3. Introduction to an sftrack/sftraj object

Some basic functionality of sf_track and sf_traj objects

print

`print()` prints out the type of object as well as specific data on the sf_track object. Additionally you can supply the number of rows or columns you'd like to display with arguments `n_row` and `n_col`. When using `n_col` the display will show the `burst` and `geometery` fields as well as any other columns starting from column 1 until `#columns + 2 = n_col`. If neither is provided than print just uses default values in the global options. `ncol` and `nrow` are optional arguments, defaults to data.frame defaults.

```
# 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
print(my_sftrack,5,10)
```

```
## Sftrack with 445 features and 14 fields (168 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
## 1        CJ11 2019-01-19 00:02:30       NA        NA     NA  0.0  0.0  NO
## 2        CJ11 2019-01-19 01:02:30 26.06945 -80.27906      7  6.2  3.2  2D
## 3        CJ11 2019-01-19 02:02:30       NA        NA     NA  0.0  0.0  NO
## 4        CJ11 2019-01-19 03:02:30       NA        NA     NA  0.0  0.0  NO
## 5        CJ11 2019-01-19 04:02:30 26.06769 -80.27431    858  5.1  3.2  2D
##      acquisition_time ...                burst                  geometry
## 1 2019-01-19 00:02:30 ... (id: CJ11, month: 1)                POINT EMPTY
## 2 2019-01-19 01:02:30 ... (id: CJ11, month: 1) POINT (-80.27906 26.06945)
## 3 2019-01-19 02:02:30 ... (id: CJ11, month: 1)                POINT EMPTY
## 4 2019-01-19 03:02:30 ... (id: CJ11, month: 1)                POINT EMPTY
## 5 2019-01-19 04:02:30 ... (id: CJ11, month: 1) POINT (-80.27431 26.06769)
##                  time
## 1 2019-01-19 00:02:30
## 2 2019-01-19 01:02:30
## 3 2019-01-19 02:02:30
## 4 2019-01-19 03:02:30
## 5 2019-01-19 04:02:30
```

summary

`summary()` works as youd normally expect for a data.frame, except it displays the burst column as a count of

each active_burst combination.

```
summary(my_sftrack)
```

```
##  sensor_code       utc_date       utc_time      latitude
##  CJ11:222    2019-01-19: 32   17:02:30: 26   Min.   :26.07
##  CJ13:223    2019-01-20: 32   23:02:30: 20   1st Qu.:26.07
##              2019-01-21: 32   00:02:30: 19   Median :26.07
##              2019-01-22: 32   18:02:30: 19   Mean   :26.07
##              2019-01-23: 32   01:02:30: 17   3rd Qu.:26.07
##              2019-01-25: 32   07:02:30: 17   Max.   :26.08
##              (Other)   :253   (Other) :327   NA's   :168
##    longitude          height             hdop            vdop
##  Min.   :-80.28   Min.   : -30.00   Min.   :0.000   Min.   :0.000
##  1st Qu.:-80.28   1st Qu.:   1.00   1st Qu.:0.000   1st Qu.:0.000
##  Median :-80.28   Median :   7.00   Median :1.300   Median :1.900
##  Mean   :-80.28   Mean   :  36.65   Mean   :1.691   Mean   :1.938
##  3rd Qu.:-80.28   3rd Qu.:  15.50   3rd Qu.:2.500   3rd Qu.:3.200
##  Max.   :-80.27   Max.   :1107.00   Max.   :9.900   Max.   :8.400
##  NA's   :168      NA's   :198
##  fix                acquisition_time     month
##  2D: 37   2019-01-19 00:02:30:  2     Min.   :1.000
##  3D:240   2019-01-19 01:02:30:  2     1st Qu.:1.000
##  NO:168   2019-01-19 04:02:30:  2     Median :1.000
##           2019-01-19 06:02:30:  2     Mean   :1.067
##           2019-01-19 17:02:30:  2     3rd Qu.:1.000
##           2019-01-20 02:02:30:  2     Max.   :2.000
##           (Other)            :433
##       time                         burst              geometry
##  Min.   :2019-01-19 00:02:30   CJ11_1:207   POINT        :445
##  1st Qu.:2019-01-22 07:02:30   CJ11_2: 15   epsg:4326    :  0
##  Median :2019-01-25 23:02:30   CJ13_1:208   +proj=long...:  0
##  Mean   :2019-01-25 22:22:18   CJ13_2: 15
##  3rd Qu.:2019-01-29 07:02:09
##  Max.   :2019-02-01 23:02:30
##
```

summary_sftrack

`summary_sftrack()` is a special summary function specific for sftrack objects. It summarizes the data based on the beginning and end of each burst as well as the total distance of the burst. This function uses `st_length` from the `sf` package and therefore outputs in units of the crs. In this example the distance is in degrees distance.

```
summary_sftrack(my_sftrack)
```

```
## Linking to GEOS 3.7.0, GDAL 2.4.0, PROJ 5.2.0
```

```
##    burst points NAs          begin_time            end_time    length_m
## 1 CJ11_1    207   0 2019-01-19 00:02:30 2019-01-31 23:02:30 24666.45395
## 2 CJ11_2     15   0 2019-02-01 00:02:30 2019-02-01 23:02:30  1924.48555
## 3 CJ13_1    208   0 2019-01-19 00:02:30 2019-01-31 23:02:30 10108.95506
## 4 CJ13_2     15   0 2019-02-01 00:02:30 2019-02-01 23:02:07    32.31794
```

You can also trigger this function by using `summary(data, stats = TRUE)`

```
summary(my_sftrack, stats = TRUE)
```

```
##    burst points NAs          begin_time            end_time    length_m
## 1 CJ11_1    207   0 2019-01-19 00:02:30 2019-01-31 23:02:30 24666.45395
## 2 CJ11_2     15   0 2019-02-01 00:02:30 2019-02-01 23:02:30  1924.48555
## 3 CJ13_1    208   0 2019-01-19 00:02:30 2019-01-31 23:02:30 10108.95506
## 4 CJ13_2     15   0 2019-02-01 00:02:30 2019-02-01 23:02:07    32.31794
```

An sftrack object attempts to act like a data.frame and sf whenever appropriate. Because of this you can subset an sftrack object as you would a data.frame. Except, like sf, it attempts to retain the geometry, burst, and time columns, in order to maintain sftrack status.

In this way row subsetting is very straight forward, as each row represents an individual point in time.

```
my_sftrack[1:10,]
```

```
## Sftrack with 10 features and 14 fields (4 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
## 1         CJ11 2019-01-19 00:02:30       NA        NA     NA  0.0  0.0  NO
## 2         CJ11 2019-01-19 01:02:30 26.06945 -80.27906      7  6.2  3.2  2D
## 3         CJ11 2019-01-19 02:02:30       NA        NA     NA  0.0  0.0  NO
## 4         CJ11 2019-01-19 03:02:30       NA        NA     NA  0.0  0.0  NO
## 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
## 8         CJ11 2019-01-19 07:02:04 26.06963 -80.27902      9  2.7  3.9  3D
## 9         CJ11 2019-01-19 08:02:30       NA        NA     NA  0.0  0.0  NO
## 10        CJ11 2019-01-19 17:02:30 26.06982 -80.27900     NA  2.0  3.3  3D
##       acquisition_time month                time                burst
## 1  2019-01-19 00:02:30     1 2019-01-19 00:02:30 (id: CJ11, month: 1)
## 2  2019-01-19 01:02:30     1 2019-01-19 01:02:30 (id: CJ11, month: 1)
## 3  2019-01-19 02:02:30     1 2019-01-19 02:02:30 (id: CJ11, month: 1)
## 4  2019-01-19 03:02:30     1 2019-01-19 03:02:30 (id: CJ11, month: 1)
## 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)
## 8  2019-01-19 07:02:04     1 2019-01-19 07:02:04 (id: CJ11, month: 1)
## 9  2019-01-19 08:02:30     1 2019-01-19 08:02:30 (id: CJ11, month: 1)
## 10 2019-01-19 17:02:30     1 2019-01-19 17:02:30 (id: CJ11, month: 1)
##                       geometry
## 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)
```

Subsetting by column however, sftrack attempts to retain imporatnt columns by default. THis mirrors sf

functionality.

```
my_sftrack[1:3,c(1:3)]
```

```
## Sftrack with 3 features and 7 fields (2 empty geometries)
## Geometry : "geometry" (XY, crs: +init=epsg:4326)
## Timestamp : "time" (POSIXct in EST)
## Burst : "burst" (*id*, *month*)
## ------------------------------
##   sensor_code   utc_date utc_time                burst                time
## 1        CJ11 2019-01-19 00:02:30 (id: CJ11, month: 1) 2019-01-19 00:02:30
## 2        CJ11 2019-01-19 01:02:30 (id: CJ11, month: 1) 2019-01-19 01:02:30
## 3        CJ11 2019-01-19 02:02:30 (id: CJ11, month: 1) 2019-01-19 02:02:30
##   fix                   geometry
## 1  NO                POINT EMPTY
## 2  2D POINT (-80.27906 26.06945)
## 3  NO                POINT EMPTY
```

}

To turn off this feature, you just use the drop = T argument, again a lot like sf.

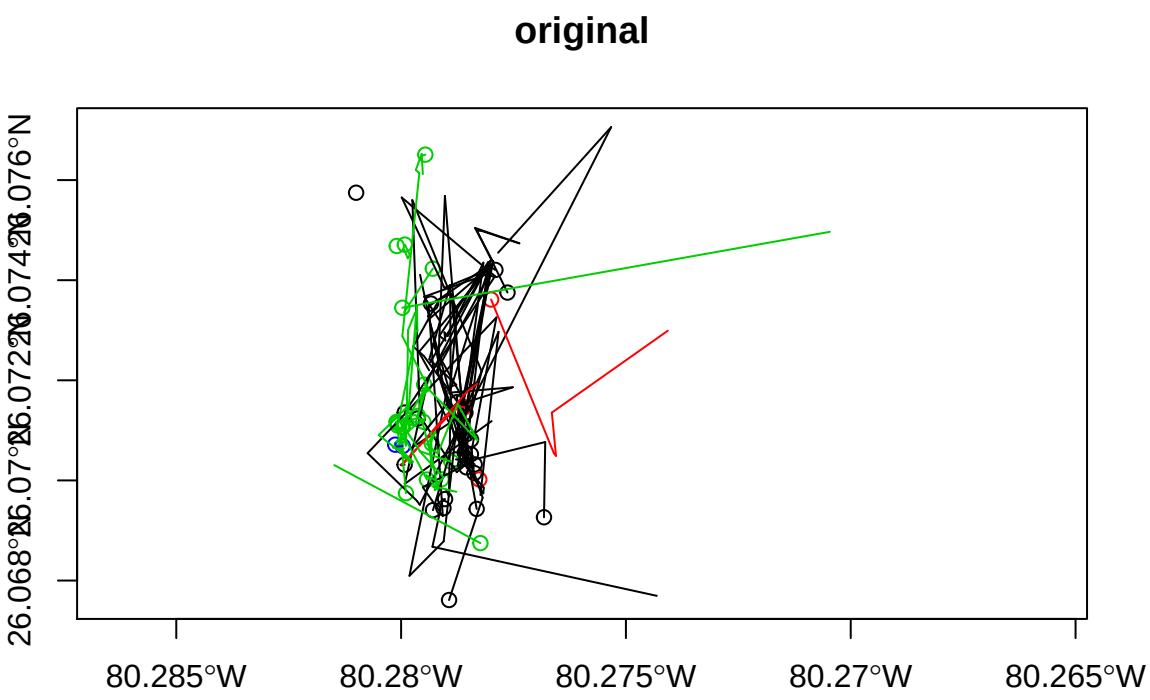
```
my_sftrack[1:3,c(1:3), drop = TRUE]
```

```
##   sensor_code   utc_date utc_time
## 1        CJ11 2019-01-19 00:02:30
## 2        CJ11 2019-01-19 01:02:30
## 3        CJ11 2019-01-19 02:02:30
```

sftrajs work nearly the same, however because they are a step model where in the steps are modeled as step1 (t1 ->t2) its important to note that subsetting will not automatically calculate any steps for you.

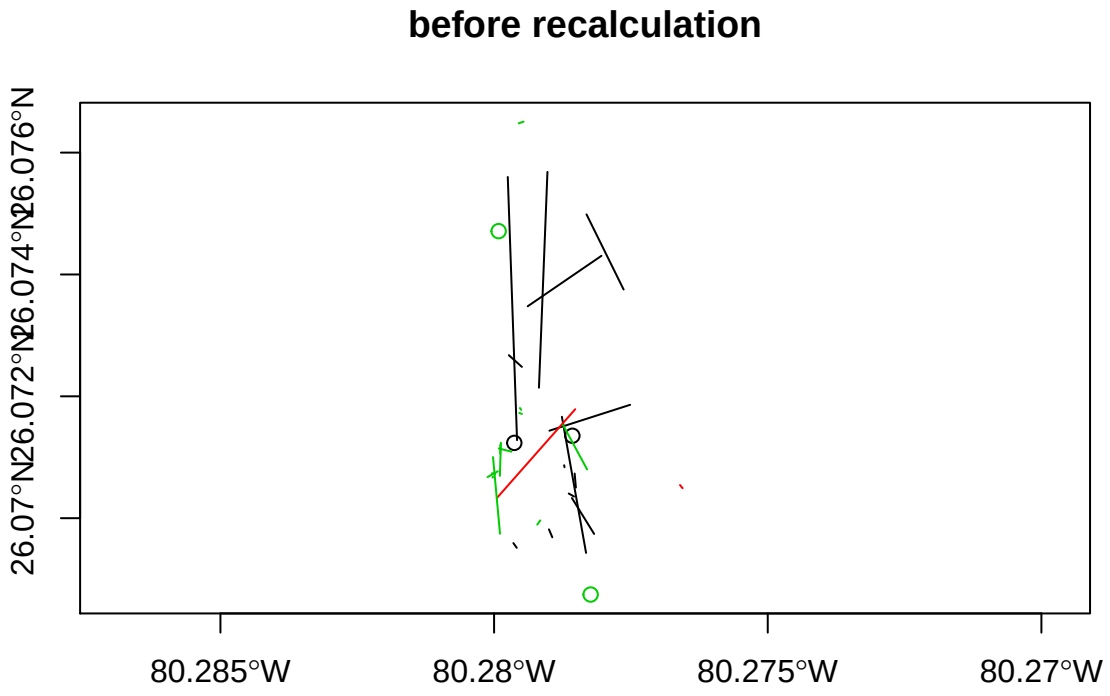
If your subsetting will also change your steps, creating new end points for steps, then you can recalculate using `step_recalc()`

```
plot(my_sftraj, main ='original', axes = T)
```



```
new_traj <- my_sftraj[seq(10,nrow(my_sftraj),10),]

plot(new_traj, main ='before recalculation', axes = T)
```




```
plot(step_recalc(new_traj),  main ='after recalculation', axes = T)
```