are used to answer RQ.1. The first method concentrates more on the spatial overlap, while the second method focuses on commuting in the sense of known start and end destinations.

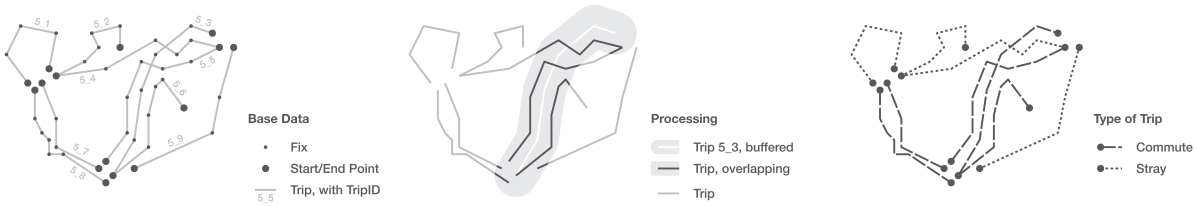

Figure 5: Buffer approach.

The first method is a buffer approach, a simplified explanation of which can be seen in figure 5. Trips per respondent are buffered with 50 metres on both sides (100 metres is the width of some of the wider bike highways in Copenhagen). Afterwards, we compare how much all trips of this same respondent lie within the buffer. If this line overlaps for more than 70% we consider this a commute. The median value of all comparisons over 70% for this trip is used to calculate its final commute proportion. A trip with no overlaps of above 70% receives a commute proportion of 0.

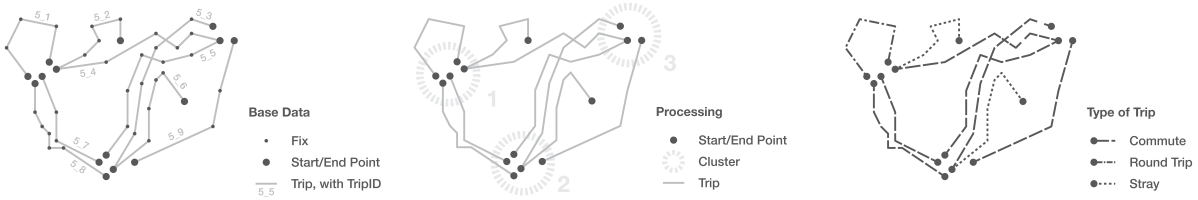

Figure 6: Cluster approach.

The second method is a clustering approach (simplified explanation in figure 6). Using the DBSCAN clustering algorithm on the start and end point of each trip, we can cluster points together with the following DBSCAN parameters: 500 metres, min. 3 points in search radius. DBSCAN allows for points to not belong to any cluster, which is what we need for stray trips. If a cyclist drives to or from a non-cluster point, this is considered a stray trip whereas driving to the same cluster they started from is considered a round trip.

An artificial example in figures 5 and 6 of both methods and their resulting classification shows their differences. The buffer approach is a local analysis and will show commonly used routes, while the cluster approach is more globally interested in movements and especially destinations over the whole city.

To answer RQ.2 the respondents have to be divided into strayers and commuters. If more than 60% of the trips of one cyclist are commutes this person is considered a commuter. This threshold was determined by trial and makes it a bit more difficult to be a commuter than a person with stray behaviour. Only the trip types of the cluster approach are used since this approach resembles the method we use to determine where a person lives. Moreover, only very few respondents (n = 16) change type (commuter versus strayer) between the two approaches. A dwelling or domicile was defined as the place where the respondent leaves first thing in the morning and returns to from the last trip in the evening. To calculate domicile coordinates for each respondent, the first and last point per day are selected, but only if they belonged to the modal cluster of that respondent. Using those fixes, the median value for x and y coordinates define the dwelling.

2.4 Problems and Limitations

Because of the limitations with the intended approach for RQ.1, the DTW similarity measurement described in section 2.3, this method was not used any further. Instead, different

data science ideas and choices have been explored. Some of these ideas rely on assumptions, which might considerably influence the results.

At the very beginning of every data science project, it is unavoidable to take a closer look at the data. Here the problems with the initial Trip IDs became obvious. An influencing decision has been made with the new trip segmentation. The limited background knowledge about the original data pre-processing complicate the parameter definitions, which define the new Trip ID. All further calculations are based on this process. As Laube and Purves (2011) point in their paper trajectory data require some filtering and segmentation that can be achieved with various approaches.

Additionally, no influence on the data collection was possible. Our data science ideas are based on the assumption that all cyclists' rides have been recorded.

Deduced from that, the modal cluster of all first and last fixes of a certain respondent (= most occurring cluster) represent the dwelling. This choice ignores exceptions as for instance overnight stays with a boy-/girlfriend. In a further context it can be discussed if such other frequently visited places should also be counted as dwelling or how workplaces can be determined.

To get the degree to which trips overlap with each other in the buffer method, we decided to use the shorter of the two trips in the comparison in order to avoid values over 100%. This leads to very short trips, which fall completely within the buffer of the longer trip to be counted as commutes. This is the final implementation of this method. Alternatively, one could just use the length of the trip not being buffered as a default. Using this approach, other complications would arise, for example different overlap values for the same two trips being compared.

Some thresholds influence all our data science choices. The best fit usually was found using histograms and testing. But still, it is impossible to exclude a certain bias of an expected pattern during these decision processes. Finally, as in the case of the DTW similarity measurement, it may happen that an idea has been tested and run with a subset but did not work for the whole dataset at the end.

3 Results

The results follow for the trip-based (RQ.1) and the respondent-based (RQ.2) analysis. After looking at differences between the two methods, general patterns in the data are discussed.

3.1 Trip-based analysis

In figure 7, examples of how the two methods work, can be seen for a typical respondent.

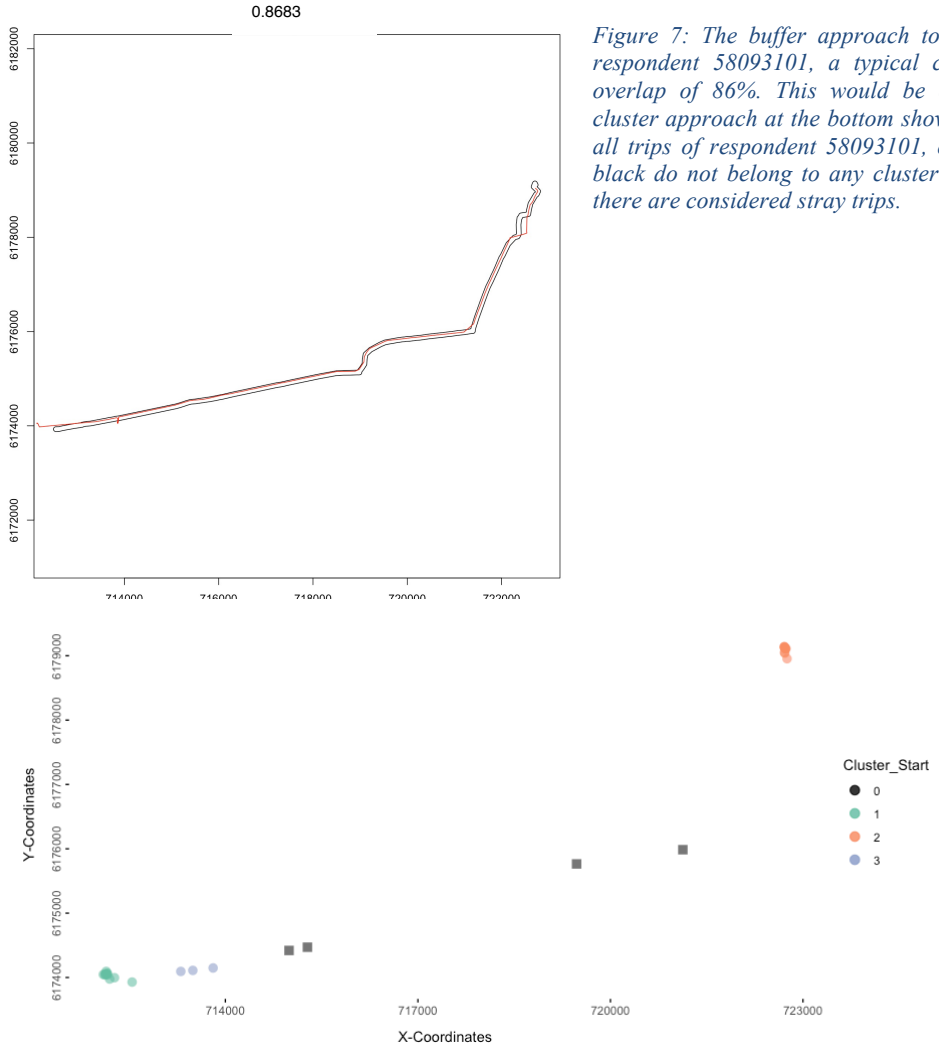

Figure 7: The buffer approach to the left shows two trips of respondent 58093101, a typical commuter, with a very high overlap of 86%. This would be considered a commute. The cluster approach at the bottom shows the start and end points of all trips of respondent 58093101, coloured by cluster. Fixes in black do not belong to any cluster and trips starting or ending there are considered stray trips.

Applying these methods to all trips, we are able to divide them into commuting and straying trips. In the case of the cluster approach, an additional type of trip results: the round trip. Round trips are trips, which start and end in the same cluster.

Table 3: Statistics of commuting, straying and round trips per approach

	Buffer Approach			**Cluster Approach**		
	Number of Trips	Mean Trip Length [m]	Mean Trip Duration [min]	Number of Trips	Mean Trip Length [m]	Mean Trip Duration [min]
Commuter	716	10’730	45	700	8828	40
Strayer	772	7420	37	694	7684	35
Round trip	0	0	0	94	21’794	91

Table 3 shows that the numbers of trips are quite evenly distributed between the two approaches and also within them. Of course, there are fewer round trips, as this is a special case. The higher numbers of stray trips for the buffer approach compared to the cluster approach can be explained with the stronger spatial constraint. The 100 meters buffer allows for fewer deviations of the other trips compared to the 500 meters search radius of the DBSCAN. The mean trip length and duration for the stray trips are very comparable as well. The longer trip length and duration for commuter trips in the buffer approach versus the cluster approach can be explained with the fact, that some very long round trips are grouped

in there (see figure 8). This round trip happens twice weekly, when one respondent rides their bike the whole morning, which results in the longest trips over all. This, of course, raises the trip lengths and duration for the round trips in the cluster approach significantly.

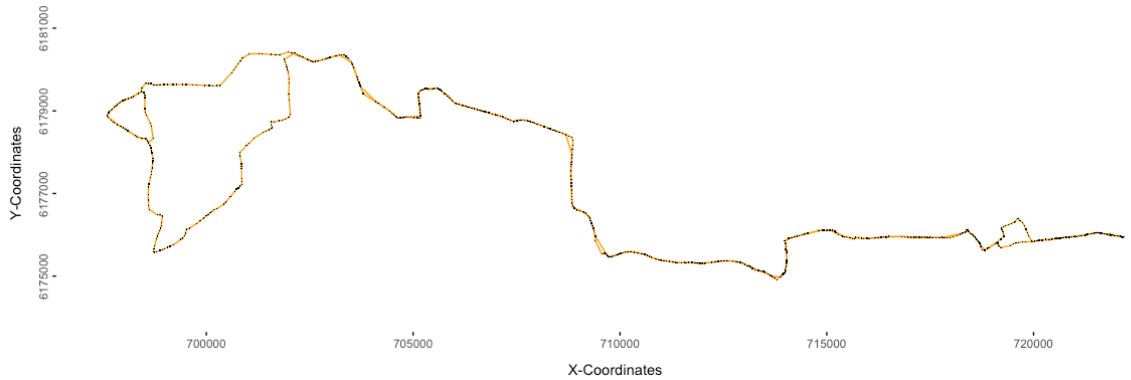

Figure 8: The longest trip in time and distance is a round trip performed by respondent 57090098, the long distance biker.

Table 4: Classification of trips with two approaches.

		Cluster approach		
		C	S	Total
Buffer approach	C	525	191	716
	S	175	597	772
	Total	700	788	1488

Table 4 shows how consistent the two methods classified the trips into commuting and straying trips. The round trips of the cluster approach were grouped in with stray trips, since this method is focused on moving from one cluster to another. 75.4% of trips stay within their type for both methods. Half (94) of the changes from commuting trips to straying trips (changing from the buffer method to the cluster method) are accounted for this since they are round trips.

Figure 9 below shows where the trips are and whether they belong to the same type in both methods or not. This difference in the two methods can be seen very well for the round trip long distance cyclist (loop to the west). The buffer method classified these long distance trips as commutes since the cyclist is very consistent in their chosen path. In general, most differences in trip classification can be seen in the city centre. Here cyclists have more bike lanes to choose from which has a great influence on the results of the buffer method. Towards the edge of the city, there are fewer bike lane options, which makes the buffer classification more consistent. Since the cluster approach does not depend on the exact lane taken it is less sensitive to the way itself but where it start/end.

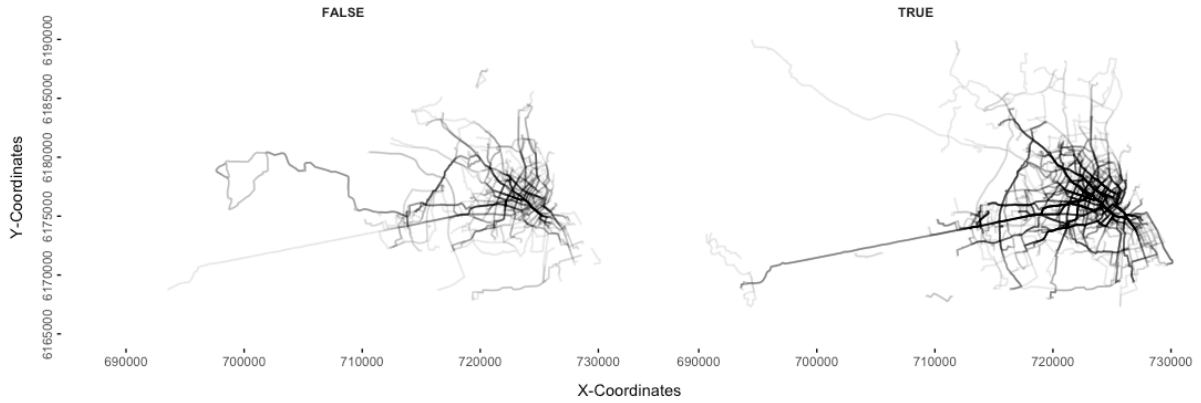

Figure 9: Trips that are not classified into the same type (commuter or strayer) by both methods (left) and trips that are classified to the same type by both methods (right) which make up 75.4% of all trips.

Even though the numbers in Table 3 for the two methods are very similar, the cluster approach models commute better, in our opinion. Looking at a distribution of the number of all trips over the day, it is clear that the buffer method does not capture the maybe more standard definition indispensable of commute, i.e. cycle somewhere in the morning, come back in the late indispensable afternoon, very well. This is because this approach focuses more on spatial overlap in general, which could happen anywhere along the route and is not bound to destinations. The cluster approach portrays the morning and evening rush hours with many commutes better (see figure 10). The temporal distribution also shows the higher numbers of stray trips on the weekend when people are driving for pleasure along known routes. The clear bimodal distribution also disappears in the last study week (24). This was the week after Whitsun, which people might have taken as a holiday.

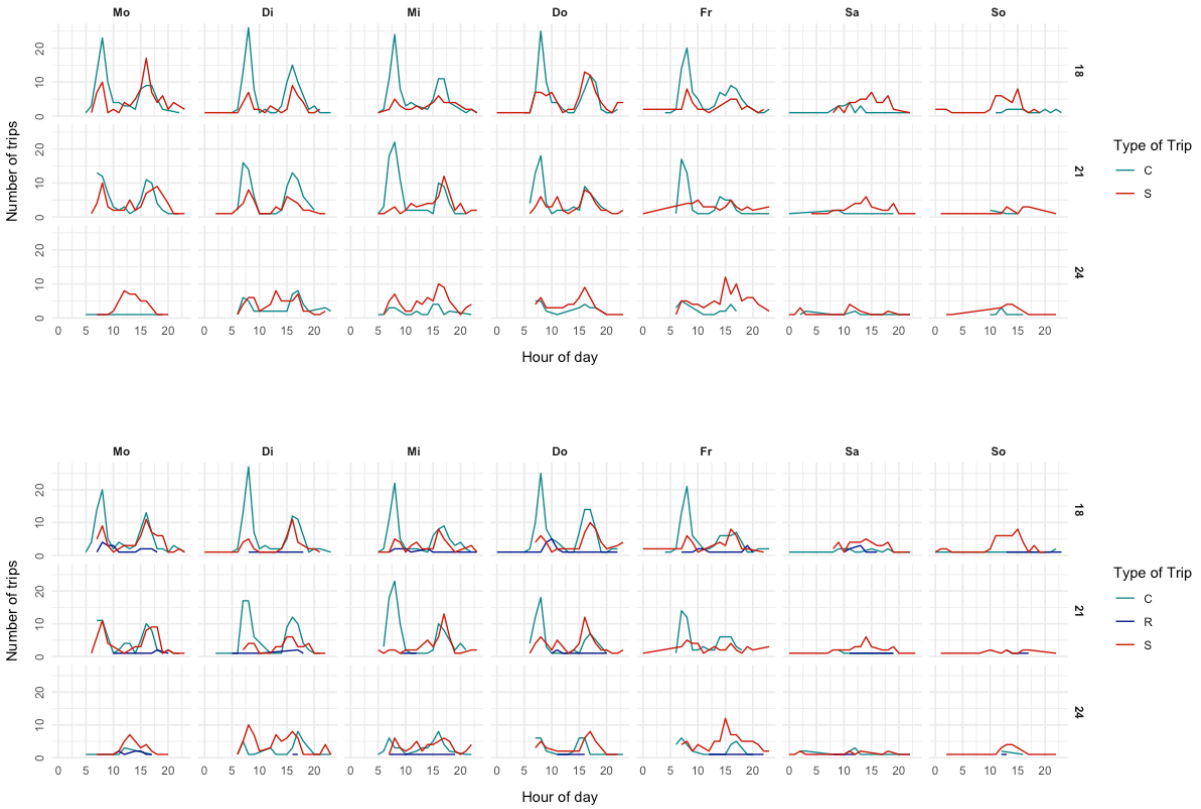

Figure 10: Temporal distribution of trip types over each day of the week for each week of the study period for the buffer method (top) and the cluster method (bottom).

The trips in figure 11 show that there is no clear difference in spatial distribution between either the two methods or the different trip types. In general, there is a high concentration of trips in the city centre with arms stretching to the southwest and northwest. The concentration of trips in the city centre is to be expected. This comes with a general increase in people and activities in the centre. The arms stretching out are probably an effect of the respondents in the study, e.g. the very straight arm to the southwest leads to the city Roskilde where one respondent probably works, and are not representative of bike trip distribution in Copenhagen in general. A bike lane leading around the city in a half circle can also be seen very well on Mondays and Fridays. The round trips do not show any pattern, except for the long distance bike trips on Mondays and Thursdays, which were already mentioned.

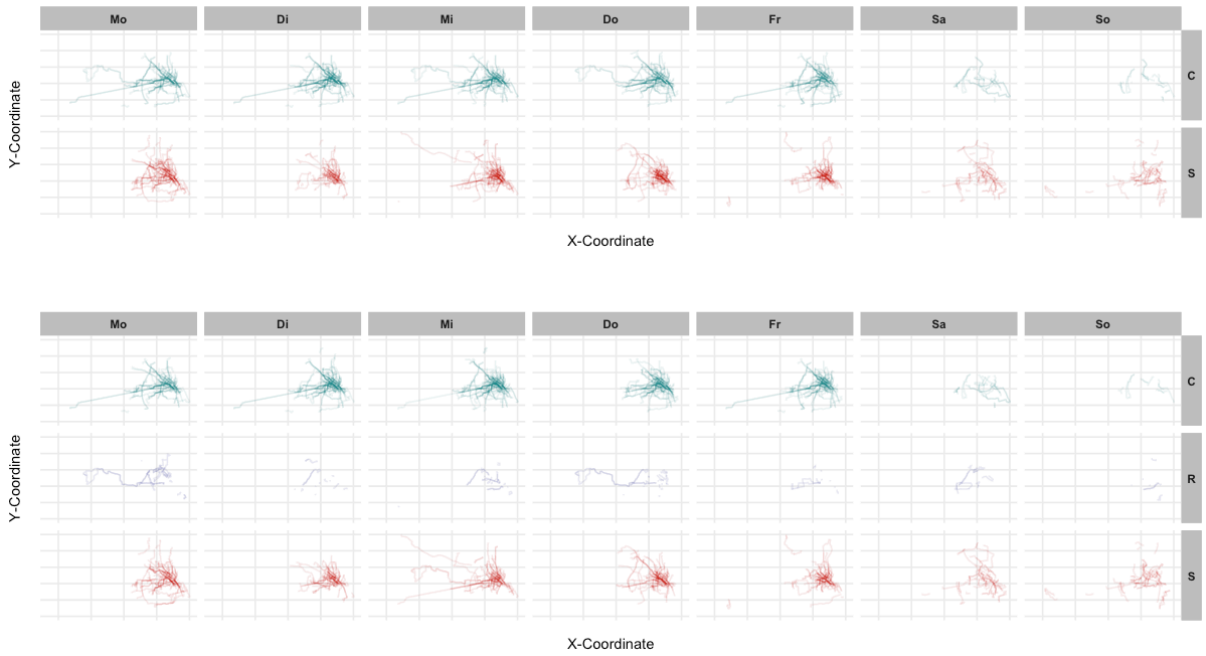

Figure 11: Spatial distribution of trip types over each day of the week for the buffer method (top) and the cluster method (bottom).

The RGB composite image made in ArcMap show where which type of trips occur most as well as the top 9 sightseeing attractions in Copenhagen from Tripadvisor (2019) (see figure 12). The city centre in purple has equal numbers of straying (red) and commuting (blue) trips with a slight tending towards more straying trips. The end of the streets in the west and centre west end are represented in blue, which means more commuting trips go or end there. The very pronounced green line (round trip) in the west is partly following a tree belt, which leads around the city. The suspicion that commuters would tend to avoid sightseeing attractions while stray trips lead to them could not be confirmed with this attempt.

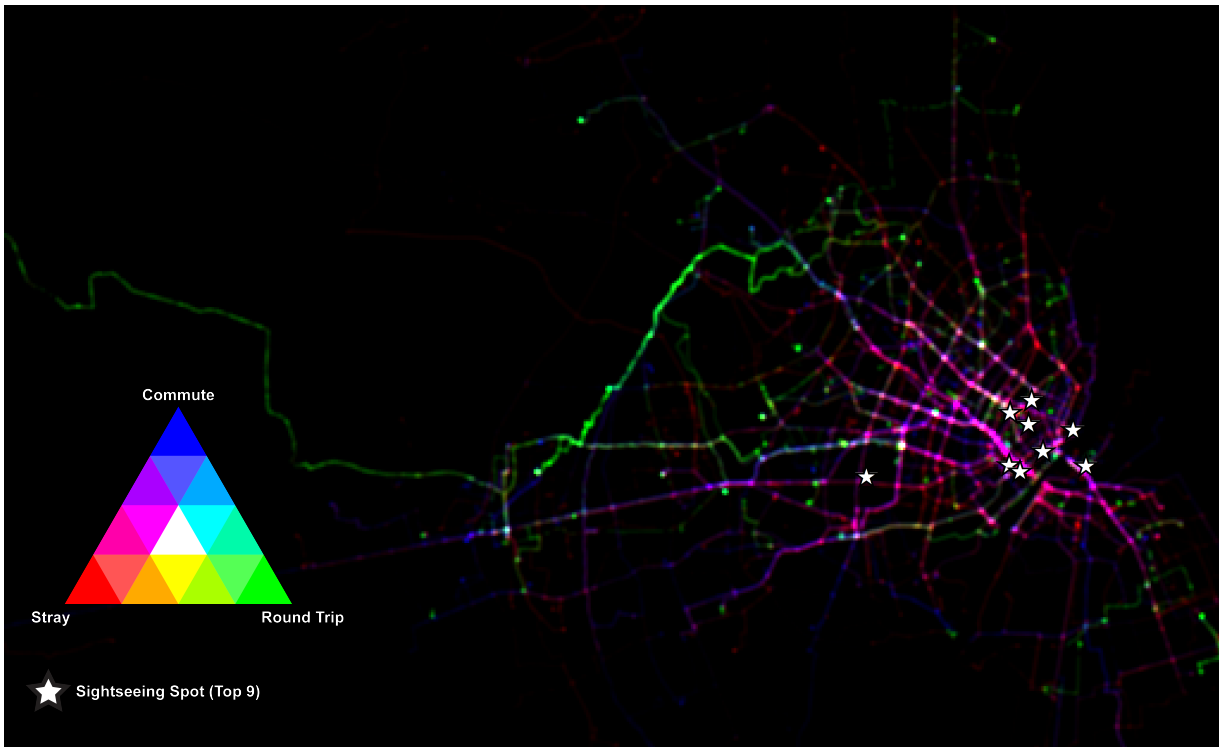

Figure 12: RGB-composite (S, R, C) with Top 9 sightseeing places from TripAdvisor (white star symbol).

3.2 Respondent-based analysis

With more than 60% of commuting trips per respondent the cyclist is classified as C (= commuter) and otherwise S (=stray) for both approaches. The relation in the matrix Table 5 state how many trips are categorised into the same type and how many change from the perspective of buffer to cluster approach. Comparing the cluster and buffer results, a relatively similar classification of commuting and straying cyclists is achieved. Overall, the consistency amounts to 79.5%.

Table 5: Classification of respondents with two approaches.

		Cluster approach		
		C	S	Total
Buffer approach	C	46	23	69
	S	15	101	116
	Total	61	124	185

A total of 185 cyclists are classified. With the buffer method 61 cyclists are C and 124 S whereas the cluster method classifies 69 respondents as C and 116 as S. While the category of C is dependent on different constraints, for example where the threshold of buffer overlapping for commute is set, the S trips are not dependent on anything. That causes a lower number of C respondents for both approaches compared to the S riders and also the discrepancy of the two methods. The greater change of C to S compared to S to C cyclists can be justified with the circumstance that the round trips are counted as S in the cluster approach.

To answer our research question RQ.2 the spatial distribution of the computed dwellings is mapped in figure 13 (for a clearer illustration clipped to city centre where the trip as well as the dwelling density is highest). The size of the domicile circle gives an idea how accurate the location with regard to the included start and end points is. A higher mean distance deviation of the median fix to all other points of a certain respondents' cluster implies a higher spatial

uncertainty. Additionally, it is important to respect the number of points laying in the defined home cluster visible in the colour saturation. In our study, the correlation between point count and mean distance deviation is non-linear. Therefore, a more precise exploration of accuracy is performed later.
In summary, no clear spatial aggregation of dwellings dependent on the type of cyclist is detectible. The dwellings are more or less homogeneous distributed in the city centre. Even though, a slightly higher concentration of domiciles is computed in the city centre for stray cyclists. Those respondents already live in the area where the densities of places of work as well as the leisure opportunities are assumed to be very high. The resulting elimination of long commuter routes and the possibility of using a great variety of roads may be a possible explanation of this pattern.

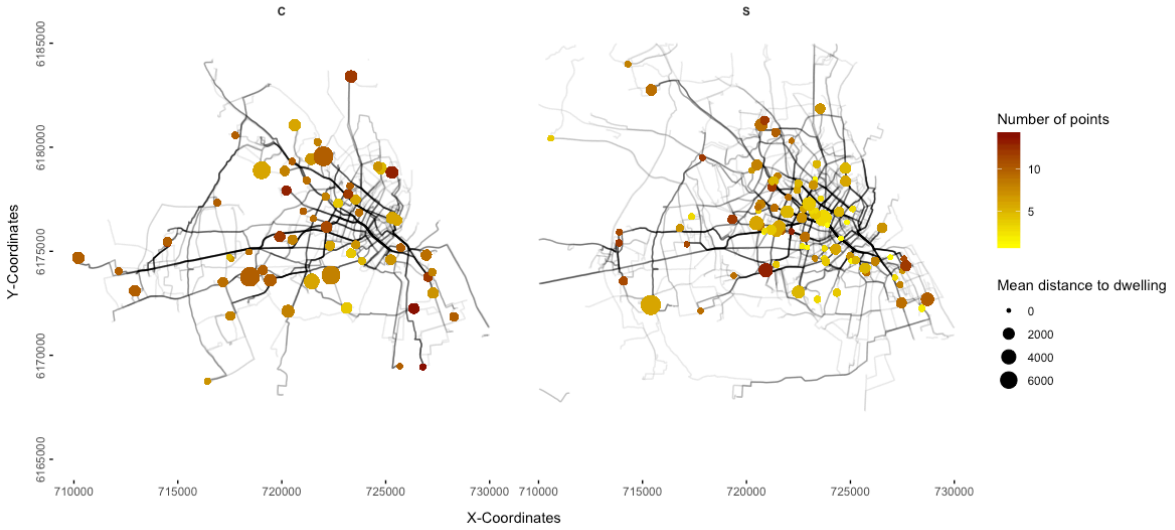

Figure 13: Computed dwellings with its accuracy, i.e. size represents mean distance [m] of considered points and colour the number of considered points. The trips are displayed in black, whereas the saturation denotes the frequency (i.e. the darker the more trips).

Figure 13 enables no statement regarding the direction of the corresponding trips to a certain dwelling. Therefore, all trips and the dwelling belonging to two respondents already mentioned in chapter 3.1 are displayed in figure 14. It allows arguing in more detail where the respondents cycle and verify our previous hypothesis. In this case study the cyclist defined as a typical commuter (RespID 58093101) clearly commutes to the city centre following a very regular path. The long distance biker (RespID 57090098), classified as a straying cycler moves from their home in the city centre in many different directions taking different paths. Both of them do not cycle very near to their homes.

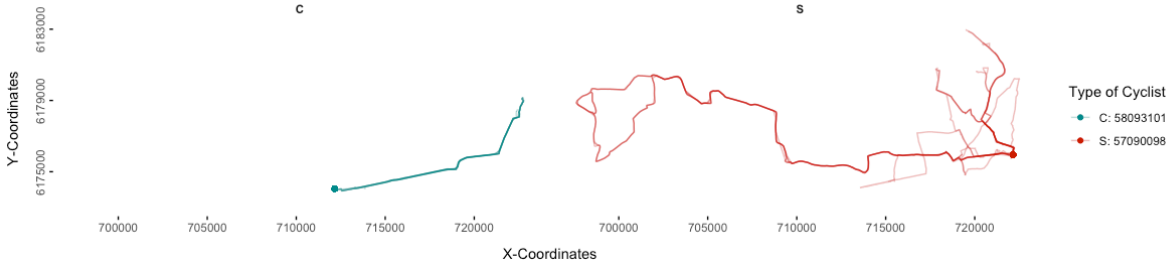

Figure 14: Extract of two respondents to visualise the corresponding trips to the dwelling.

The spatial accuracy of the dwelling depends on the distance of all contributing points. The nearer all the points are together the smaller the area in which the home of a respondent can potentially be found. Figure 15 shows that for the dwellings the distance of most contributing points are between 0 and 2 kilometres, with more being at the lower end of the spectrum. Even though 2 kilometres might seem like a large distance, for our very simple approach it is quite a good result. It also proves that the data is not linearly correlated, i.e. a higher number of points does not lead to a higher mean distance to dwelling, and is therefore more robust.

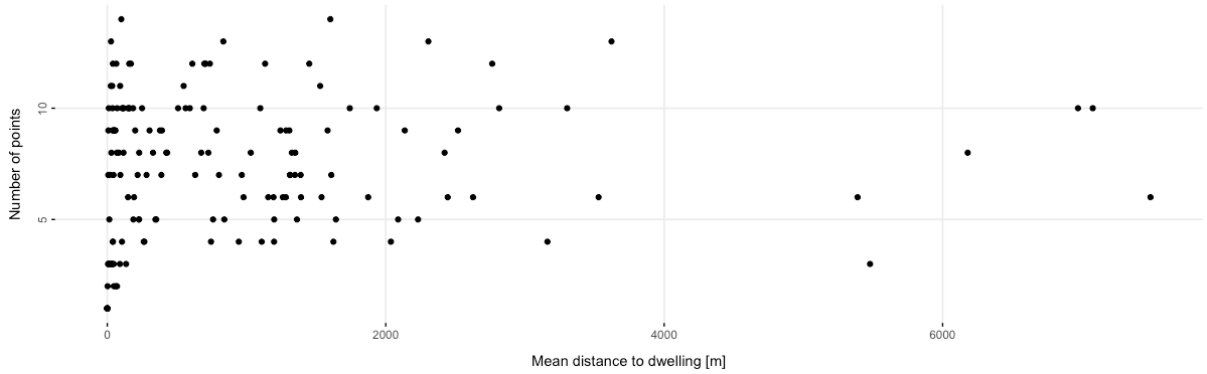

Figure 15: Accuracy of dwellings.

4 Discussion

The answer to "Using two methods, how can trips of individual cyclists be categorised into commute and stray trips" (RQ.1) is that it works very well with two different methods. The more local constraint of overlapping trips in the buffer approach shows us where commuting trips happen. The more global approach of start and end destinations using the cluster approach lead us to find a new type of trip: the round trip. Both methods have their merit and a majority of trip are categorised in the same class in either method.

The dwellings of different types of cyclists can be found using their first and last GPS recording of the day to a reasonable high degree of spatial certainty (RQ.2). The home locations do not differ between more commuting or more straying cyclists.

Our results depend highly on the methods we choose (Laube 2019). At the same time, the methods depend on the way we understand the world. An example of this is RQ.1 where the phrasing of the assignment leaves room for interpretation and two methods were applied to answer the research question. The choices of methods and thresholds have a big influence on our results. Two examples of this are explained in the following section.

The categorisation of trips into stray or commute in the buffer method depends on how much it overlaps with the other trips. For respondents with more than two trips, all overlaps of one trip have to be aggregated. The aggregation can be based on the mean or the median of all or just a number of buffer comparisons. We compared two variants: calculating commuting proportion of each trip using the mean and median of all buffer comparisons of this specific trip which have overlaps of more than 70%. All trips with no overlaps of more than 70% were not considered to be commutes and therefore received 0 commuting proportions. Interestingly, comparing the two methods shows that trips with high commuting proportions do not differ a lot between the methods. Using the mean over the median allows for all trips to have a value, however we run into the problem that a person may commute on different routes. Even if they use only two routes all the time, the comparison of two trips on those two routes will result in zero overlap, which might drag the average down. This is why, in the end, we went with the method using the median and chose not to include all trip comparisons.

The second example is the very high influence of prior modelling choices for results further on. The location of the dwelling is dependent on the points' cluster information. This

cluster information was used for RQ.1 to determine start and end destinations and was at first not meant to be used to find the homes of the respondents. However, it turned out to be quite useful for this task as well. Would we have looked for clustering in the first and last points per day of each respondent, we might have chosen different parameters in the DBSCAN or another clustering algorithm altogether. The result of the home locations therefore depended on an earlier modelling choice.

We need to be aware as well of the highly private data at our disposal here. It is not at all difficult to identify respondents based on their movement behaviour over three weeks, having an idea where they live and work. Obfuscation of their homes, for example, might be a sensible choice for safeguarding their privacy in a more public study than this one.

References

Banister, D. (2008). The sustainable mobility paradigm. *Transport policy*, 15(2), 73-80.

Collier, M. J., Nedović-Budić, Z., Aerts, J., Connop, S., Foley, D., Foley, K., ... & Verburg, P. (2013). Transitioning to resilience and sustainability in urban communities. *Cities*, 32, 21–28.

Ministry of Foreign Affairs of Denmark (n.d.). A nation of cylcists. Retrieved from https://denmark.dk/people-and-culture/biking.

Laube, P., & Purves, R. S. (2011). How fast is a cow? Cross-scale analysis of movement data. *Transactions in GIS, 15(3),* 401-418.

Laube, P. (2014). Computational Movement Analysis. Wädenswil: Springer.

Laube, P. (2017). Representation, trajectories. *The International Encyclopedia of Geography,* 1–13.

Laube, P. (2019). T1 – Modelling matters! Lecture slides course „Patterns & Trends in Environmental Data – GEO 880 Computational Movement Analysis.

Toohey, K., & Duckham, M. (2015). Trajectory similarity measures. *Sigspatial Special*, *7*(1), 43-50.

Tripadvisor (2019). Die besten Sehenswürdigkeiten in Kopenhagen. < *https://www.tripadvisor.ch/Attractions-g189541-Activities-Copenhagen_Zealand.html*> [Access: 20.06.19].

Appendix A: Fixing the TripIDs

```
Options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
  for (package in packages_list){
    if (!require(package, character.only = T)) {
      install.packages(package)
    }
    library(package, character.only = T)
  }
}

packages_checker(
  c('plyr',
    'tidyverse',
    'data.table',
    'lubridate')
)
```

```
#### FIXING THE TRIP ID
read_csv(
  "data/Bikeability_CPH.txt",
  col_types = cols(
    OBJECTID_1 = col_double(),
    RespID = col_character(),
    TripID = col_character(),
    POINT_X = col_double(),
    POINT_Y = col_double(),
    DT_String = col_datetime("%d-%m-%Y %H:%M:%S"))) %>%
  rename(x = POINT_X,
         y = POINT_Y,
         datetime = DT_String) %>%
  na.omit() %>%
  group_by(RespID) %>%
  arrange(datetime, .by_group = TRUE) %>%
  mutate(
    FixTimediffMin = as.numeric(difftime(datetime, lag(datetime,1, default = NA), units = "mins"), units = "mins"),
    FixTimediffMin = ifelse(is.na(FixTimediffMin), 120, FixTimediffMin),
    FixDistMeter = sqrt((lag(x,1, default = NA)-x)^2 + (lag(y,1, default = NA)-y)^2),
    FixDistMeter = ifelse(is.na(FixDistMeter), 0, FixDistMeter),
    FixSpeedKmh = (FixDistMeter / 1000) / (FixTimediffMin / 60),
    FixSpeedKmh = ifelse(is.na(FixSpeedKmh), 0, FixSpeedKmh),
    trip_start = FixTimediffMin > 30 | FixDistMeter > 1500,
    TripID_new = paste(RespID,as.character(cumsum(trip_start)), sep = "_")) %>%
  ungroup() %>%
  dplyr::select(-c(trip_start, OBJECTID_1)) %>%
  rename(TripID_old = TripID, TripID = TripID_new) %>%
  group_by(TripID) %>%
  mutate(weekday = wday(datetime, week_start = 1, label = T),
         FixTimediffMin = as.numeric(difftime(datetime, lag(datetime,1, default = NA), units = "mins"), units = "mins"),
         FixTimediffMin = ifelse(is.na(FixTimediffMin), 0, FixTimediffMin),
         FixDistMeter = sqrt((lag(x,1, default = NA)-x)^2 + (lag(y,1, default = NA)-y)^2),
         FixDistMeter = ifelse(is.na(FixDistMeter), 0, FixDistMeter),
         FixSpeedKmh = (FixDistMeter / 1000) / (FixTimediffMin / 60),
         #FixSpeedKmh = ifelse(FixSpeedKmh > 200, lead(FixSpeedKmh, 1, default = 0), FixSpeedKmh),
         FixSpeedKmh = ifelse(is.na(FixSpeedKmh), 0, FixSpeedKmh),
         TripDistMeter = sum(FixDistMeter),
         TripDurationMin = sum(FixTimediffMin),
         TripSpeedKmh = (TripDistMeter / 1000) / (TripDurationMin / 60),
         TripFixCount = n()) %>%
  ungroup() %>%
  filter(TripFixCount >= 10,
         FixSpeedKmh < 120,
         TripDistMeter > 0,
         TripDurationMin > 5) %>%
  write_delim(path = "data/Bikeability_CPH_new3.txt", delim = ",")
```

Appendix B: Data exploration

```
options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
  for (package in packages_list){
    if (!require(package, character.only = T)) {
      install.packages(package)
    }
    library(package, character.only = T)
  }
}

packages_checker(
```

```
  c('ggrepel',
    'dplyr',
    'tidyverse',
    'data.table',
    'lubridate',
    'ggplot2',
    'ggsn',
    'sf',
    'tmap')
)
```

```
bike_cph_raw3 <- read_csv(
  "data/Bikeability_CPH_new3.txt",
  col_types = cols(
    RespID = col_character(),
    TripID_old = col_character(),
    x = col_double(),
    y = col_double(),
    datetime = col_datetime(format = ""),
    TripID = col_character(),
    weekday = col_character(),
    FixTimediffMin = col_double(),
    FixDistMeter = col_double(),
    FixSpeedKmh = col_double(),
    TripDistMeter = col_double(),
    TripDurationMin = col_double(),
    TripSpeedKmh = col_double(),
    TripFixCount = col_integer()
  ))
```

```
############################ DATA DISTRIBUTION, STATISTICS ##########
# show speed / time diff / dist diff
bike_cph_raw3 %>%
  filter(RespID == "58093101") %>%
  group_by(TripID) %>%
  ggplot() +
  geom_hline(aes(yintercept = median(FixTimediffMin * 60), group = TripID), color = "darkred", alpha = 0.5) +
  geom_hline(aes(yintercept = median(FixDistMeter / 10), group = TripID), color = "darkblue", alpha = 0.5) +
  geom_point(aes(datetime, FixTimediffMin * 60), col = "red", shape = 20, alpha = 0.5) +
  geom_point(aes(datetime, FixDistMeter / 10), col = "blue", shape = 20, alpha = 0.5) +
  geom_path(aes(datetime, FixSpeedKmh)) +
  ylim(c(0, 100)) +
  facet_wrap(~ TripID, scales = "free")

# get some statistics
length(unique(bike_cph_raw3$RespID))
length(unique(bike_cph_raw3$TripID))
length(unique(bike_cph_raw3$TripID_old))
summary(bike_cph_raw3)

# get statistics of Trip count per Resp
RespID_TripID_LUT <- bike_cph_raw3 %>%
  group_by(RespID) %>%
  dplyr::mutate(TripCount = n_distinct(TripID))

# get statistics from park driver (longest trip (distance))
bike_cph_raw3 %>%
  filter(TripID == "57090098_21") %>%
  summary()
```

```
################################### SPECIAL CYCLISTS MAPS ##################
# a typical commuter
# show why at least 10 fix point needed --> one very short and random trip by this RespID
bike_cph_raw3 %>%
  filter(RespID == "58093101",
         TripID  %in%  c("58093101_15","58093101_16",  "58093101_19",  "58093101_2",  "58093101_20",  "58093101_3",  "58093101_4",
"58093101_5" )) %>%
  group_by(TripID) %>%
  ggplot(aes(x,y)) +
  geom_path(col = "orange") +
  geom_point(shape = ".") +
  theme(legend.position = "none") +
  facet_wrap(~ TripID, scales = "free", nrow = 2) +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        plot.margin = margin(t = 0,r = 20,b=5,l=10),
        strip.text = element_text(face = "bold")) +
  labs(title = "",
```

```
      x = "X-Coordinates",
      y = "Y-Coordinates")

# plot typical round trip (long distance driver)
bike_cph_raw3 %>%
  filter(TripID == "57090098_21") %>%
  ggplot(aes(x,y)) +
  geom_path(col = "orange") +
  geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none") +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0))) +
  labs(title = "",
       x = "X-Coordinates",
       y = "Y-Coordinates") +
  scale_y_continuous(limits = c(6174000, 6181000), breaks = c(6175000, 6177000, 6179000, 6181000))

# look at fastest rider, etc.
bike_cph_raw3 %>%
  group_by(RespID) %>%
  summarise(TripDistMeter_mean = mean(TripDistMeter),
            TripDurationMin_mean = mean(TripDurationMin),
            TripSpeedKmh_mean = mean(TripSpeedKmh),
            TripCount = n_distinct(TripID)) %>%
  filter(TripCount >= 5) %>%
  arrange(-TripDistMeter_mean)
# problem with mean > outliers!! (RespID: 57090098)

# average fastest rider
bike_cph_raw3 %>%
  filter(RespID == "57282430") %>%
  group_by(TripID) %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = TripID)) +
  #geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none")

# average longest rider (time)
bike_cph_raw3 %>%
  filter(RespID == "57036630") %>%
  group_by(TripID) %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = TripID)) +
  #geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none")

# average furtherst rider (distance)
bike_cph_raw3 %>%
  filter(RespID == "57036630") %>%
  group_by(TripID) %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = datetime)) +
  #geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none")
```

```
################################### MAPS ############################
# overview all trips NEW
bike_cph_raw3 %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(col = "black", alpha = 0.1, size = 0.6) +
  coord_equal(ratio = 1) +
  theme(legend.position = "none") +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0))) +
  labs(title = "",
       x = "X-Coordinates",
       y = "Y-Coordinates") +
  ylim(c(6165000, 6190000))

# overview all trips NEW
```

```
bike_cph_raw3 %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = TripID)) +
  geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none")

# overview all trips OLD
bike_cph_raw3 %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = TripID_old)) +
  geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none")

# look at city center: trips
bike_cph_raw3 %>%
  mutate(x = round_any(x, 5),
         y = round_any(y, 5))  %>%
  group_by(TripID) %>%
  ggplot(aes(x,y)) +
  geom_path(aes(col = RespID)) +
  #geom_point(shape = ".") +
  coord_equal(ratio = 1) +
  theme(legend.position = "none") +
  xlim(c(720000,730000)) +
  ylim(c(6170000, 6180000))

tmap_mode("view")
bike_cph_raw3_sf <- bike_cph_raw3 %>%
  filter(RespID == "57014007") %>%
  st_as_sf(coords = c("x", "y"), crs = 25832) %>%
  tm_shape() +
  tm_symbols(col = "black",alpha = 0.1, border.col = NULL, size = 0.2) +
  tm_legend(bg.color = "white")

################################## HEAT MAPS ####################
#heat map for all trips over 3 weeks
#get dates ready
start_date <- bike_cph_raw3 %>%
  mutate(datetime_floored = floor_date(datetime - hours(4), unit = "days")) %>%
  arrange(datetime_floored) %>%
  summarise(start = first(datetime_floored)) %>%
  pull(start)

end_date <- bike_cph_raw3 %>%
  mutate(datetime_floored = floor_date(datetime - hours(4), unit = "days")) %>%
  arrange(datetime_floored) %>%
  summarise(end = last(datetime_floored)) %>%
  pull(end)

date_vector = seq(start_date, end_date,  by="days")

bike_cph_raw3 %>%
  mutate(datetime_floored = floor_date(datetime - hours(4), unit = "days")) %>%
  group_by(RespID, TripID) %>%
  summarise(hour = median(hour(datetime - hours(4))),
            datetime_floored = first(datetime_floored)) %>%
  ungroup() %>%
  group_by(datetime_floored, hour) %>%
  summarise(n = n()) %>%
  complete(hour = as.integer(full_seq(0:23, 1)), fill = list(n = 0)) %>%
  ungroup() %>%
  group_by(datetime_floored) %>%
  mutate(n = n / sum(n)) %>%
  ungroup() %>%
  mutate(weeknumber = isoweek(datetime_floored)) %>%
  ggplot(aes(hour, datetime_floored)) +
  geom_tile(aes(fill = n), colour = "white") +
  scale_fill_gradientn(name = "Bike trips, normalized per day [%]", colours= c("#4575b4","orange","#FF0000","#660000")) +
  theme_minimal() +
  theme(legend.key.width = unit(1, "cm"),
        panel.grid = element_blank(),
        axis.ticks.y= element_blank(),
        legend.position = "bottom") +
  scale_y_datetime(breaks =  date_vector) +
  scale_x_continuous(expand = c(0,0),breaks = c(0:23),labels = c(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,1,2,3)) +
  labs(title = "",
       x = "Hour of day",
       y = "Date") +
  facet_grid(weeknumber ~., scales = "free")
```

```
# heat map for trip fixes
bike_cph_raw3 %>%
  mutate(x = round(x, -2),
         y = round(y, -2)) %>%
  group_by(x,y) %>%
  summarise(n = n()) %>%
  #complete(x = seq(710000,730000, 1000),y = seq(6170000, 6190000, 1000), fill = list(n = 0)) %>%
  ggplot(aes(x,y)) +
  geom_tile(aes(fill = n)) +
  xlim(c(710000,730000)) +
  ylim(c(6170000, 6190000)) #+
  #theme(legend.position = "none")
```

```
################################### TRIP DISTRIBUTION OVER TIME ##################
# location: per weekday (all 3 weeks together)
bike_cph_raw3 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So"))) %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = RespID)) +
  coord_equal(ratio = 1) +
  facet_grid(~ weekday) +
  theme(legend.position = "none") #+

# location: per weekday and daytime (distribution over daytime)
bike_cph_raw3 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         time_of_day = case_when(hour(datetime) < 6  ~ "0-6",
                                 hour(datetime) < 12 ~ "6-12",
                                 hour(datetime) < 18 ~ "12-18",
                                 hour(datetime) < 24 ~ "18-24")) %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = RespID)) +
  coord_equal(ratio = 1) +
  facet_grid(time_of_day ~ weekday) +
  theme(legend.position = "none")

# speed and distance: per weekday and daytime
bike_cph_raw3 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         time_of_day = case_when(hour(datetime) < 6  ~ "0-6",
                                 hour(datetime) < 12 ~ "6-12",
                                 hour(datetime) < 18 ~ "12-18",
                                 hour(datetime) < 24 ~ "18-24")) %>%
  ggplot(aes(TripDistMeter,TripSpeedKmh)) +
  geom_point(aes(col = RespID)) +
  #coord_equal(ratio = 1) +
  facet_grid(time_of_day ~ weekday) +
  theme(legend.position = "none")
```

Appendix C: Trip similarity, similarity measurement DTW

```
## Task 1: Calculate Similarity of Trips per respondent
#measure similarity: create matrix
trips_matrix <- bike_cph_raw3 %>%
  ungroup() %>%
  mutate(RespID = as.double(RespID)) %>%
  split(.$TripID_new) %>%
    map(function(x){
      x %>%
        select(x, y) %>%
        as.matrix()
      })
trips_matrix

# count number of TripID_new of bikeability_selected
count_TripID_new <- n_distinct(bike_cph_raw3 $TripID_new)
count_TripID_new

# calculate DTW for dataframe
trips_measures <- data.frame()
start.time <- Sys.time()
for (i in 1:count_TripID_new){
  temp_result <- imap_dfr(trips_matrix,
                          ~data_frame(
                            #traj = .y,
                            TripName = DTW(
                              .x,trips_matrix[[i]]
                              ),
                            )
                          )
 if(i==1){
```

```
  trips_measures <- temp_result
 } else {
  trips_measures <- cbind(trips_measures,temp_result)
  }
}

row.names(trips_measures) <- unique(bike_cph_raw3 $TripID_new)
colnames(trips_measures) <- unique(bike_cph_raw3 $TripID_new)

# calculate DTW for inverse dataframe to eliminate problem of trajectory direction
trips_measures_reverse <- data.frame()
for (i in 1:count_TripID_new){
  temp_result <- imap_dfr(trips_matrix,
                          ~data_frame(
                            #traj = .y,
                            TripName = DTW(
                              .x,trips_matrix[[i]][nrow(trips_matrix[[i]]):1, ]
                              ),
                            )
                          )
 if(i==1){
   trips_measures_reverse <- temp_result
 } else {
     trips_measures_reverse <- cbind(trips_measures_reverse,temp_result)
  }
}

row.names(trips_measures_reverse) <- unique(bikeability_select$TripID_new)
colnames(trips_measures_reverse) <- unique(bikeability_select$TripID_new)

# take the smaller value of the original and inversed matrix
trips_similarity <- pmin(trips_measures, trips_measures_reverse)
trips_similarity %>%
  mutate(similarity_mean = rowMeans(trips_similarity))

dtw_summary <- summary(trips_similarity)

end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

kmeans(trips_similarity, centers = 3) #%>% #nrow(trips_similarity)-1)

table(kmeans$cluster)
```

Appendix D: Trip similarity, buffer approach

```
options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
  for (package in packages_list){
    if (!require(package, character.only = T)) {
      install.packages(package)
    }
    library(package, character.only = T)
  }
}

packages_checker(
  c('ggrepel',
    'dplyr',
    'tidyverse',
    'data.table',
    'lubridate',
    'ggplot2',
    'sf',
    'trip')
)


bike_cph_raw3 <- read_csv(
  "data/Bikeability_CPH_new3.txt",
  col_types = cols(
    RespID = col_character(),
    TripID_old = col_character(),
    x = col_double(),
    y = col_double(),
    datetime = col_datetime(format = ""),
    TripID = col_character(),
    weekday = col_character(),
    FixTimediffMin = col_double(),
    FixDistMeter = col_double(),
```

```
  FixSpeedKmh = col_double(),
  TripDistMeter = col_double(),
  TripDurationMin = col_double(),
  TripSpeedKmh = col_double(),
  TripFixCount = col_integer()
))

################################### TYPE OF TRIPS PER RESPONDENT ##########################
################################### buffer approach ########################################################
################################### PREPARE DATA ##########################################
# make sf through trip package
bike_cph_trip_sf <- bike_cph_raw3 %>%
 #filter(RespID == "58093101") %>% # filter for Resp example for report
 group_by(RespID) %>%
 mutate(TripCount = n_distinct(TripID)) %>%
 ungroup() %>%
 dplyr::select(x,y,datetime,TripID) %>%
 trip() %>%
 as("sf")

# add crs
st_crs(bike_cph_trip_sf) <- 25832

# get info on RespID and their respective TripID
RespID_TripID_LUT <- bike_cph_raw3 %>%
 #filter(RespID == "58093101") %>% # filter for Resp example for report
 group_by(RespID) %>%
 distinct(TripID)

# extract only RespID
RespID_LUT <- RespID_TripID_LUT %>%
 #filter(RespID == "58093101") %>% # filter for Resp example for report
 select(RespID) %>%
 distinct(RespID) %>%
 pull(RespID)

# RespID_LUT <- c("58193858") # for testing ,"57014403


############## CALCULATE BUFFER FOR EACH TRIP AND COMPARE OVERLAP WITH TRIPS (LINES) #########
TripID_vector <- c()
Trip_commute_B <- c()
Trip_commute_B_perc_mean <- c()


# first loop: go through all RespIDs
# more than 70% spatial overlap --> commute otherwise stray
threshold <- 0.7

for (RespID_index in RespID_LUT){
 TripID_per_RespID <- RespID_TripID_LUT[RespID_TripID_LUT$RespID == RespID_index,"TripID"] %>% pull()

 Trip_array <- array(NA, dim = c(length(TripID_per_RespID), length(TripID_per_RespID)))
 #print(length(TripID_per_RespID))
 array_x_index = 1

 # second loop: go through all TripIDs of that RespID
 # buffer the selected TripID
 for (TripID_index in TripID_per_RespID){
  #print(TripID_index)

  Trip_data <- bike_cph_trip_sf[bike_cph_trip_sf$tripID == TripID_index,]
  Trip_buffered <- st_buffer(st_zm(Trip_data), dist = 50) # 50 map unit buffer (= m)

  array_y_index = 1

  # third loop: compare this one TripID to all other TripIDs (including itself)
  # interesect buffered line (which is a polygon now) with each TripID
  # calculate ratio of line within polygon and the length of the shorter line
  # write result to array
  for(Other_TripIDs_index in TripID_per_RespID) {
   Trip_data_other <- bike_cph_trip_sf[bike_cph_trip_sf$tripID == Other_TripIDs_index,] %>% st_zm()
   Trip_intersection <- st_intersection(Trip_data_other, Trip_buffered)

   # what should be the reference?
   #reference_length <- min(st_length(Trip_data), st_length(Trip_data_other)) # always take shorter path as reference
   reference_length <- st_length(Trip_data_other)
   intersection_length <- st_length(Trip_intersection)
   result <- as.numeric((intersection_length / reference_length)[1])
   result = ifelse(is.na(result), 0, result)

   Trip_array[array_x_index,array_y_index] <- result
```

```
    # #plot buffer and line to get an idea if it works
    # if(array_x_index == 1){ # && array_y_index == 1
    # options(sf_max.plot=1)
    # plot(Trip_buffered, main = "", col = "white", reset = FALSE, axes = TRUE)
    # title(main = as.character(result))
    # plot(Trip_data_other, main = "", col = "red", add = T)
    # }

    array_y_index = array_y_index + 1
   }
   array_x_index = array_x_index + 1
 }

 # calculate "similarity" based on how much of a Trip lies within the respective polygon
 # here 70% are used
 for(array_col in 1:length(TripID_per_RespID)){
   values <- Trip_array[,array_col]
   values_filtered <- values[-array_col] # get rid of self intersection (1)


   TripID_vector <- c(TripID_vector, TripID_per_RespID[array_col])
   Trip_commute_B <- c(Trip_commute_B, sum(values_filtered > threshold))
   Trip_commute_B_perc_mean <- c(Trip_commute_B_perc_mean, ifelse(sum(values_filtered > threshold) >= 2,
                                                                  mean(values_filtered[values_filtered > threshold]),
                                                                  0))
   #Trip_commute_B_perc_mean_all <- c(Trip_commute_B_perc_mean_all, mean(values_filtered))
 }
}

result_table <- cbind.data.frame(TripID_vector, Trip_commute_B, Trip_commute_B_perc_mean)
summary(result_table)
```

```
################################### RESULTS ###################################################################
# Join results from overlap to LUT --> percentage of commuter (percentag of lines laying in the buffer) --> define type of trip (Commuter (C) or stray)
RespID_TripID_LUT <- RespID_TripID_LUT %>%
 left_join(result_table, by = c("TripID" = "TripID_vector")) %>%
 group_by(RespID) %>%
 dplyr::mutate(Trip_sum = n_distinct(TripID),
        Type_Trip_B = ifelse(Trip_commute_B_perc_mean > threshold, "C", "S")) %>%
 write_delim(path = "data/Bikeability_CPH_Type_Trip_B_info.txt", delim = ",")

# save extended table (join Type_Trip_B information of each trip to the original data set)
bike_cph_raw3 %>%
 left_join(subset(RespID_TripID_LUT, select = c(TripID, Type_Trip_B, Trip_commute_B, Trip_sum, Trip_commute_B_perc_mean)), by = "TripID") %>%
 write_delim(path = "data/Bikeability_CPH_new3_Type_Trip_B.txt", delim = ",")
```

Appendix E: Analysing buffer approach

```
options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
 for (package in packages_list){
   if (!require(package, character.only = T)) {
     install.packages(package)
   }
   library(package, character.only = T)
 }
}

packages_checker(
 c('ggrepel',
   'dplyr',
   'tidyverse',
   'data.table',
   'lubridate',
   'ggplot2',
   'sf',
   'trip')
)

bike_cph_raw4 <- read_csv(
 "data/Bikeability_CPH_new3_Type_Trip_B.txt",
 col_types = cols(
   RespID = col_character(),
   TripID_old = col_character(),
   x = col_double(),
   y = col_double(),
   datetime = col_datetime(format = ""),
   TripID = col_character(),
```

```
  weekday = col_character(),
  FixTimediffMin = col_double(),
  FixDistMeter = col_double(),
  FixSpeedKmh = col_double(),
  TripDistMeter = col_double(),
  TripDurationMin = col_double(),
  TripSpeedKmh = col_double(),
  TripFixCount = col_integer(),
  Type_Trip_B =col_character(),
  Trip_commute_B = col_integer(),
  Trip_sum = col_integer(),
  Trip_commute_B_perc_mean = col_double()
))

################################### DISTRIBUTION STATISTICS ######################################
# get number of trips by Type of trip (C, S) and per week and weekday
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
bike_cph_raw4 %>%
 mutate(datetime_floor = floor_date(datetime, unit = "day")) %>%
 group_by(datetime_floor) %>%
 mutate(week_number = isoweek(first(datetime))) %>%
 ungroup() %>%
 mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
       hour_of_day = as.numeric(hour(datetime)),
       time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                               hour(datetime) < 12 ~ "6-10, morning commute",
                               hour(datetime) < 15 ~ "10-15, noon",
                               hour(datetime) < 19 ~ "15-19, afternoon commute"),
       time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
 group_by(week_number, weekday, Type_Trip_B, hour_of_day) %>%
 summarise(n = n_distinct(TripID)) %>%
 ungroup() %>%
 group_by(Type_Trip_B) %>%
 ggplot(aes(hour_of_day,n)) +
 geom_line(aes(col = Type_Trip_B), size = 0.5) +
 facet_grid(week_number ~ weekday) +
 scale_color_manual(values = cols) +
 theme_minimal() +
 theme(axis.text.y = element_text(angle = 90, hjust = 0.5),
       axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
       axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
       strip.text = element_text(face = "bold")) +
 labs(title = "",
      x = "Hour of day" ,
      y = "Number of trips",
      col = "Type of Trip")

# histogram with count TripID's per hour (over all 3 study weeks)
bike_cph_raw4 %>%
 group_by(RespID, TripID) %>%
 summarise(datetime = first(datetime)) %>%
 ungroup() %>%
 mutate(hour_of_day = hour(datetime)) %>%
 group_by(hour_of_day) %>%
 mutate(n = n()) %>%
 ungroup() %>%
 ggplot(aes(hour_of_day)) +
 geom_histogram(bins = 100)

# speed and distance: per weekday and daytime
bike_cph_raw4 %>%
 mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
       time_of_day = case_when(hour(datetime) < 6  ~ "0-6",
                               hour(datetime) < 12 ~ "6-12",
                               hour(datetime) < 18 ~ "12-18",
                               hour(datetime) < 24 ~ "18-24")) %>%
 ggplot(aes(TripDistMeter,TripSpeedKmh)) +
 geom_point(aes(col = Type_Trip_B)) #+
 #coord_equal(ratio = 1) +
 #facet_grid(time_of_day ~ weekday) #+
 #theme(legend.position = "none")

# get some interesting numbers for S and C trips for report
summary_C <- bike_cph_raw4 %>%
 filter(Type_Trip_B == "C")

length(unique(summary_C$TripID))
summary(summary_C)

summary_S <- bike_cph_raw4 %>%
 filter(Type_Trip_B == "S")
```

```
length(unique(summary_S$TripID))
summary(summary_S)
```

```
################################## DISTRIBUTION MAPS ############################
# show number of C and S trips per weekday
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
bike_cph_raw4 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         hour_of_day = as.numeric(hour(datetime)),
         time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                                 hour(datetime) < 12 ~ "6-10, morning commute",
                                 hour(datetime) < 15 ~ "10-15, noon",
                                 hour(datetime) < 19 ~ "15-19, afternoon commute"),
         time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
  group_by(Type_Trip_B) %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = Type_Trip_B), alpha = 0.1, size = 0.5) +
  facet_grid(Type_Trip_B~weekday) +
  coord_equal(ratio = 1) +
  scale_color_manual(values = cols) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        #axis.ticks = element_line(),
        #axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.text = element_blank(),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        strip.text = element_text(face = "bold"),
        strip.text.y = element_text(angle = 0, hjust = 0.5),
        strip.background = element_rect(colour = "grey", fill = "grey")) +
  labs(title = "",
       x = "X-Coordinate",
       y = "Y-Coordinate",
       col = "Type of Trip") +
  ylim(c(6165000, 6190000)) +
  theme(legend.position = "none")

# show number of C and S trips per daytime
bike_cph_raw4 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         hour_of_day = as.numeric(hour(datetime)),
         time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                                 hour(datetime) < 12 ~ "6-10, morning commute",
                                 hour(datetime) < 15 ~ "10-15, noon",
                                 hour(datetime) < 19 ~ "15-19, afternoon commute"),
         time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
  group_by(Type_Trip_B) %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = Type_Trip_B), alpha = 0.1, size = 0.5) +
  facet_grid(Type_Trip_B~time_of_day) +
  coord_equal(ratio = 1) +
  scale_color_manual(values = cols)
#theme(legend.position = "none")
```

Appendix F: Trip similarity, cluster approach incl. analyse

```
options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
  for (package in packages_list){
    if (!require(package, character.only = T)) {
      install.packages(package)
    }
    library(package, character.only = T)
  }
}

packages_checker(
  c('ggrepel',
    'dplyr',
    'tidyverse',
    'data.table',
    'lubridate',
    'ggplot2',
    'sf',
    'trip',
    'dbscan')
)
```

```
bike_cph_raw5 <- read_csv(
 "data/Bikeability_CPH_new3_Type_Trip_B.txt",
 col_types = cols(
  RespID = col_character(),
  TripID_old = col_character(),
  x = col_double(),
  y = col_double(),
  datetime = col_datetime(format = ""),
  TripID = col_character(),
  weekday = col_character(),
  FixTimediffMin = col_double(),
  FixDistMeter = col_double(),
  FixSpeedKmh = col_double(),
  TripDistMeter = col_double(),
  TripDurationMin = col_double(),
  TripSpeedKmh = col_double(),
  TripFixCount = col_integer(),
  Type_Trip_B =col_character(),
  Trip_commute_B = col_integer(),
  Trip_sum = col_integer(),
  Trip_commute_B_perc_mean = col_double()
 ))

################################## TYPE OF TRIPS PER RESPONDENT ########################
################################## point clustering approach ######################
# get info on RespID and their respective TripID
RespID_TripID_LUT <- bike_cph_raw5 %>%
 group_by(RespID) %>%
 distinct(TripID)

# extract only RespID
RespID_LUT <- RespID_TripID_LUT %>%
 select(RespID) %>%
 distinct(RespID) %>%
 pull(RespID)

#RespID_LUT <- c("58218300") # for testing ,"57014403

first_last_entries_per_Trip <- bike_cph_raw5 %>%
 group_by(TripID) %>%
 slice(c(1, n())) %>%
 ungroup()

# get first and last point of each TripID
TripID_Cluster <- c()
for (RespID_index in RespID_LUT){
 temp_cluster_data <- first_last_entries_per_Trip[first_last_entries_per_Trip$RespID == RespID_index, ] %>%
  select(x,y) %>%
  dbscan::dbscan(eps = 500, minPts = 3)

 TripID_Cluster <- c(TripID_Cluster, temp_cluster_data$cluster)

}
first_last_entries_per_Trip <- first_last_entries_per_Trip %>%
 mutate(Cluster =TripID_Cluster) %>%
 group_by(TripID) %>%
 summarise(Cluster_Start = first(Cluster, order_by = TripID),
           Cluster_End = last(Cluster, order_by = TripID)) %>%
 group_by(TripID) %>%
 mutate(Type_Trip_P = ifelse(Cluster_Start == 0 || Cluster_End == 0 , "S", ifelse(Cluster_Start != Cluster_End, "C", "R")),
        Type_Trip_P_grouped = ifelse(Cluster_Start == 0 || Cluster_End == 0 || Cluster_Start == Cluster_End, "S", "C"))

# join information of comuuter, rounder, stray to original dataset by left join
bike_cph_raw5 <- left_join(bike_cph_raw5, subset(first_last_entries_per_Trip, select = c(Type_Trip_P, Type_Trip_P_grouped, TripID,
Cluster_Start, Cluster_End), by = "TripID"))

write_csv(bike_cph_raw5, "data/Bikeability_CPH_5_Type_Trip_P.csv")

# export bike_cph_raw5 for KDE in ArcMap
bike_cph_raw5 %>%
 filter(Type_Trip_P == "C") %>%
 write_csv("data/Bikeability_CPH_5_Type_Trip_P_C.csv")
bike_cph_raw5 %>%
 filter(Type_Trip_P == "R") %>%
 write_csv("data/Bikeability_CPH_5_Type_Trip_P_R.csv")
bike_cph_raw5 %>%
 filter(Type_Trip_P == "S") %>%
 write_csv("data/Bikeability_CPH_5_Type_Trip_P_S.csv")

################################## DISTRIBUTION STATISTICS ##########################################
# get number of trips by Type of trip (C, R, S) and per week and weekday
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
```

```
bike_cph_raw5 %>%
  mutate(datetime_floor = floor_date(datetime, unit = "day")) %>%
  group_by(datetime_floor) %>%
  mutate(week_number = isoweek(first(datetime))) %>%
  ungroup() %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         hour_of_day = as.numeric(hour(datetime)),
         time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                                 hour(datetime) < 12 ~ "6-10, morning commute",
                                 hour(datetime) < 15 ~ "10-15, noon",
                                 hour(datetime) < 19 ~ "15-19, afternoon commute"),
         time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
  group_by(week_number, weekday, Type_Trip_P, hour_of_day) %>%
  summarise(n = n_distinct(TripID)) %>%
  ungroup() %>%
  group_by(Type_Trip_P) %>%
  ggplot(aes(hour_of_day,n)) +
  geom_line(aes(col = Type_Trip_P), size = 0.5) +
  facet_grid(week_number ~ weekday) +
  scale_color_manual(values = cols) +
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        strip.text = element_text(face = "bold")) +
  labs(title = "",
       x = "Hour of day" ,
       y = "Number of trips" ,
       col = "Type of Trip")

# speed and distance: per weekday and daytime
bike_cph_raw5 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         time_of_day = case_when(hour(datetime) < 6  ~ "0-6",
                                 hour(datetime) < 12 ~ "6-12",
                                 hour(datetime) < 18 ~ "12-18",
                                 hour(datetime) < 24 ~ "18-24")) %>%
  ggplot(aes(TripDistMeter,TripSpeedKmh)) +
  geom_point(aes(col = Type_Trip_P)) #+
  #coord_equal(ratio = 1) +
  #facet_grid(time_of_day ~ weekday) #+
#theme(legend.position = "none")

# get some interesting numbers for S, R and C trips for report
summary_C <- bike_cph_raw5 %>%
  filter(Type_Trip_P == "C")

length(unique(summary_C$TripID))
summary(summary_C)

summary_S <- bike_cph_raw5 %>%
  filter(Type_Trip_P == "S")

length(unique(summary_S$TripID))
summary(summary_S)

summary_R <- bike_cph_raw5 %>%
  filter(Type_Trip_P == "R")

length(unique(summary_R$TripID))
summary(summary_R)
```

```
################################### DISTRIBUTION MAPS ##################
# show number of C and S trips per weekday
# visualize all trips colored by Type of trip (C, R, S)
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
bike_cph_raw5 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         hour_of_day = as.numeric(hour(datetime)),
         time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                                 hour(datetime) < 12 ~ "6-10, morning commute",
                                 hour(datetime) < 15 ~ "10-15, noon",
                                 hour(datetime) < 19 ~ "15-19, afternoon commute"),
         time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
  #group_by(RespID) %>%
  #mutate(Resp_is_commute = (sum(Trip_is_commmute) / n()) > 0.8) %>%
  group_by(TripID) %>%
  #ungroup() %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = Type_Trip_P), alpha = 0.1, size = 0.5) +
```

```
  facet_grid(Type_Trip_P~weekday) +
  coord_equal(ratio = 1)+
  scale_color_manual(values = cols) +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        #axis.ticks = element_line(),
        #axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.text = element_blank(),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        strip.text = element_text(face = "bold"),
        strip.text.y = element_text(angle = 0, hjust = 0.5),
        strip.background = element_rect(colour = "grey", fill = "grey")) +
  labs(title = "",
       x = "X-Coordinate",
       y = "Y-Coordinate",
       col = "Type of Trip") +
  ylim(c(6165000, 6190000)) +
  theme(legend.position = "none")

# show number of C and S trips per daytime
# visualize all trips colored by Type of trip (C, R, S)
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
bike_cph_raw5 %>%
  mutate(weekday = factor(weekday, c("Mo", "Di", "Mi", "Do", "Fr","Sa", "So")),
         hour_of_day = as.numeric(hour(datetime)),
         time_of_day = case_when((hour(datetime) < 6 | hour(datetime) >=  19) ~ "19-23 and 0-6, night",
                                 hour(datetime) < 12 ~ "6-10, morning commute",
                                 hour(datetime) < 15 ~ "10-15, noon",
                                 hour(datetime) < 19 ~ "15-19, afternoon commute"),
         time_of_day = factor(time_of_day, c("6-10, morning commute","10-15, noon","15-19, afternoon commute","19-23 and 0-6, night")))
%>%
  #group_by(RespID) %>%
  #mutate(Resp_is_commute = (sum(Trip_is_commmute) / n()) > 0.8) %>%
  group_by(TripID) %>%
  #ungroup() %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(aes(col = Type_Trip_P), alpha = 0.1, size = 0.5) +
  facet_grid(Type_Trip_P~time_of_day) +
  coord_equal(ratio = 1) +
  scale_color_manual(values = cols)
#theme(legend.position = "none")

################################### CLUSTER MAP ######################################
# show all first and last points colored by their clusters
cols2 <- c("1" = "#66c2a5", "2" = "#fc8d62", "3" = "#8da0cb", "0" = "black")

bike_cph_raw5 %>%
  filter(RespID == "58093101") %>%
  mutate(Cluster_Start = as.character(Cluster_Start),
         Cluster_End = as.character(Cluster_End)) %>%
  group_by(TripID) %>%
  slice(c(1, n())) %>%
  ungroup() %>%
  group_by(TripID) %>%
  summarise(x_start = first(x, order_by = TripID),
            y_start = first(y, order_by = TripID),
            Cluster_Start = first(Cluster_Start, order_by = TripID),
            x_end = last(x, order_by = TripID),
            y_end = last(y, order_by = TripID),
            Cluster_End = last(Cluster_End, order_by = TripID)) %>%
  ggplot() +
  scale_shape_identity() +
  geom_point(aes(x_start, y_start,col = Cluster_Start, shape = case_when(Cluster_Start == 0 ~ 15, Cluster_Start >0 ~ 16)), size = 3,
alpha = 0.6) +
  geom_point(aes(x_end, y_end, col=Cluster_End, shape = case_when(Cluster_End == 0 ~ 15, Cluster_End >0 ~ 16)), size = 3, alpha =
0.6) +
  #guides(col = guide_legend(title = "Cluster number")) +
  coord_equal(ratio = 1) +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0))) +
  labs(title = "",
       x = "X-Coordinates",
       y = "Y-Coordinates") +
  scale_color_manual(values = cols2)
```

Appendix G: Trip similarity, approach comparison

```
################ COMPARISON OF BUFFER AND CLUSTER APPROACH FOR TYPE TRIPS #############
bike_cph_raw5 %>%
  mutate(difference = (Type_Trip_B == Type_Trip_P_grouped)) %>%
  group_by(TripID) %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(col="black", alpha = 0.1, size = 0.6) +
  coord_equal(ratio = 1) +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        strip.text = element_text(face = "bold")) +
  theme(legend.position = "none") +
  labs(title = "",
       x = "X-Coordinates",
       y = "Y-Coordinates") +
  ylim(c(6165000, 6190000)) +
  facet_grid(~difference)

Trip_Type_LUT <- bike_cph_raw5 %>%
  group_by(TripID) %>%
  summarise(Type_Trip_B = first(Type_Trip_B),
            Type_Trip_P_grouped = first(Type_Trip_P_grouped))

# get numbers for conversion matrix (how and how many types of trips changed with the two approaches)
C_same <- sum(Trip_Type_LUT$Type_Trip_B == "C" & Trip_Type_LUT$Type_Trip_P_grouped == "C")
S_same <- sum(Trip_Type_LUT$Type_Trip_B == "S" & Trip_Type_LUT$Type_Trip_P_grouped == "S")
C_to_S <- sum(Trip_Type_LUT$Type_Trip_B == "C" & Trip_Type_LUT$Type_Trip_P_grouped == "S")
S_to_C <- sum(Trip_Type_LUT$Type_Trip_B == "S" & Trip_Type_LUT$Type_Trip_P_grouped == "C")
```

Appendix H: Calculating cyclist types

```
options(scipen=6)        # display digits properly, not the scientific version

packages_checker <- function(packages_list){
  for (package in packages_list){
    if (!require(package, character.only = T)) {
      install.packages(package)
    }
    library(package, character.only = T)
  }
}

packages_checker(
  c('raster',
    'ggrepel',
    'tidyverse',
    'data.table',
    'lubridate',
    'ggplot2',
    'sf',
    'dplyr',
    'trip',
    'dbscan'
    )
)

bike_cph_raw6 <- read_csv(
    "data/Bikeability_CPH_5_Type_Trip_P.csv",
    col_types = cols(
    RespID = col_character(),
    TripID_old = col_character(),
    x = col_double(),
    y = col_double(),
    datetime = col_datetime(format = ""),
    TripID = col_character(),
    weekday = col_character(),
    FixTimediffMin = col_double(),
    FixDistMeter = col_double(),
    FixSpeedKmh = col_double(),
    TripDistMeter = col_double(),
    TripDurationMin = col_double(),
    TripSpeedKmh = col_double(),
    TripFixCount = col_integer(),
    Type_Trip_B =col_character(),
    Trip_commute_B = col_integer(),
    Trip_sum = col_integer(),
    Trip_commute_B_perc_mean = col_double(),
```

```
  Type_Trip_P = col_character(),
  Type_Trip_P_grouped = col_character(),
  Cluster_Start = col_integer(),
  Cluster_End = col_integer()
))

################## DEFINITION OF CYCLIST TYPE OF A RESPONDENT ######################
################################## BUFFER METHOD ###################################
bike_cph_perc_B_C_n_C <- bike_cph_raw6 %>%
 group_by(TripID) %>%
 summarise(RespID = first(RespID),
           Type_Trip_B = first(Type_Trip_B)) %>%
 ungroup() %>%
 group_by(RespID) %>%
 summarise(n_Trips = n_distinct(TripID) ,
           n_Commuter_Trips_B = sum(Type_Trip_B == "C")) %>%
 mutate(perc_Commuter_B = (n_Commuter_Trips_B/n_Trips)*100,
        Type_Cyclist_B = ifelse(perc_Commuter_B >= 60, "C", "S"))

bike_cph_raw6 <- left_join(bike_cph_raw6, bike_cph_perc_B_C_n_C, by = "RespID")#join percentage of commuting to original datatable

# plot histogram of percentage of commuts per Resp (each Resp has a percentage to which he/she is commuting)
bike_cph_perc_B_C_n_C %>%
 ggplot(aes(perc_Commuter_B)) +
 geom_histogram(bins = 50) +
 ylim(c(0,70))

################################## POINT METHOD ####################################################
# calculate percentage of Type_cylist (defined as counted C strips devided by total number of Trips per Resp)
bike_cph_perc_P_C_n_C <- bike_cph_raw6 %>%
 group_by(TripID) %>%
 summarise(RespID = first(RespID),
           C = first(Type_Trip_P),
           Cluster_Start = first(Cluster_Start),
           Cluster_End = first(Cluster_End) ) %>%
 ungroup() %>%
 group_by(RespID) %>%
 summarise(n_Trips = n_distinct(TripID) ,
           n_Commuter_Trips_P = sum(C == "C")) %>%
 mutate(perc_Commuter_P = (n_Commuter_Trips_P/n_Trips)*100,
        Type_Cyclist_P = ifelse(perc_Commuter_P >= 60, "C", "S"))

bike_cph_raw6 <- left_join(bike_cph_raw6, subset(bike_cph_perc_P_C_n_C, select = c(RespID, n_Commuter_Trips_P,
perc_Commuter_P, Type_Cyclist_P)), by = "RespID")#join percentage of commuting to original datatable

# plot histogram of percentage of commuts per Resp (each Resp has a percentage to which he/she is commuting)
bike_cph_perc_P_C_n_C %>%
 ggplot(aes(perc_Commuter_P)) +
 geom_histogram(bins = 50)+
 ylim(c(0,70))

############## COMPARISON OF BUFFER AND POINT APPROACH FOR TYPE RESPONDENTS ##################
cols <- c("FALSE" = "red", "TRUE" = "blue")
bike_cph_raw6 %>%
 mutate(difference = (Type_Cyclist_P == Type_Cyclist_B)) %>%
 group_by(TripID) %>%
 ggplot(aes(x,y, group = TripID)) +
 geom_path(aes(col=difference, alpha = 0.1), size = 0.1) +
 scale_color_manual(values = cols)

Resp_Type_LUT <- bike_cph_raw6 %>%
 group_by(RespID) %>%
 summarise(Type_Cyclist_B = first(Type_Cyclist_B),
           Type_Cyclist_P = first(Type_Cyclist_P))

# get numbers for conversion matrix (how and how many types of cyclists changed with the two approaches)
C_same <- sum(Resp_Type_LUT$Type_Cyclist_B == "C" & Resp_Type_LUT$Type_Cyclist_P == "C")
S_same <- sum(Resp_Type_LUT$Type_Cyclist_B == "S" & Resp_Type_LUT$Type_Cyclist_P == "S")
C_to_S <- sum(Resp_Type_LUT$Type_Cyclist_B == "C" & Resp_Type_LUT$Type_Cyclist_P == "S")
S_to_C <- sum(Resp_Type_LUT$Type_Cyclist_B == "S" & Resp_Type_LUT$Type_Cyclist_P == "C")
```

Appendix I: Dwelling

```
################################## FIND DWELLINGS OR RESPONDENTS ##############################
# get first and last points of the days per Resp
dwelling <- bike_cph_raw6 %>%
 group_by(TripID) %>%
 mutate(datetime_floor = first(floor_date(datetime, unit = "day"))) %>% #take first date of a Trip for all other fixes of that one
 group_by(RespID, datetime_floor) %>%
 slice(c(1,n())) %>%
 mutate(Cluster = case_when(row_number() == 1 ~ Cluster_Start,
                            row_number() == 2 ~ Cluster_End)) %>%
```

```
  ungroup() %>%
  filter(Cluster != 0) %>%
  dplyr::select(c(RespID, datetime, x, y, Type_Trip_P, Cluster)) %>%
  group_by(RespID) %>%
  mutate(Max_Cluster = modal(Cluster))%>%  # get most visited cluster --> we assume this to be home
  group_by(RespID) %>%
  filter(Max_Cluster %in% Cluster) %>%
  mutate(x_home = mean(x),
         y_home = mean(y),
         distance = sqrt((x-x_home)^2+(y-y_home)^2),
         x_home_median = median(x),
         y_home_median = median(y),
         distance_median = sqrt((x-x_home_median)^2+(y-y_home_median)^2)) %>%
  summarise(x_home = first(x_home),
            y_home = first(y_home),
            mean_distance = mean(distance),
            point_count = n(),
            accuracy = ifelse(mean_distance > 0, point_count/mean_distance, 1),
            x_home_median = first(x_home_median),
            y_home_median = first(y_home_median),
            mean_distance_median = mean(distance_median),
            accuracy_median = ifelse(mean_distance_median > 0, point_count/mean_distance_median, 1)) %>%
  ungroup()

bike_cph_raw6 <- left_join(bike_cph_raw6, dwelling, by = "RespID")
write_csv(bike_cph_raw6, "data/Bikeability_CPH_6_full.csv")

sum(dwelling$mean_distance < 1000) # get number of homes with higher accuracy than 1km
n_distinct(dwelling$RespID) # get number of RespID for which we can predict "home"
```

```
################################### DWELLING: VISUALISATIONS ########################
#plot start and end points to check if clusters make sense
dwelling %>%
  ggplot(aes(x_home,y_home)) +
  geom_point(aes(col = mean_distance)) +
  coord_cartesian() #+
  #theme(legend.position = "none")

# visualize histogram of mean_distances --> the smaller the value and the higher the point count (number of points for mean calculastions) the better the prediction of home
dwelling %>%
  ggplot(aes(mean_distance)) +
  geom_histogram(bins = 100) #+
  # xlim(c(0,2000))

# plot mean distance of all start/end points to home against the number of points included for home calculation
dwelling %>%
  ggplot(aes(mean_distance, point_count)) +
  geom_point() +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
        axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
        strip.text = element_text(face = "bold"),
        strip.text.y = element_text(angle = 0, hjust = 0.5),
        strip.background = element_rect(colour = "grey", fill = "grey")) +
  labs(title = "",
       x = "Mean distance to dwelling [m]",
       y = "Number of points") +
  theme(legend.position = "none")


# plot all Trips as lines and the home locations colored by C and S based on Start/End calculation
bike_cph_raw6 %>%
  ggplot(aes(x,y, group = TripID)) +
  geom_path(size = 0.5, col = "black", alpha = 0.1) +
  #geom_point(aes(x_home, y_home, col = point_count, size = mean_distance)) +
  geom_point(aes(x_home_median, y_home_median, col = point_count, size = mean_distance_median)) +
  coord_fixed() +
  #theme(legend.position = "none") +
  #scale_alpha_continuous(range = c(0,1)) +
  facet_grid(.~Type_Cyclist_P) +
  scale_colour_gradient(low = "yellow", high = "darkred") +
  xlim(c(710000,730000)) +
  ylim(c(6165000, 6185000))+
  theme_minimal() +
  theme(panel.grid = element_blank(),
        axis.ticks = element_line(),
        axis.text.y = element_text(angle = 90, hjust = 0.5),
        axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
```

```
      axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
      plot.margin = margin(t = 0,r = 20,b=5,l=10),
      strip.text = element_text(face = "bold")) +
 labs(title = "",
     x = "X-Coordinates",
     y = "Y-Coordinates",
     col = "Number of points",
     size = "Mean distance to dwelling")

# plot all Trips as lines and the home locations colored by C and S based on Start/End calculation: zoom into two RespID
cols <- c("C" = "cyan4", "R" = "blue4", "S" = "red3")
target <- c("57090098", "58093101")
bike_cph_raw6 %>%
 filter(RespID %in% target) %>%
 ggplot(aes(x,y, group = RespID)) +
 geom_path(aes(col = Type_Cyclist_P ), size = 0.5, alpha = 0.3) +
 #geom_point(aes(x_home, y_home, col = point_count, size = mean_distance)) +
 geom_point(aes(x_home_median, y_home_median, col = Type_Cyclist_P)) +
 coord_fixed() +
 #theme(legend.position = "none") +
 #scale_alpha_continuous(range = c(0,1)) +
 facet_grid(.~Type_Cyclist_P) +

 #xlim(c(710000,730000)) +
 #ylim(c(6165000, 6185000))
 theme_minimal() +
 theme(panel.grid = element_blank(),
      axis.ticks = element_line(),
      axis.text.y = element_text(angle = 90, hjust = 0.5),
      axis.title.x = element_text(margin = margin(t = 10,r = 0,b=0,l=0)),
      axis.title.y = element_text(margin = margin(t = 0,r = 10,b=0,l=0)),
      plot.margin = margin(t = 0,r = 20,b=5,l=10),
      strip.text = element_text(face = "bold")) +
 labs(title = "",
     x = "X-Coordinates",
     y = "Y-Coordinates",
     col = "Type of Cyclist") +
 scale_y_continuous(limits = c(6173000, 6183000), breaks = c(6175000, 6179000, 6183000))+
 scale_color_manual(values = cols, labels = c("C: 58093101", "S: 57090098"))
```