5. Getting Spatial with sftrack

```
# 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()`.

```
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 `POINT` 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.

```
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...
```

Working with sf

As sftrack object is an sf object, all of the sf functions should apply to it.

```
st_length(my_sftraj)
```

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

```
## Units: [m]
##   [1]   0.0000000   0.0000000   0.0000000   0.0000000 510.8714556
##   [6] 107.0287059   5.6349599   0.0000000   0.0000000  15.7035150
##  [11] 227.7663429 239.7379004 170.7660219  82.5258644  26.9687055
##  [16] 256.5123603 238.5230189   0.0000000   0.0000000   0.0000000
##  [21] 363.5362545 244.1993753  98.1483499   0.0000000   0.0000000
##  [26]   0.0000000   0.0000000 374.8201033 938.1725945  10.5660282
##  [31] 152.3002015 415.1248903 308.0857699   0.0000000   0.0000000
##  [36]  64.9078204 156.0560792 245.6805628 155.1411918   3.7345709
##  [41]  30.5290531   1.9061395 421.6968259 181.1519842   6.7694778
##  [46] 176.0342771   3.1272742 195.9210748 114.5172427 163.0354153
##  [51] 129.2719920 335.8737555  77.6903438 114.4672753 376.7654781
##  [56]  50.0466066   0.0000000   0.0000000   0.0000000 154.9130369
##  [61] 141.4234961 284.9891073 105.0646659 221.4704890 271.0043825
##  [66] 362.6610168   0.0000000   0.0000000   0.0000000  76.8325967
##  [71]  12.6665281  59.2567386   0.0000000   0.0000000   0.0000000
##  [76]   0.0000000   0.0000000 216.0339162 409.3219388 478.8476670
##  [81] 148.4180312  40.6480339 170.7565263 166.5145447   0.0000000
##  [86]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
##  [91]   0.0000000   0.0000000 131.6923624   0.0000000   0.0000000
##  [96]   0.0000000   0.0000000  44.3932525 201.8836802 392.6867171
## [101] 598.7014100  13.4733146   4.2066562   0.0000000   0.0000000
## [106] 238.2652793 237.5518116 295.9592039 340.5362043   0.0000000
## [111]   0.0000000 418.6155508 107.8181759 236.5969944 173.6311798
## [116]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [121]   0.0000000   0.0000000 250.3304944  17.0655356   0.8293456
## [126]   0.0000000   0.0000000   0.0000000 119.0036570  31.9722845
## [131] 126.1090348   0.0000000   0.0000000   0.0000000   0.0000000
## [136]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [141]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [146]  46.3109531 249.4813339 325.2991667   0.0000000   0.0000000
## [151]   0.0000000   0.0000000 376.3184949 406.7889326   1.4233156
## [156]  83.3399275 104.4106785  89.8023530   8.8990177 152.5428298
## [161]   0.0000000   0.0000000   0.0000000   0.0000000   7.5071903
## [166] 104.5919263   0.0000000   0.0000000   0.0000000  25.6396316
## [171] 404.9271287 196.2486338  16.7368212  18.2610383  10.9697672
## [176] 180.4926587 283.7361067  23.6135161  50.9656066 251.3444419
## [181]   0.0000000   0.0000000   0.0000000   0.0000000  21.2363910
## [186]  29.3129199   0.0000000   0.0000000   0.0000000   0.0000000
## [191]   0.0000000 168.4128397 152.8808436 135.1427922  25.8317628
## [196]  75.2758434  40.9341826   6.6988690  14.7821058  11.1944984
## [201]  14.5344967   0.0000000   0.0000000   3.2958454   6.2373376
## [206]   0.0000000   0.0000000   0.0000000  26.5005017 216.0371881
```

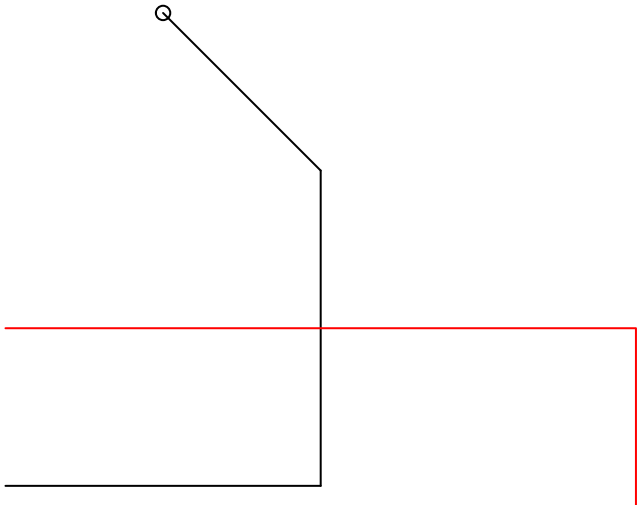
```
## [211]   4.3470222 179.8324356  14.3869685   0.0000000   0.0000000
## [216]   0.0000000   0.0000000 315.4746587  96.1479293   7.5176431
## [221] 367.5706975   0.0000000   0.0000000  78.1698240  51.2340492
## [226]  31.1237615  33.2210336  31.3572201 118.6291605  91.8129256
## [231] 171.2130339   0.0000000   0.0000000   0.0000000   0.0000000
## [236]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [241]   0.0000000  55.4857603  16.7746529  22.4831281  17.4070597
## [246]  26.8661140   0.0000000   0.0000000   0.0000000   9.4109854
## [251]   0.0000000   0.0000000  44.6934626  37.7246547  10.5194537
## [256] 362.6553617 132.5737308  15.0758303   1.8672854   5.0728935
## [261]  84.6934612   0.0000000   0.0000000   0.0000000   0.0000000
## [266]   0.0000000  20.6062984   8.7678869  25.3419680   0.0000000
## [271]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [276]   0.0000000   0.0000000 966.0786870   0.0000000   0.0000000
## [281]  48.3323282   7.7735963  12.7196006  47.2618808  97.5237486
## [286]   3.6262626  14.8476250  22.9771464   9.1479643  23.7982843
## [291]  25.9535196  13.5200260   9.2686038   0.0000000   0.0000000
## [296]   0.0000000   3.1379141   5.6499140   5.2829820   4.0501372
## [301]   2.0405863   9.6285167  10.5068246   7.3304186  64.7368563
## [306]   0.0000000   0.0000000   0.0000000  17.9045571 139.9946161
## [311]   0.0000000   0.0000000  80.1795921  26.8171112   3.7400596
## [316]   9.1512513  12.3123962  40.7260452  32.5010958   9.4917346
## [321]  17.8297841   0.0000000   0.0000000   0.0000000   0.0000000
## [326] 115.8627361   0.0000000   0.0000000 368.1790264   0.0000000
## [331]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [336]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [341]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [346]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [351]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [356]   0.0000000   0.0000000  41.4977777 195.7512835  60.3498683
## [361]  20.0245130 107.4395232   0.0000000   0.0000000   0.0000000
## [366]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [371]   0.0000000   0.0000000   0.0000000   6.1455038 240.1723319
## [376]  56.9502530   2.4075210 213.4351114   0.0000000   0.0000000
## [381]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [386]   0.0000000   0.0000000   0.0000000   0.0000000  21.6028916
## [391]  76.8510782  87.7443313  96.5784230  49.3533231   2.0406039
## [396]   2.4880698   5.2829976  36.1319741  33.8940514   5.7033148
## [401]  20.4422629  14.1781878  14.2385447  80.2622241   6.5162633
## [406]   3.5551634   0.0000000   0.0000000   0.0000000  16.0052810
## [411]  14.6496874  75.8269003  73.2178407   0.0000000   0.0000000
## [416]  15.5675831  16.5019860  47.1963671  44.4738051   2.0260187
## [421]  56.1690783   0.0000000   0.0000000   0.0000000   0.0000000
## [426]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [431]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [436]   0.0000000   0.0000000   0.0000000   0.0000000   0.0000000
## [441]  13.0956767   0.0000000   0.0000000   2.7013581   0.0000000
```

```r
df1 <- data.frame(
  id = c(1,1,1,1),
  time = as.POSIXct('2020-01-01 12:00:00', tz = 'UTC') + 60*60*(1:4),
  x = c(1,3,3,2),
  y = c(1,1,3,4)
)
```

```
road <- st_linestring(rbind(
  c(1,2),
  c(5,2),
  c(5,0)
)
)

animal1 <- as_sftraj(df1)
plot(animal1)

plot(road, add = T, col = 'red')
```



```
# Does the animal cross the road?

any(st_intersects(animal1, road, sparse = F))
```

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

```
# When?
animal1$time[st_intersects(animal1, road, sparse = F)]
```

```
## [1] "2020-01-01 14:00:00 UTC"
```

```
# How often does the animal stay near the road?
st_is_within_distance(animal1, road, 1)
```

```
## Sparse geometry binary predicate list of length 4, where the predicate was `is_within_distance'
##  1: 1
##  2: 1
##  3: 1
##  4: (empty)
```

```
# How close is the animal from the road?

st_distance(animal1, road)
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    1
```

```
## [4,]    2
```

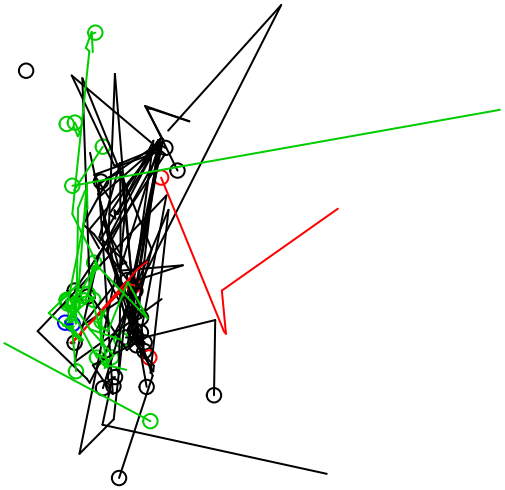
The only thing to remember, is that a sftraj is a `GEOMETRY` column, and that not all functions will work with it. This is when using is_linestring maybe appropriate.

Plotting

Base plotting

Currently there are some basic plotting methods. Base plotting is built ontop of the sf functionality, while simply controlling how to group and color the points.

```
plot(my_sftraj)
```

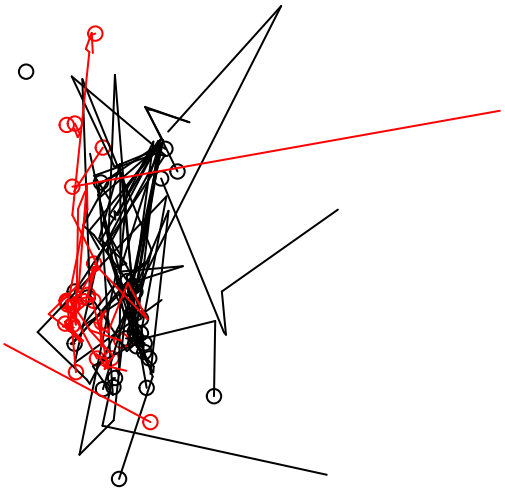


And changing the active burst will change the plot view

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

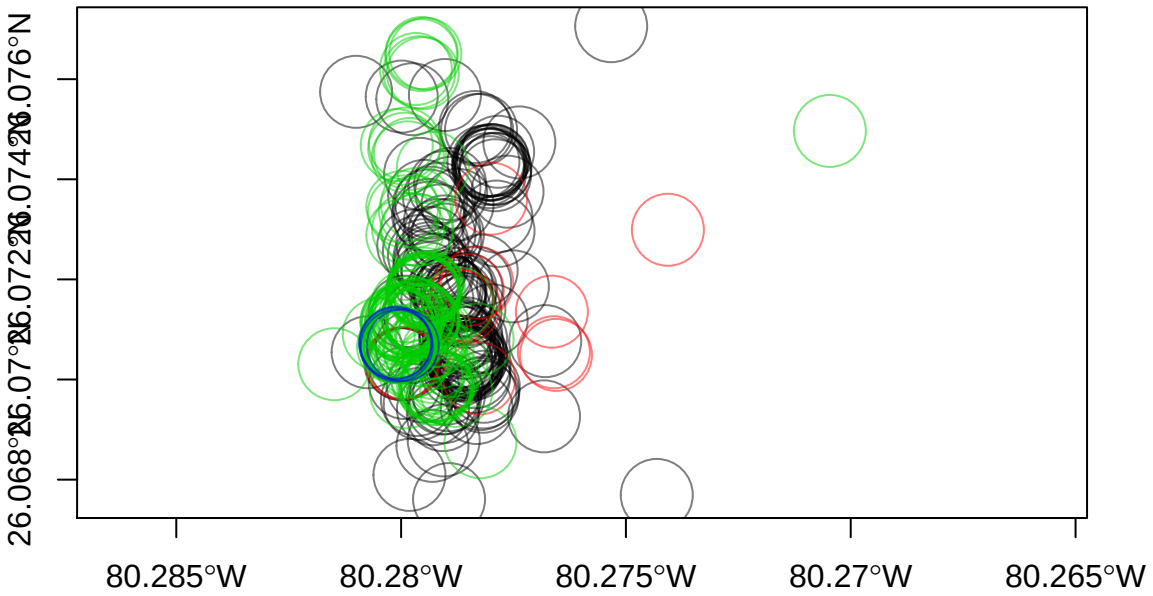
```
## [1] "id"
```

```
plot(my_sftraj)
```



Because it feeds plot an sf object, the same arguments for `plot.sf` are all available.

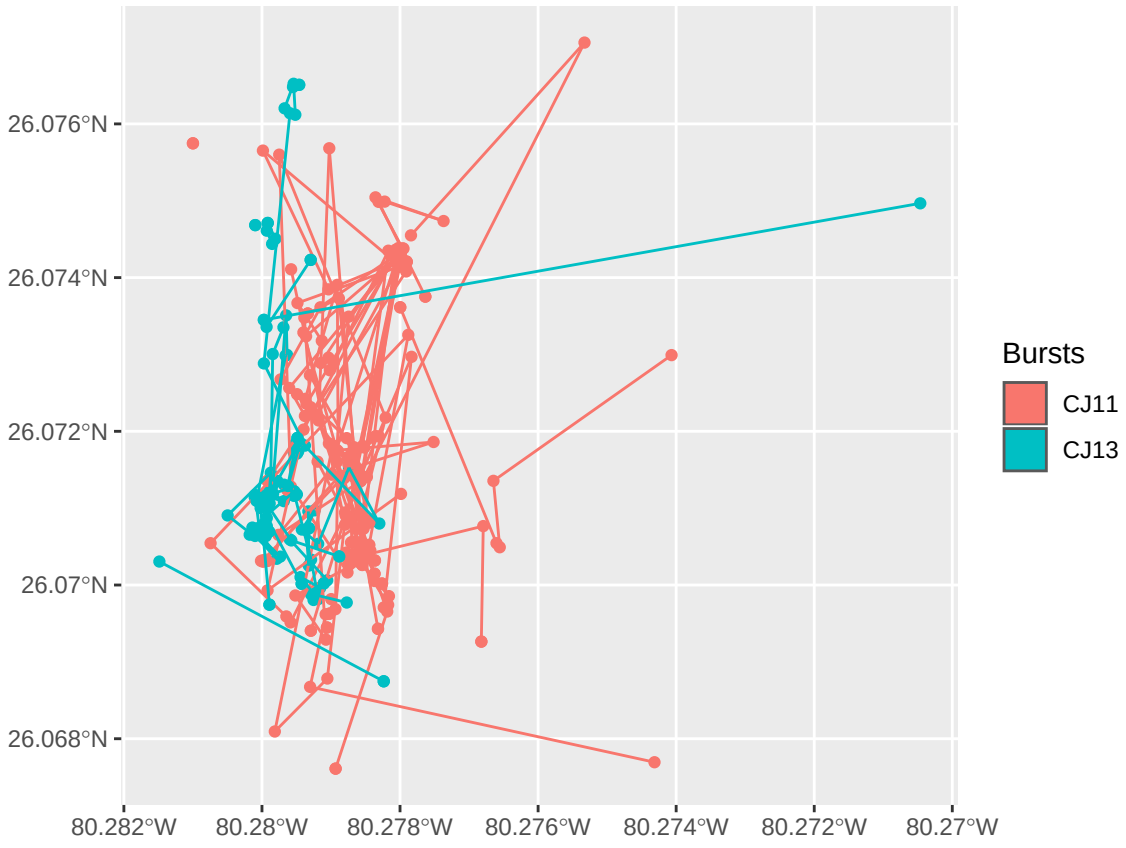
```
plot(my_sftrack, axes = TRUE, cex=5)
```



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 and not into `ggplot()`. Again ggplot assumes active_burst is the grouping variable. Plots vary slightly based on if they're track of traj

```
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