Подгружаем основной пакет для работы. Автоматически подгрузятся зависимости, прежде всего, **`sf`**.

```
        require(ursa)
```

```
Loading required package: ursa
```

```
        isKnitr <- ursa:::.isKnitr()
```

Читаем эксель (CSV) и смотрим структуру, обращая внимание на названия столбцов, с долготой и широтой.

```
        a <- read.csv("Example-data-3_Atlantic.csv")
        str(a)
```

```
'data.frame':   21592 obs. of  17 variables:
 $ shipid        : int  17 18 19 20 21 22 23 24 21 23 ...
 $ date_time_utc : chr  "21/06/2017 03:22" "22/06/2017 07:57" "23/06/2017 11:11" "24/06/2017 06:06"
...
 $ flagname      : chr  "" "" "" "" ...
 $ flagcode      : chr  "" "" "" "" ...
 $ iceclass      : chr  "" "" "" "" ...
 $ sizegroup_MM  : chr  "" "" "" "" ...
 $ astd_cat      : chr  "Unknown" "Unknown" "Unknown" "Unknown" ...
 $ fuelquality   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ fuelcons      : num  NA NA NA NA NA 0.0003 NA 0.00125 NA NA ...
 $ co2           : num  NA NA NA NA NA ...
 $ co            : num  NA NA NA NA NA 2e-06 NA 9e-06 NA NA ...
 $ nox           : num  NA NA NA NA NA 1.33e-05 NA 5.49e-05 NA NA ...
 $ so2           : num  NA NA NA NA NA ...
 $ dist_nextpoint: num  1682.106 175.399 0.888 3.014 0.069 ...
 $ sec_nextpoint : int  378 381 360 371 381 371 362 367 390 362 ...
 $ longitude     : num  -6.7 -6.92 -6.57 -6.56 -6.57 ...
 $ latitude      : num  37.1 37.1 37 37 37 ...
```

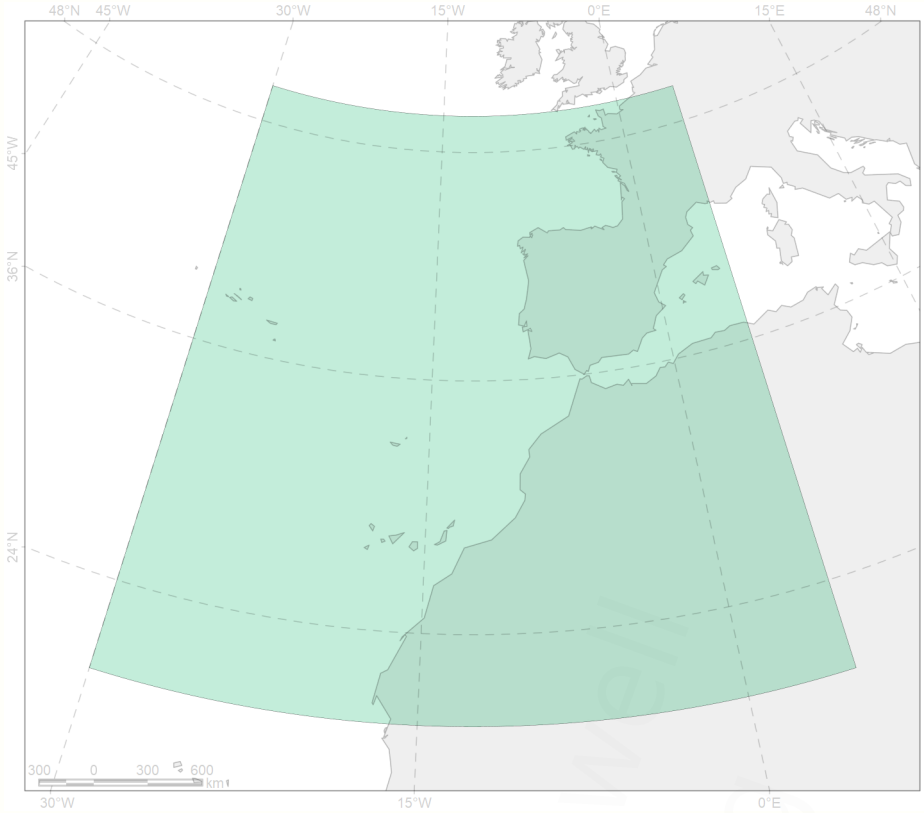
Преобразовываем в пространственный объект (simple features), явно указывая, где долгота (сперва), а где широта (затем).

```
        a <- sf::st_as_sf(a,coords=c("longitude","latitude"),crs=4326)
```

Был запрос обрезать данные по study area. Лучше всего сформировать study area в виде сетки, тогда один пространственный объект для двух задач. Study area под рукой нет, поэтому сгенерируем ее. Потом обрежем по ней исходные данные. Ненужных данных по AIS много, поэтому чем раньше обрезать, тем меньший массив данных обрабатывать

```
        if (T) {
           session_grid(NULL)
           aoi <- ursa:::spatialize(c(-30,20,5,50))
           spatial_count(a) |> print()
           a <- spatial_intersection(aoi,a)
           spatial_count(a) |> print()
           glance(aoi,blank="white",coast.fill="#00000010")
        }
```

```
[1] 21592
Error in geos_op2_geom("intersection", x, y, ...) :
  st_crs(x) == st_crs(y) is not TRUE
[1] 20790
```



Время - во внутренний форма. Сортировка по кораблями и времени регистрации

```
        a$date_time_utc <- as.POSIXct(a$date_time_utc,format="%d/%m/%Y %H:%M",tz="UTC")
        a <- a[with(a,order(shipid,date_time_utc)),]
```

Можно сохранить эти данные для последующей загрузки в QGIS

```
        # spatial_write(a,"Example-data-3_Atlantic.geojson")
```

Возможно, какие-то будут выбросы на сушу. Исключаем их

```
        if (T) {
           land <- spatial_read(file.path("D:/users/platt/shapefile/auxiliary"
                                          ,"naturalearth/5.1.2/10m_physical/ne_10m_land.shp"))
           spatial_count(a) |> print()
           a <- a[!as.logical(sapply(sf::st_intersects(a,land),length)),]
          # a <- a[!a$shipid %in% spatial_intersection(a,land)$shipid,]
           spatial_count(a) |> print()
        }
```

```
[1] 20790
[1] 15720
```

Возможно, нужно будет создавать теплокарты по месяцам. Здесь рассмотрим выделение периода с 05 июня по 25 июня, включительно.

```
        if (T) {
           print(range(a$date))
           a <- a[as.Date(a$date_time_utc)>=as.Date("2017-06-05") &
                  as.Date(a$date_time_utc)<=as.Date("2017-06-25"),]
           print(range(a$date))
        }
```

```
[1] "2017-06-01 04:05:00 UTC" "2017-06-28 23:59:00 UTC"
[1] "2017-06-05 00:00:00 UTC" "2017-06-25 23:56:00 UTC"
```

Возможно, нужно исключить корабли, подававшие сигнал менее 2-3 раз. Здесь оставим лишь те, у кого больше 400 сигналов за наш период

```
        ta <- table(a$shipid)
        ta <- ta[ta>400]
        if (T) {
           print(ta)
           print(spatial_count(a))
           a <- a[as.character(a$shipid) %in% names(ta),]
           print(spatial_count(a))
        }
```

```
389 390 439 547
670 442 479 723
[1] 12925
[1] 2314
```

Создадим траектории из точек. Нужно лишь для визуализации

```
        tr <- segmentize(a,by=a$shipid)
```

Создадим сетку. Допустим, нужна проекция 6931 и ячейка 5 км. Другой способ - это когда исследуемая территория в виде сетки. Если не было фильтрации данных ранее, автоматическая обрезка осуществится на этом шаге.

```
        a <- ursa:::spatialize(a,resetGrid=TRUE,crs=6931)
        #glance(a["shipid"],style="mapnik")
        session_grid(regrid(res=5000,expand=1.25))
        g1 <- ursa_new(NA_integer_,bandname="cell")
        ursa_value(g1) <- seq(prod(dim(g1)))+10000L
```

Раскидывание данных AIS по ячейкам сетки

```
        ursa:::.elapsedTime("intersection -- start")
```

```
*** render.R: intersection -- start: 7.62(7.21) seconds ***
```

```
        b <- spatial_intersection(g1,a)
        ursa:::.elapsedTime("intersection -- finish")
```

```
*** render.R: intersection -- finish: 7.70(0.08) seconds ***
```

```
        str(b)
```

```
Classes 'sf' and 'data.frame':  2314 obs. of  17 variables:
 $ cell          : int  10038 10038 10038 10038 10038 10038 10038 10038 10038 10038 ...
 $ shipid        : int  389 389 389 389 389 389 389 389 389 389 ...
 $ date_time_utc : POSIXct, format: "2017-06-05 19:30:00" "2017-06-05 20:38:00" "2017-06-05
21:21:00" ...
 $ flagname      : chr  "" "" "" "" ...
 $ flagcode      : chr  "" "" "" "" ...
 $ iceclass      : chr  "" "" "" "" ...
 $ sizegroup_MM  : chr  "" "" "" "" ...
 $ astd_cat      : chr  "Unknown" "Unknown" "Unknown" "Unknown" ...
 $ fuelquality   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ fuelcons      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ co2           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ co            : num  NA NA NA NA NA NA NA NA NA NA ...
 $ nox           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ so2           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ dist_nextpoint: num  1.06 2.692 1.352 0.255 4.471 ...
 $ sec_nextpoint : int  361 371 372 362 370 361 361 370 378 392 ...
 $ coords        :sfc_POINT of length 2314; first list element:  'XY' num  -1408300 -6056590
 - attr(*, "sf_column")= chr "coords"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "names")= chr [1:16] "cell" "shipid" "date_time_utc" "flagname" ...
```

Получение характеристик отдельного корабля для каждой ячейки

```
cond <- list(cell=b$cell,shipid=b$shipid)
d <- aggregate(list(duration=b$sec_nextpoint),cond,sum)
d$duration <- d$duration/(24*60*60)
d$firstentry <- aggregate(list(x=b$date_time_utc),cond,min)$x
d$lastentry <- aggregate(list(x=b$date_time_utc),cond,max)$x
d
```

cell	shipid	duration	firstentry	lastentry
10038	389	5.0430903	2017-06-05 19:30:00	2017-06-25 05:45:00
10026	390	1.1941667	2017-06-05 20:31:00	2017-06-19 19:35:00
10038	390	0.6538889	2017-06-06 13:57:00	2017-06-18 04:18:00
10039	390	0.0169907	2017-06-07 13:43:00	2017-06-14 15:58:00
10026	439	2.7166898	2017-06-06 03:28:00	2017-06-25 23:39:00
10023	547	5.2380324	2017-06-07 19:29:00	2017-06-25 23:35:00

Получение характеристик для каждой ячейки по всем кораблям (без весов)

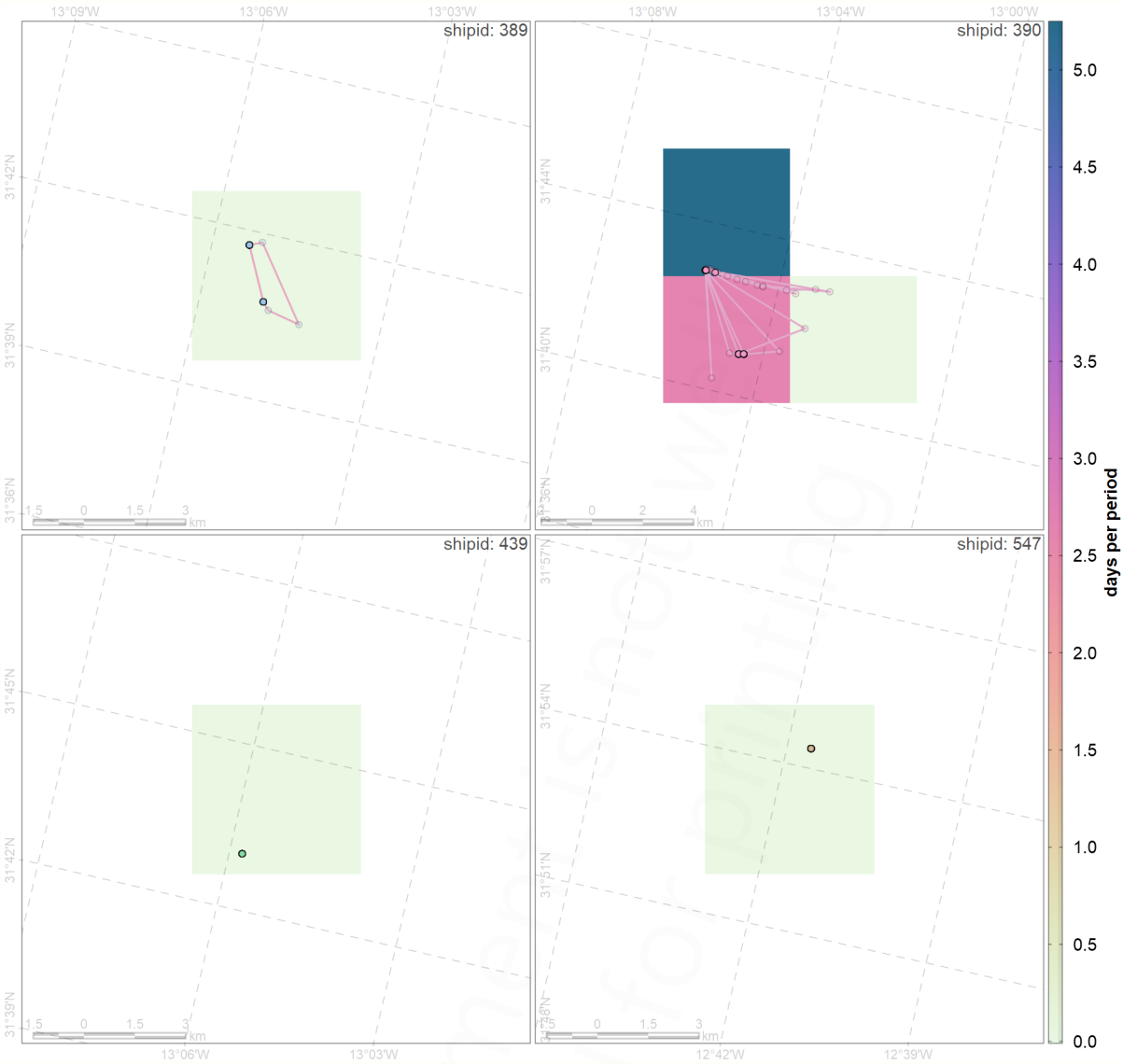
```
d2 <- aggregate(list(duration=d$duration),by=list(cell=d$cell),sum)
d2
```

cell	duration
10023	5.2380324
10026	3.9108565
10038	5.6969792
10039	0.0169907

Визуализация по каждому кораблю

```
ct <- colorize(d$duration,stretch="positive") |> ursa_colortable()
if (FALSE) {
   b <- by(a,a$shipid,\(a2) {
     # print(series(spatial_data(a2)))
      b2 <- allocate(a2["shipid"],fun="n")
      print(b2)
      list(b2)
   }) |> as_ursa()
   b
   #display(b,blank="white",decor=FALSE)
}
dim0 <- c(600,600)
session_grid(dim0)
compose_open(length(ta),legend="right")
if (!isKnitr)
   cat("regionalization")
ret <- by(d,d$shipid,\(ship) {
   if (!isKnitr)
      cat(".")
   a2 <- a[a$shipid %in% ship$shipid,]
   a2 <- a2[order(a2$date_time_utc),]
  # print(spatial_count(a2))
  # print(series(spatial_data(a2[,c("shipid","date_time_utc","sec_nextpoint")])))
  # str(segmentize(a2["date_time_utc"],connect="consequent"))
  # q()
   g2 <- g1
   ind <- ursa_value(g1) %in% ship$cell
   ursa_value(g2)[!ind] <- NA
   ursa_value(g2)[ind] <- ship$duration
   g2 <- ursa_crop(g2,border=1)
   session_grid(consistent_grid(g2,dim0))
   panel_new("white")
   panel_raster(g2,pal=ct)
  # panel_plot(segmentize(a2))
   panel_plot(tr[tr$shipid %in% ship$shipid,])
   panel_plot(a2)
   panel_decor(coast.fill="#00000010")
   panel_annotation(paste("shipid:",as.character(head(ship$shipid,1))),pos="topright")
```

```
  # str(ursa_value(g1) %in% ship$cell)
  # g2 <- g1[g1 %in% ship$cell]
  # print(g2)
})
if (!isKnitr)
   cat(" done!\n")
compose_legend(ct,units="days per period")
compose_close(render=TRUE)
```



Теплокарта для исследуемого периода

```
g3 <- g1
ind <- ursa_value(g1) %in% d2$cell
ursa_value(g3)[!ind] <- NA
ursa_value(g3)[ind] <- d2$duration
g3 <- ursa_crop(g3,border=1)
display(g3,stretch="positive",blank="white",coast.fill="#00000010")
```