5. Getting Spatial with sftrack

```r
# Make tracks from raw data
data <- read.csv(system.file('extdata/raccoon_data.csv', package='sftrack'))
data$month <- as.POSIXlt(data$acquisition_time)$mon+1

data$time <- as.POSIXct(data$acquisition_time, tz='EST')
coords = c('longitude','latitude')
burst = list(id = data$sensor_code, month = as.POSIXlt(data$acquisition_time)$mon+1)
time = 'time'
error = 'fix'
crs = '+init=epsg:4326'
my_sftrack <- as_sftrack(data = data, coords = coords, burst = burst, time = time, error = error, crs =
my_sftraj <- as_sftraj(data = data, coords = coords, burst = burst, time = time, error = error, crs = c
```

Geometry column

As stated earlier, the geometry column is built using `sf`, so functions exactly as it would in `sf`. You can modify it and redefine it using the `sf` tools. More specifically the geometry column of an sf_track object is a `sfc` column. The main difference between a standard `sf` object created using `st_as_sf` is that we automatically allow empty geometries, where as this option is turned off by default in `st_as_sf()`.

```r
my_sftrack$geometry
```

```
## Geometry set for 445 features  (with 168 geometries empty)
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: -80.28149 ymin: 26.06761 xmax: -80.27046 ymax: 26.07706
## CRS:            +init=epsg:4326
## First 5 geometries:

## POINT EMPTY

## POINT (-80.27906 26.06945)

## POINT EMPTY
## POINT EMPTY

## POINT (-80.27431 26.06769)
```

An `sftrack` object is simply an `sfc` of `sf_POINTS`, this contrasts with an `sftraj` object which is a mixture of a `GEOMETERYCOLLECTION` and `LINESTRING`. This is because a trajectory can have a start point and an NA end point, a line segment, or an NA and an end point. This allows no-loss conversion back and forth between `sftrack` and an `sftraj`, and because linestrings can not have a NULL point in them.

```r
my_sftraj$geometry
```

```
## Geometry set for 445 features  (with 168 geometries empty)
## geometry type:  GEOMETRY
## dimension:      XY
## bbox:           xmin: -80.28149 ymin: 26.06761 xmax: -80.27046 ymax: 26.07706
## CRS:            +init=epsg:4326
## First 5 geometries:

## POINT EMPTY
## POINT (-80.27906 26.06945)

## POINT EMPTY
## POINT EMPTY

## LINESTRING (-80.27431 26.06769, -80.2793 26.06867)
```

This does mean that not all `sf` functions will handle an `sftraj` object like it would an `sftrack` if there are NAs in the data set. To help with working with sftraj objects, there are two functions that help extract points from `sftraj` objects.

coord_traj

This function returns a data.frame (x,y,z) of the beginning point of each sftraj geometry.

```
coord_traj(my_sftraj$geometry)[1:10,]
```

```
##             [,1]     [,2]
##  [1,]         NA       NA
##  [2,] -80.27906 26.06945
##  [3,]         NA       NA
##  [4,]         NA       NA
##  [5,] -80.27431 26.06769
##  [6,] -80.27930 26.06867
##  [7,] -80.27908 26.06962
##  [8,] -80.27902 26.06963
##  [9,]         NA       NA
## [10,] -80.27900 26.06982
```

pts_traj

And `pts_traj` returns a list of the beginning point of each sftraj geometry.

```
pts_traj(my_sftraj$geometry)[1:10]
```

```
## [[1]]

## POINT EMPTY

##
## [[2]]

## POINT (-80.27906 26.06945)

##
## [[3]]

## POINT EMPTY

##
## [[4]]

## POINT EMPTY

##
## [[5]]

## POINT (-80.27431 26.06769)

##
## [[6]]
## POINT (-80.2793 26.06867)

##
## [[7]]

## POINT (-80.27908 26.06962)

##
## [[8]]

## POINT (-80.27902 26.06963)

##
## [[9]]

## POINT EMPTY

##
## [[10]]

## POINT (-80.279 26.06982)
```

is_linestring

May help if you'd like to quickly filter an `sftraj` object to just contain pure linestrings. `is_linestring()` returns TRUE or FALSE if the geometry is a linestring. This does not recalculate anything, it just filters out steps that contained NAs in either phase.

```
is_linestring(my_sftraj$geometry)[1:10]
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
```

```
new_sftraj <- my_sftraj[is_linestring(my_sftraj$geometry),]
head(new_sftraj)
```

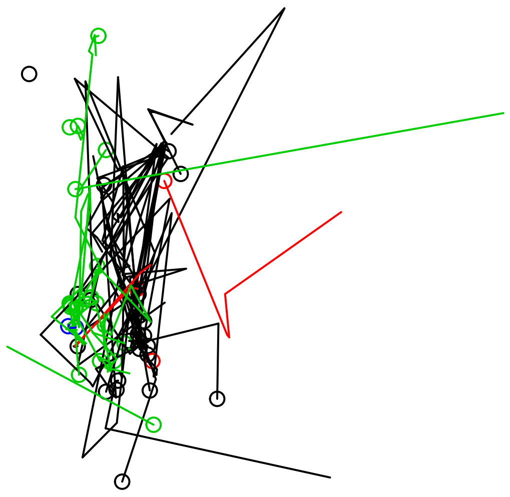
```
## Sftraj with 6 features and 14 fields (0 empty geometries)
## Geometry : "geometry" (XY, crs: +init=epsg:4326)
## Timestamp : "time" (POSIXct in EST)
## Burst : "burst" (*id*, *month*)
## -------------------------------
##    sensor_code   utc_date utc_time latitude longitude height hdop vdop fix
## 5         CJ11 2019-01-19 04:02:30 26.06769 -80.27431    858  5.1  3.2  2D
## 6         CJ11 2019-01-19 05:02:30 26.06867 -80.27930    350  1.9  3.2  3D
## 7         CJ11 2019-01-19 06:02:30 26.06962 -80.27908     11  2.3  4.5  3D
## 10        CJ11 2019-01-19 17:02:30 26.06982 -80.27900     NA  2.0  3.3  3D
## 11        CJ11 2019-01-19 18:02:05 26.06969 -80.27894      8  4.2  2.5  3D
## 12        CJ11 2019-01-19 19:02:04 26.07174 -80.27890     -3  0.9  1.5  3D
##       acquisition_time month                time                burst
## 5  2019-01-19 04:02:30     1 2019-01-19 04:02:30 (id: CJ11, month: 1)
## 6  2019-01-19 05:02:30     1 2019-01-19 05:02:30 (id: CJ11, month: 1)
## 7  2019-01-19 06:02:30     1 2019-01-19 06:02:30 (id: CJ11, month: 1)
## 10 2019-01-19 17:02:30     1 2019-01-19 17:02:30 (id: CJ11, month: 1)
## 11 2019-01-19 18:02:05     1 2019-01-19 18:02:05 (id: CJ11, month: 1)
## 12 2019-01-19 19:02:04     1 2019-01-19 19:02:04 (id: CJ11, month: 1)
##                         geometry
## 5  LINESTRING (-80.27431 26.06...
## 6  LINESTRING (-80.2793 26.068...
## 7  LINESTRING (-80.27908 26.06...
## 10 LINESTRING (-80.279 26.0698...
## 11 LINESTRING (-80.27894 26.06...
## 12 LINESTRING (-80.2789 26.071...
```

Plotting

Base plotting

Currently there are some basic plotting methods. Base plotting currently does not have any thrills built into it, and assumes that the `active_burst` is the grouping/coloring variable.

```
plot(my_sftraj)
```

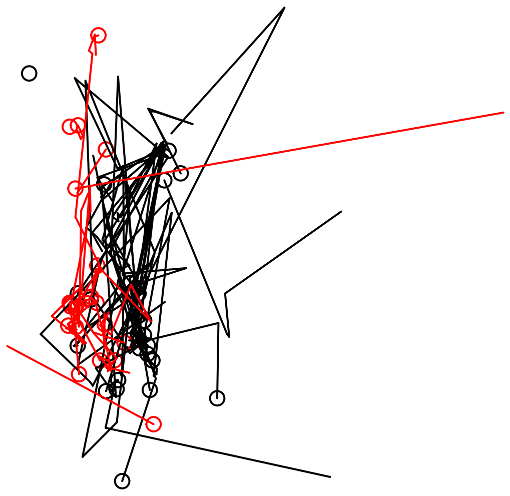


And changing the active burst will change the plot view

```
active_burst(my_sftraj$burst) <- 'id'
active_burst(my_sftraj$burst)
```

```
## [1] "id"
```

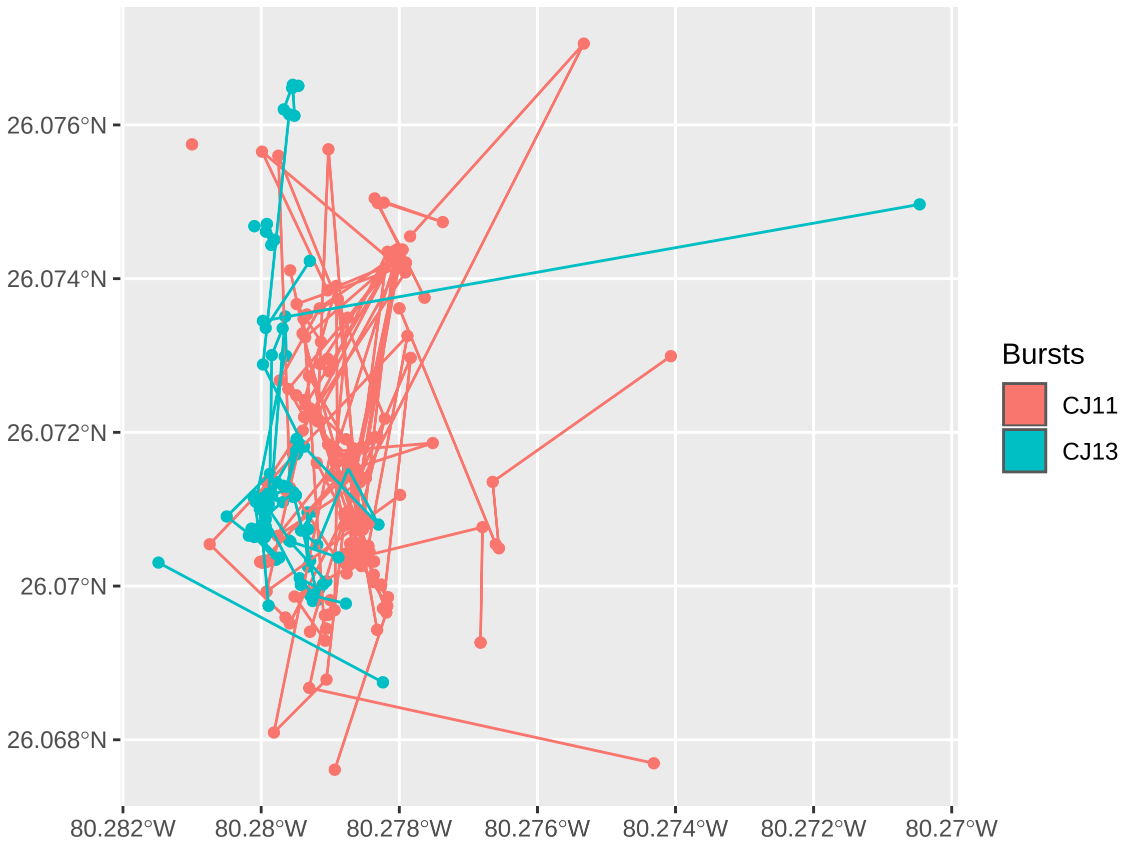
```
plot(my_sftraj)
```



ggplot

This is a work in progress, but theres a rudimentary geom_sftrack function. As of now you have to input `data` into the geom_sftrack function. That'll change as I look more into it. Again ggplot assumes active_burst is the grouping variable. Plots vary slightly based on if they're track of traj

```r
library(ggplot2)
ggplot() + geom_sftrack(data = my_sftraj)
```


26.076°N
26.074°N
26.072°N
26.07°N
26.068°N
80.282°W
80.28°W
80.278°W
80.276°W
80.274°W
80.272°W
80.27°W
Bursts
CJ11
CJ13