1 Installation

On Pop!_OS 22.04, which is the same as Ubuntu 22.04: - First ran `sudo apt install libgdal-dev`, without which `rgdal` would not install. - Remaining `install.packages(...)` as per the README provided. There were dependency issues:

```
ERROR: dependencies 'systemfonts', 'textshaping' are not available for
   package 'ragg'
* removing '/home/pranav/R/x86_64-pc-linux-gnu-library/4.1/ragg'
ERROR: dependency 'ragg' is not available for package 'tidyverse'
* removing '/home/pranav/R/x86_64-pc-linux-gnu-library/4.1/tidyverse'

The downloaded source packages are in
    '/tmp/Rtmp9z4dpw/downloaded_packages'
Warning messages:
1: In install.packages(c("dbscan", "rgdal", "lubridate", "stringr",  :
  installation of package 'systemfonts' had non-zero exit status
2: In install.packages(c("dbscan", "rgdal", "lubridate", "stringr",  :
  installation of package 'textshaping' had non-zero exit status
3: In install.packages(c("dbscan", "rgdal", "lubridate", "stringr",  :
  installation of package 'ragg' had non-zero exit status
4: In install.packages(c("dbscan", "rgdal", "lubridate", "stringr",  :
  installation of package 'tidyverse' had non-zero exit status
```

Another dependency is ggplot2, which produces the plots but isn't mentioned in the dependency list. I have installed ggplot2 (3.4.3) but am faced with a bunch of errors with plotting when I try to run the command.

2 Functions

Below, I have gone through the functions file-by-file. Wherever relevant, I have flagged appropriate lines and mentioned the line numbers here. I have left detailed comments in the files themselves. I have also added general comments, if any, over here.

2.1 coati_function_library_V1.R

2.1.1 latlon.to.utm(...)

- The argument `utm.zone` defaults to 34, where the KRR is. Pay close attention to this while using it in the dataset from Panama (which falls in 16N and 17N)! Also keep in mind the default value for the argument `southern_hemisphere` is `TRUE`.
- l30 flagged.

2.1.2 utm.to.latlon(...)

- Same concerns as the argument above, about default argument values. Not a bug precisely, but a likely source of potential bugs.

2.1.3 get_subgroup_data(...)

- I want to raise a concern about the NaN policy espoused here. The DBSCAN algorithm is run, and subgroups determined, when even one location is not NaN. This would be sort of fine in cases where you have an enormous number of data-points, but in this case, the max number of data-points at any time is the number of tagged coatis. This makes the grouping particularly susceptible to missing individuals, especially in some spatial conformations of the group. Depending on how much NaNs are in the data, I would strongly urge going for a tougher NaN policy, i.e., not computing subgroup identities for a point of time if *any* location there is missing. If this is not feasible because of a lot of NaNs in some individuals, some sensitivity analyses trying out varying values of the DBSCAN radius might become necessary.

2.1.4 visualize_network_matrix(...)

- checked

2.1.5 visualize_network_matrix_trago(...)

- checked
- This seems, however, to be more or less a duplicate of the function above. Why not implement this as an argument in the above function instead? Much easier to avoid small bugs that way.

2.1.6 get_proximity_data(...)

- function generates an overall-proximity matrix
- I am not entirely sure why the loop over time is necessary. You can call arithmetic operators on vectors pretty easily, e.g., `ds <- sqrt(xs[i,]^2 + xs[j,]^2)`. A similar point about pointless computation is that i and j are both looped over all individuals. This isn't necessary because proximity and distances are both symmetric measures. You could therefore use

```
```
for(i in 1:n_ind){
  for(j in i+1:n_ind){
      ...
  }
}
```
```

- However, of course, the above points don't get in the way of the correctness of the results. This function therefore checks out.

2.1.7 randomise_splits(...)

- checked
- TODO: confirm what n_sub1, n_sub2, and n_sub3 are

2.1.8 dist_to_0_or_1(...)

- checked
- but could also simply be implemented as

```
return(min(q, 1-q))
```

:p

2.1.9 get_consistency(...)

- checked code
- however the basis for this definition itself took me a while to get, even after reading the manuscript and looking at the code. Maybe a slightly more elaborate motivation of this metric is needed in the manuscript.

2.1.10 get_p_dyad_together(...)

- once again I need to clarify what sub1, sub2, and sub3 exactly are.
- otherwise, checked.
- Good that here j loops over (i+1):n_inds_local.

2.2 fission_fusion_galaxy_V1.R

2.2.1 mode

- checked

2.2.2 the ggplot call

The ggplot call at the very end fails, and as a result no plots are generated. This could be an issue with the version specific to my computer (R 4.1.2 and ggplot2 3.4.3). However, here are the errors I encountered. **Note that no plots were generated in the first run.**

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
Warning messages:
1: The 'size' argument of 'element_line()' is deprecated as of ggplot2
3.4.0.
info: Please use the 'linewidth' argument instead.
This warning is displayed once every 8 hours.
Call 'lifecycle::last_lifecycle_warnings()' to see where this warning
was generated.
2: All formats failed to parse. No formats found.
3: Removed 20 rows containing missing values ('geom_rect()').
4: Removed 12172 rows containing missing values ('geom_point()').
5: Removed 12172 rows containing missing values ('geom_line()').
```

I made a few small corrections (not bugs, but guidelines about how dir names used must always end in "/") to start generating the images.

- Fig 2a, b, c check out.

- Figure 3a checks out, but why is there a re-ordering done? Was it just to put the two subgroups together?
- Figure 3b checks out. The symmetrization block seems irrelevant, since this data should be symmetric anyway (and is, looking at the previous figure).
- Fig 4a, mostly checks out but the orange line is almost on top of 0.3 for me. Ideally this should be the same, since there are no random effects, right? It's a very very small difference, but it's better to be safe.

  Additionally, I have concerns about the way group splits are defined in this block. Crucially, I am concerned about this bit:

```
#determine if this time step is a split
#if we have one group that goes to more than one, and there are no
   singletons subsequently, then it's a split
if(n_subgroups_now==1
   & n_subgroups_later >1
   & singletons_later==0
){
  splits <- c(splits, t)
}

#if we have more than one group, but rest are singletons, and
   number of singletons doesn't change (so we don't have just one
   loner moving off), then it's a split
if(n_subgroups_now > 1
   & (singletons_now+1) == n_subgroups_now
   & n_subgroups_later > n_subgroups_now
   & singletons_now == singletons_later
){
  splits <- c(splits, t)
```

  I feel this doesn't account for a bunch of cases, e.g., groups splitting from 2 to 3 large subgroups (which might visually not happen from watching the animation, but might still happen with the DBSCAN approach). If an individual is initially characterised as part of one subgroup and moves to another one, without being a singleton in the middle (i.e., if two subgroups were close to each other and an individual moved from one to the other), this wouldn't be recognised either. These are obviously rare events, but given the duration of data we have, it might be important to address them, either in the code or in the manuscript.
- Figure 5 checks out. However the code here is a bit smelly, since the y-axis values and coati names are pre-programmed magic numbers in the code. This means this code can't easily survive if the data is modified, or if we want to run this on another group. However, since Figure 5 is only an illustration of fission-fusion, this should still be okay.
- Figure 6a checks out. However, I am not a big fan of this code block. Variable names like xs_nogus make the code very specific to this paper alone and not generally usable in the future. Further, are we assuming that R=10 is a good distance for within group dyads?
- Figure S2a sort of checks out. There are some differences that are subtle. E.g. the bottom-most image looks subtly different in mine. Were there changes made to the code since the image in the manuscript was generated?
- Figure S3a checks out. However, why have you considered the mean subgroup size in the violinplot, as opposed to just looking at all non-singleton groups and making violinplots of those? I feel that would be more informative.
- Figure S4a, b check out