VPS prelim analysis

Packages

```
library(ggplot2); library(dplyr); library(sf)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1
```

Data description

Although I'll use the `dplyr` packaged for the rest of this (I assume it's what you're most used to), I'm using `fread` from the `data.table` package to import the CSV detections. We wind up with >42k detections, and `fread` can read that *very* quickly.

```
positions <- data.table::fread(file.path('p:/obrien/biotelemetry/marshyhope/vps',
                                         'VPS-NanticokeRiver-Brookview-01-Results-20210106',
                                         'positions',
                                         'all-calc-positions.csv'),

                               # this just applies base::tolower to the column names
                               col.names = tolower)

positions
```

```
##        transmitter     detectedid            datetime       x      y    d
##     1:   Brookview A69-9006-12727 2020-08-27 16:15:33 1170.32 772.70 0.05
##     2:          S2 A69-1601-60787 2020-08-27 16:17:42  445.53 944.73 4.50
##     3:          N2 A69-1601-60921 2020-08-27 16:18:39  463.48 965.85 4.50
##     4:          S4 A69-1601-60935 2020-08-27 16:19:02  837.89 678.55 3.50
##     5:          S3 A69-1601-60767 2020-08-27 16:20:29  620.17 782.76 2.50
##    ---
## 49176:          N3 A69-1601-60675 2020-11-03 16:14:51  655.03 843.66 3.50
## 49177:          N2 A69-1601-60921 2020-11-03 16:18:33  464.67 971.59 4.50
```

```
## 49178:          N2 A69-1601-60921 2020-11-03 16:28:27  464.88 975.07 4.50
## 49179:          N2 A69-1601-60921 2020-11-03 16:39:10  464.15 968.61 4.50
## 49180:          N2 A69-1601-60921 2020-11-03 16:49:34  464.41 969.30 4.50
##             lat       lon n   hpe hpem temp  depth accel
##     1: 38.57443 -75.78475 1  16.3  7.1   NA 0.7518    NA
##     2: 38.57598 -75.79307 1 121.4 31.5   NA     NA    NA
##     3: 38.57617 -75.79287 1  25.3  8.2   NA     NA    NA
##     4: 38.57358 -75.78857 3  14.4  4.3   NA     NA    NA
##     5: 38.57452 -75.79107 6  11.4  6.9   NA     NA    NA
##    ---
## 49176: 38.57507 -75.79067 2  37.7  4.0   NA     NA    NA
## 49177: 38.57622 -75.79285 1  37.9  2.3   NA     NA    NA
## 49178: 38.57625 -75.79285 1  39.8  1.2   NA     NA    NA
## 49179: 38.57619 -75.79286 1  36.4  5.4   NA     NA    NA
## 49180: 38.57620 -75.79285 1  36.7  4.6   NA     NA    NA
##                                         drx             urx
##     1:                    Brookview N5 N6 S5 Brookview N5 S5
##     2:                        N2 N3 N4 S2 S4        N2 N4 S4
##     3:                        N2 N3 S2 S3 S4        N3 S2 S3
##     4: Brookview N2 N3 N4 N5 N6 S2 S3 S4 S5  N3 N5 N6 S3 S5
##     5:                  N2 N4 N5 S2 S3 S4 S5  N2 N5 S2 S4 S5
##    ---
## 49176:                        N2 N3 S2 S3 S4     N2 S2 S3 S4
## 49177:                        N2 N3 S2 S3 S4        N3 S2 S3
## 49178:                        N2 N3 S2 S3 S4        N3 S2 S3
## 49179:                        N2 N3 S2 S3 S4        N3 S2 S3
## 49180:                           N2 N3 S2 S3        N3 S2 S3
```

We have columns of:

- **transmitter**
  - These are transmitter names. Sync tags will be associated with a station name ("Brookview", "S2", e.g.); animal tags will have the numeric back matter code.
- **detectedid**
  - Full transmitter string
- **datetime**
  - Date and time (yyyy-mm-dd hh:mm:ss) in UTC
- **x/y**
  - The X and Y coordinates in the azimuthal equidistant coordinate system that VEMCO uses to calculate positions
- **d**
- The depth used to calculate the speed of sound. If it is referring to a depth-transponding tag, that depth will be here. Otherwise the depth that VEMCO has assumed the transmitter to be is listed here.
- **lat/lon**
  - The back-transformed X/Y coordinates to longitude and latitude
- **n**
  - The number of triangles used to calculate position

- **hpe**
  - "Horizontal Position Error" this is unitless and self-referential to the system, as the error is **hyperbolic** in shape.
- **hpem**
  - This is the HPE in meters, calculated based off of an assumed known position of the stations. Unfortunately, HPE and HPEm do not correlate as well as we would like them to (see Smith 2013)
- **temp/depth/accel**
  - Values reported by the transmitters
- **drx**
  - Receivers that heard the detection
- **urx**
  - Detections from **drx** that were used to calculate the position

Since I'll use it to filter which detections we want to use, here are the transmitters listed in the data set. Note that, in this case, our VPS stations contain text while fish detections only contain numbers.

```
unique(positions$transmitter)
```

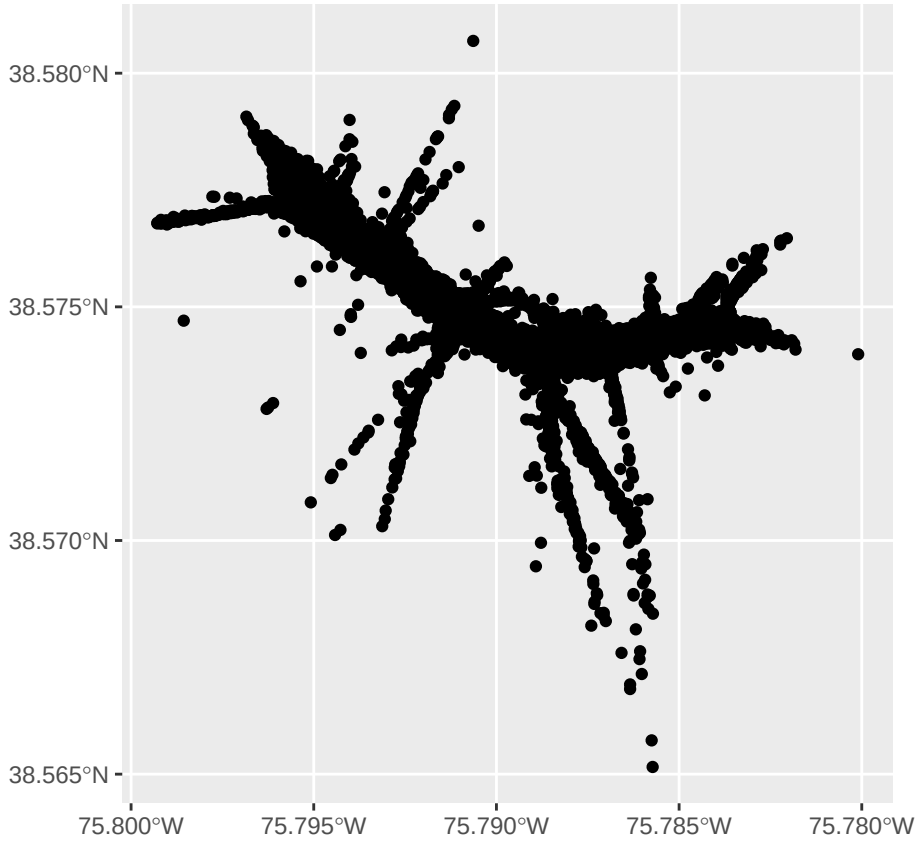
```
##  [1] "Brookview"       "S2"              "N2"              "S4"
##  [5] "S3"              "N6"              "23903"           "S5"
##  [9] "N3"              "S1"              "27543"           "N4"
## [13] "N5"              "26353"           "Above Brookview" "18981"
## [17] "21069"           "21065"           "23904"           "21071"
## [21] "18983"           "26352"           "21909"           "18010"
## [25] "10157"           "Float through"
```

Start mapping

Since this is, inherently, spatial data, I'm going to treat it as such using functions in the `sf` package. The first step is to tell R that this is spatial data.

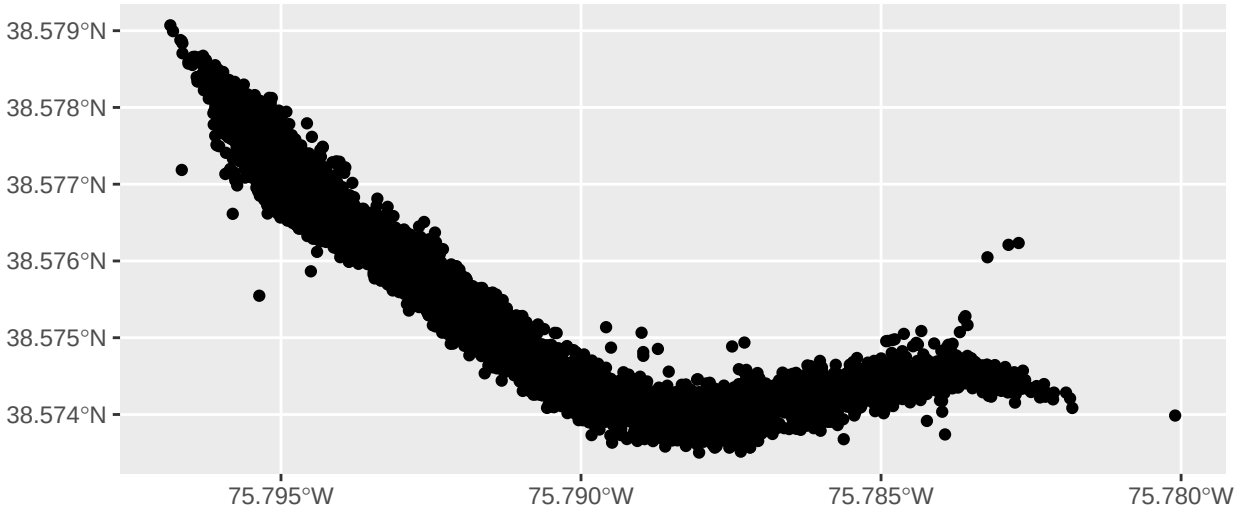
- `st_as_sf(positions`
  - "R, treat the data frame called 'positions' as a spatial object"
- `coords = c('lon', 'lat')`
  - "The X coordinate is called 'lon' and the Y coordinate is called 'lat' "
- `crs = 4326`
  - "The coordinate reference system (CRS) is EPSG code 4326 (i.e., they're GPS coordinates)"

```
positions <- positions %>%
  st_as_sf(coords = c('lon', 'lat'),
           crs = 4326)

ggplot(data = positions) +
  geom_sf()
```



What about just the fish positions? The code below uses regular expressions to filter the data for values in the transmitter column that start with ("^") a number ("\\d").

```
fish <- positions %>%
  filter(grepl('^\\d', transmitter))

ggplot(data = fish) +
  geom_sf()
```



It will be helpful if we have a base map. *However*, the shapefile that we are about to use to map the data is in a different CRS:

```
st_read('data/raw/NHD_H_0208_HU4_GDB.gdb',
        layer = 'nhdarea',
        query = 'SELECT * FROM nhdarea WHERE fid = 1') %>%
  st_crs()
```

```
## Reading layer ‘nhdarea’ from data source ‘C:\Users\darpa2\Analysis\MarshyhopeAS-2019-22\Data\Raw\NHD_
```

```
## Warning in CPL_read_ogr(dsn, layer, query, as.character(options), quiet, :
## GDAL Message 1: organizePolygons() received a polygon with more than 100 parts.
## The processing may be really slow. You can skip the processing by setting
## METHOD=SKIP, or only make it analyze counter-clock wise parts by setting
## METHOD=ONLY_CCW if you can assume that the outline of holes is counter-clock
## wise defined
```

```
## Simple feature collection with 1 feature and 12 fields
## geometry type:  MULTIPOLYGON
## dimension:      XYZ
## bbox:           xmin: -78.92715 ymin: 37.49826 xmax: -78.92676 ymax: 37.49865
## z_range:        zmin: 0 zmax: 0
## geographic CRS: NAD83
```

```
## Coordinate Reference System:
```

```
##   User input: NAD83
##   wkt:
## GEOGCRS["NAD83",
##     DATUM["North American Datum 1983",
##         ELLIPSOID["GRS 1980",6378137,298.257222101,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["unknown"],
##         AREA["North America - NAD83"],
##         BBOX[14.92,167.65,86.46,-47.74]],
##     ID["EPSG",4269]]
```

See the last line, the one that has "EPSG" in it? That's the EPSG number of the CRS that we're going to transform the positions into so that we can map them on top of each other.

```
fish <- fish %>%
  st_transform(4269)
```

Import a big shapefile

The map that we're going to use comes straight from USGS and contains a whole bunch of information on waterways from the Appalachians to the coastal bays. Because of this, I'm going to use some code gymnastics to crop the shapefile before pulling it into memory. This prevents my computer from blowing up.

```
crop_box <- fish %>%
  # Find the bounding box
  st_bbox() %>%
  # Increase the bounds a little
  + c(-0.001, -0.001, 0.001, 0.001) %>%
  # Convert that bounding box to "simple features collection"
  st_as_sfc()

mh_shape <- st_read('data/raw/NHD_H_0208_HU4_GDB.gdb',
                    layer = 'nhdarea',

                    # st_as_text converts the cropping box to "well-known text", which is
                    #   a specific way to format spatial data. wkt_filter imports
                    #   anything inside or touching this box.
                    wkt_filter = st_as_text(crop_box))
```

```
## Reading layer ‘nhdarea’ from data source ‘C:\Users\darpa2\Analysis\MarshyhopeAS-2019-22\Data\Raw\NHD_
## Simple feature collection with 1 feature and 12 fields
## geometry type:  MULTIPOLYGON
```

```
## dimension:      XYZ
## bbox:           xmin: -75.97139 ymin: 38.2467 xmax: -75.55369 ymax: 38.70698
## z_range:        zmin: 0 zmax: 0
## geographic CRS: NAD83
```

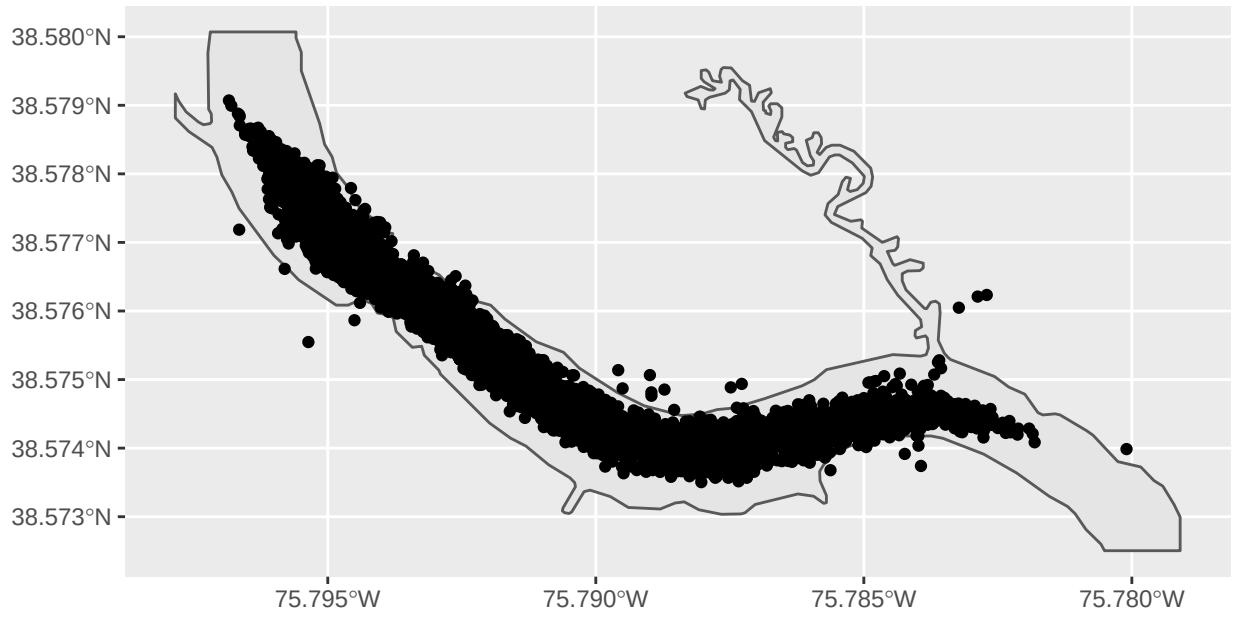
```
# st_crop cuts off anything that was touching, but is outside of, the box.
mh_shape <- mh_shape %>%
  st_crop(crop_box)
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they are planar
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

Now plot them together.

```
ggplot() +
  geom_sf(data = mh_shape) +
  geom_sf(data = fish)
```



Similarly, we will want to import the habitat polygons, which can be rather large (1.6 GB!!).

```
habitat <- st_read('data/raw/2015 Atlantic Sturgeon Habitat Geodatabase Nanticoke and Tributaries 01132
                   layer = 'RiverBed_Habitat_Polygons_CMECS_SC_01132016',
```

```
                  wkt_filter = st_as_text(crop_box %>% st_transform(26918))) %>%
  st_transform(st_crs(mh_shape)) %>%
  st_make_valid() %>%
  st_crop(crop_box)
```

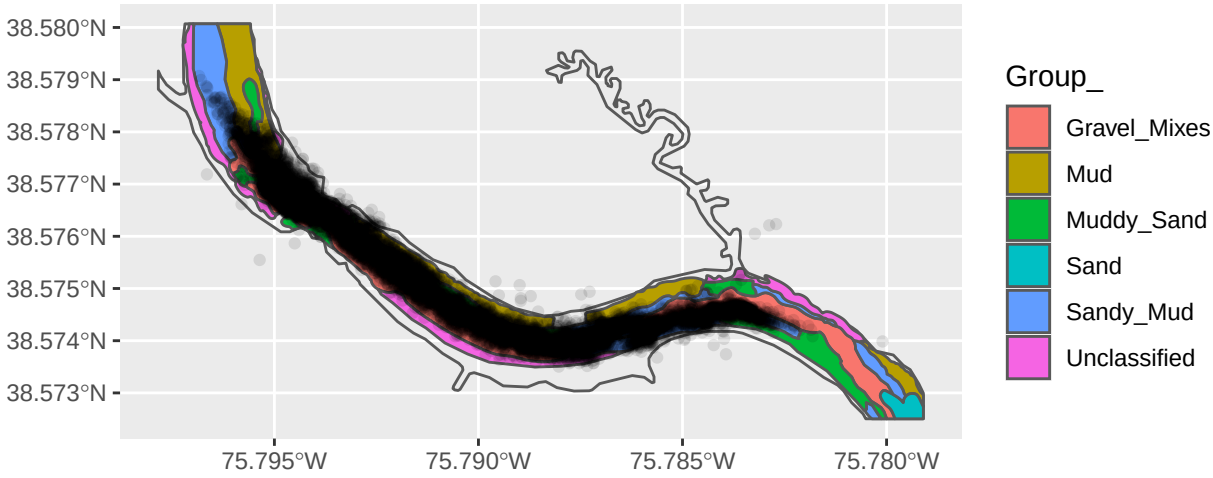
```
## Reading layer 'RiverBed_Habitat_Polygons_CMECS_SC_01132016' from data source 'C:\Users\darpa2\Analysi
## Simple feature collection with 30 features and 16 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 430558.5 ymin: 4269367 xmax: 432186.4 ymax: 4270901
## projected CRS:  NAD83 / UTM zone 18N
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they are planar
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

Huzzah!

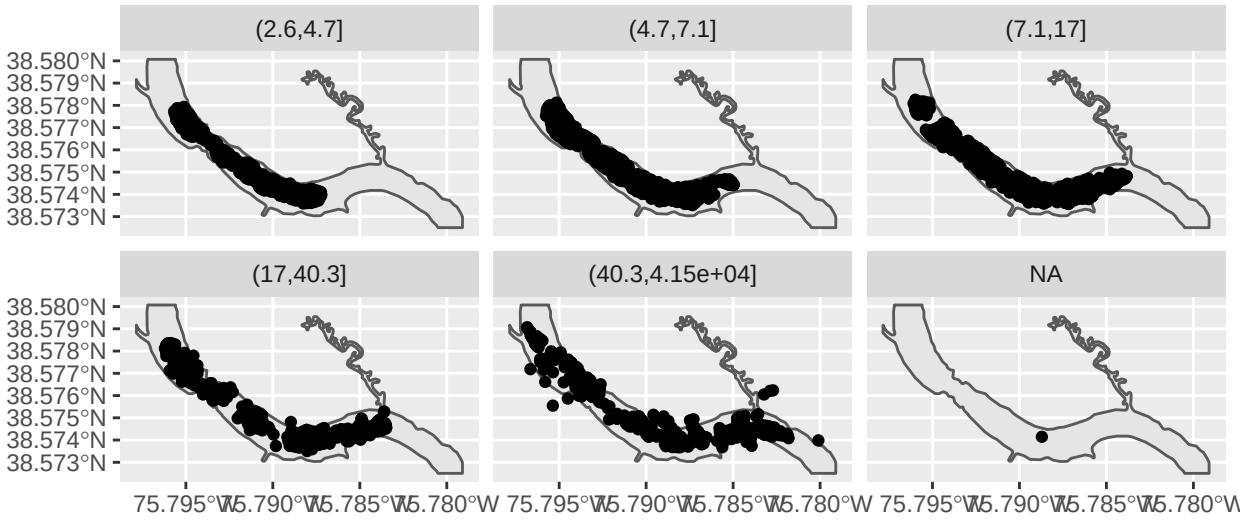
```
ggplot() +
  geom_sf(data = mh_shape, fill = NA) +
  geom_sf(data = habitat, aes(fill = Group_)) +
  geom_sf(data = fish, alpha = 0.1)
```



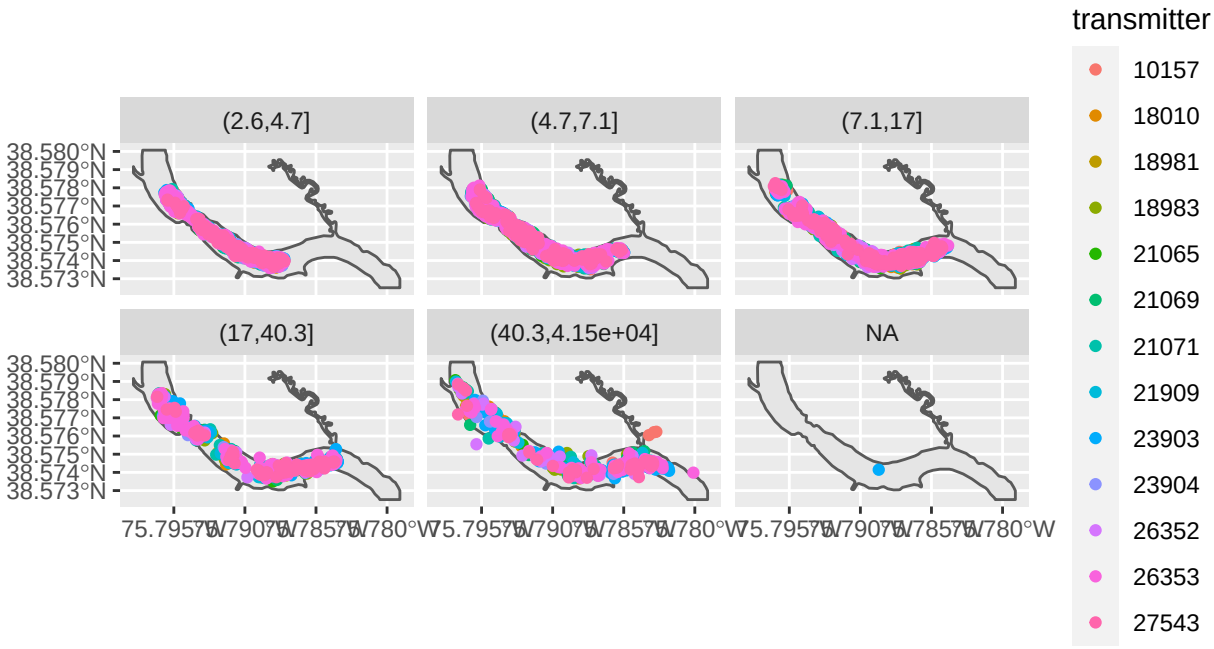
Filtering high-error points

The issue, now, is that some of those points has pretty high error. Since the HPE reported is self-referential, VEMCO suggests creating a cutoff percentile (75%, 80%, 90%, etc.) and selecting all detections under that

```
fish <- fish %>%
  mutate(quant_bin = cut(hpe,
                         breaks =quantile(hpe,
                                  probs = c(0, .25 ,.5, .75, .9, 1)
                                  )
                         ),
         )

ggplot() +
  geom_sf(data = mh_shape) +
  geom_sf(data = fish) +
  facet_wrap(~ quant_bin)
```

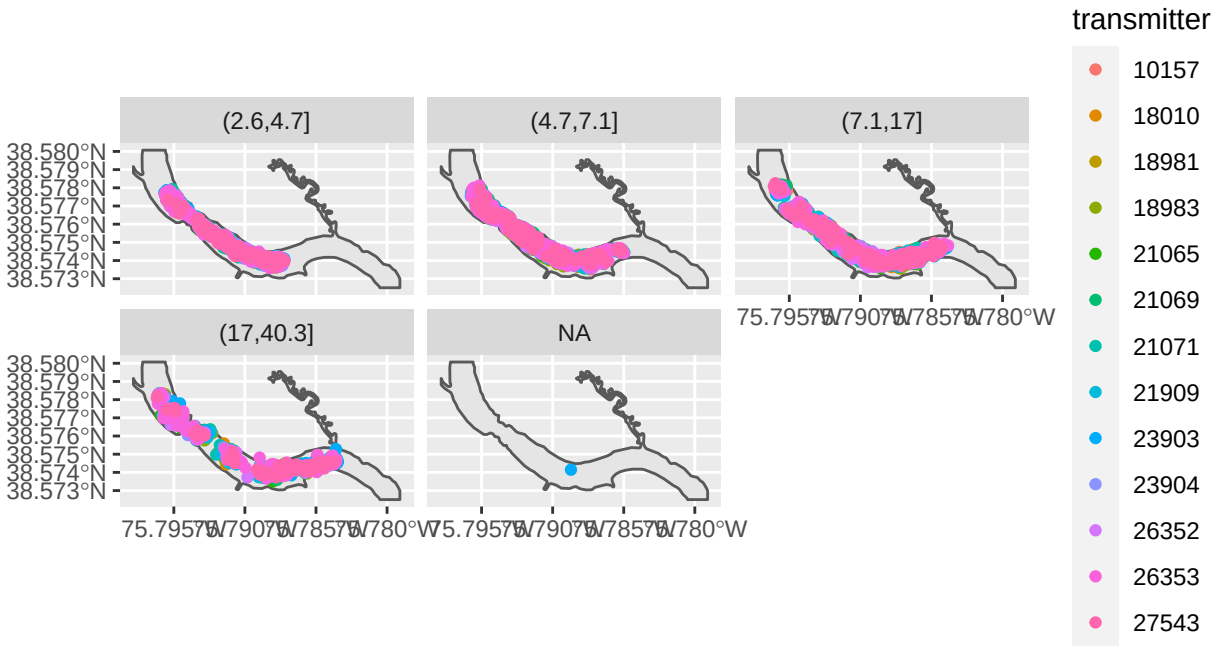



```
ggplot() +
  geom_sf(data = mh_shape) +
  geom_sf(data = fish, aes(color = transmitter)) +
  facet_wrap(~ quant_bin)
```



Drop detections >90th percentile in error. Note that the one random NA is actually HPE = 2.6, the minimum value of HPE. I don't know why the `quantile` function does that.

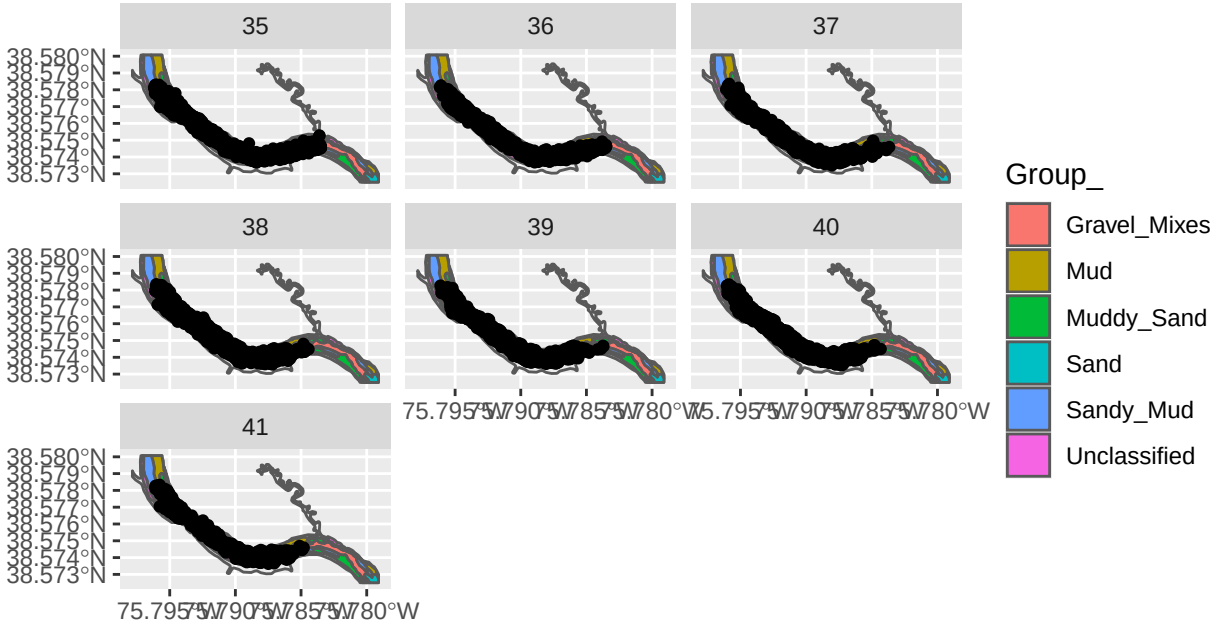
```
fish <- fish %>%
  filter(hpe < 40.3)

ggplot() +
  geom_sf(data = mh_shape) +
  geom_sf(data = fish, aes(color = transmitter)) +
  facet_wrap(~ quant_bin)
```



Start visualizing

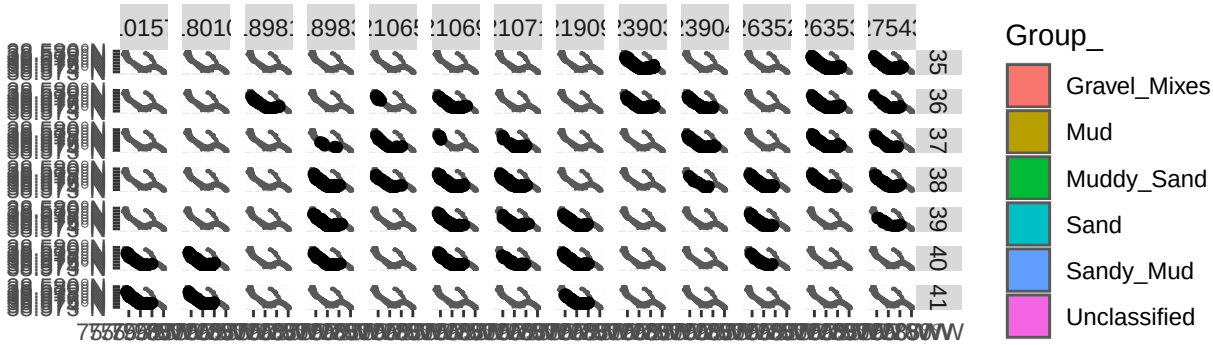
Previously, we liked to look at things by week.

```
fish <- fish %>%
  mutate(wk = lubridate::week(datetime))


ggplot() +
  geom_sf(data = mh_shape, fill = NA) +
  geom_sf(data = habitat, aes(fill = Group_)) +
  geom_sf(data = fish) +
  facet_wrap(~ wk)
```



That's a lot of points. Let's replace with a lot of plots!

```
ggplot() +
  geom_sf(data = mh_shape, fill = NA) +
  geom_sf(data = habitat, aes(fill = Group_)) +
  geom_sf(data = fish) +
  facet_grid(wk ~ transmitter)
```



We'll need a bigger screen for that, but you get the gist.

Spatial analyses

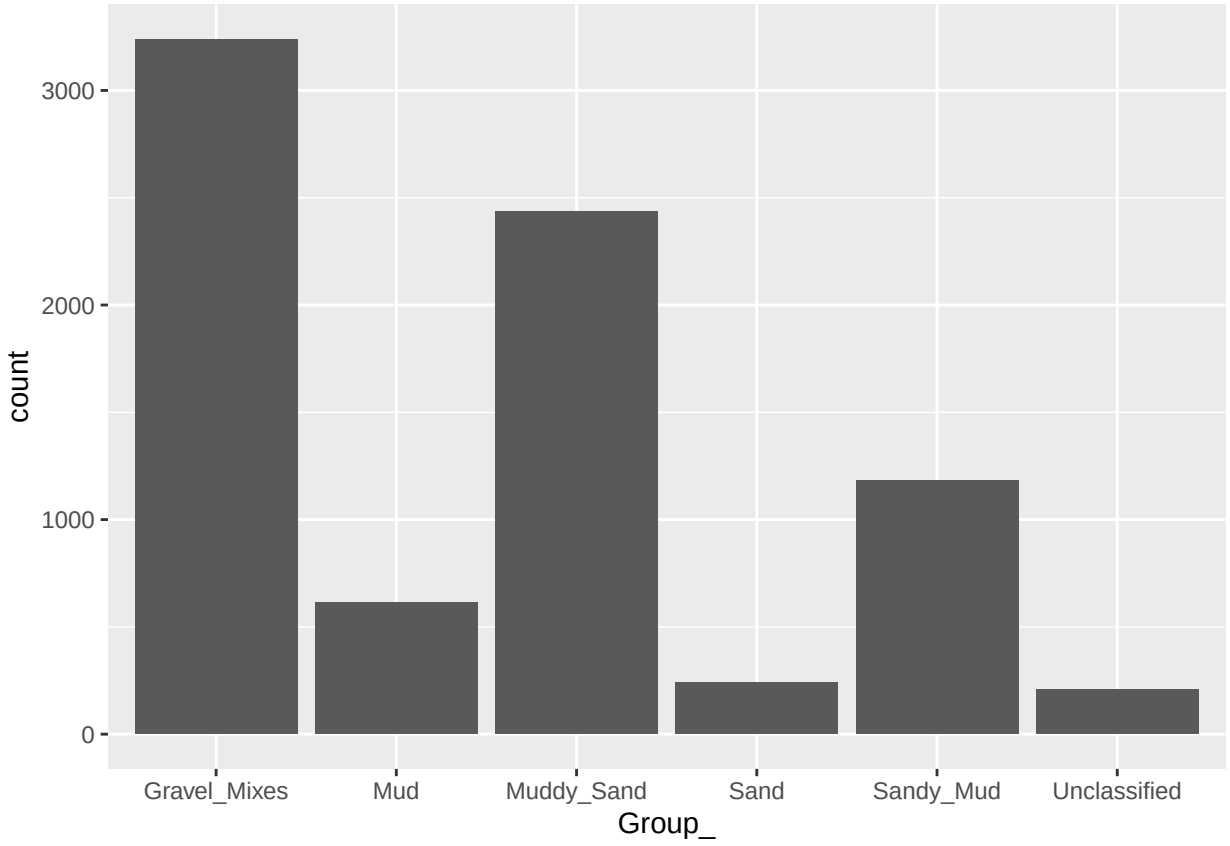
One thing we wanted to do the last time around was point-in-polygon analyses. This just counts the number of points that fall within a given polygon.

```
pip <- fish %>%
  st_intersection(habitat)
```

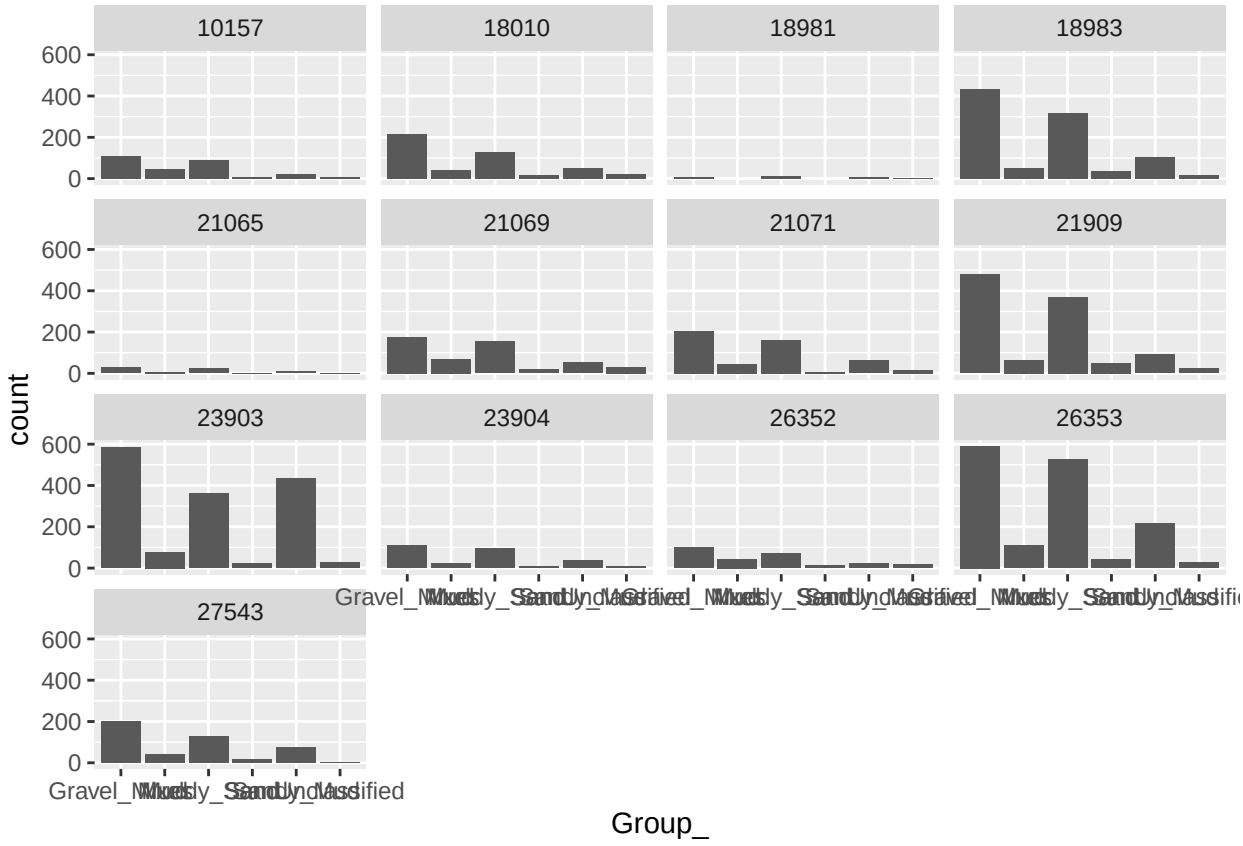
```
## although coordinates are longitude/latitude, st_intersection assumes that they are planar
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
ggplot() +
  geom_bar(data = pip, aes(x = Group_))
```




```
ggplot() +
  geom_bar(data = pip, aes(x = Group_)) +
  facet_wrap(~transmitter)
```



What about per-area?

```
area_agg <- habitat %>%
  group_by(Group_) %>%
  summarize(area = st_area(
    st_union(
      Shape
    )
  )) %>%
  st_drop_geometry()
```

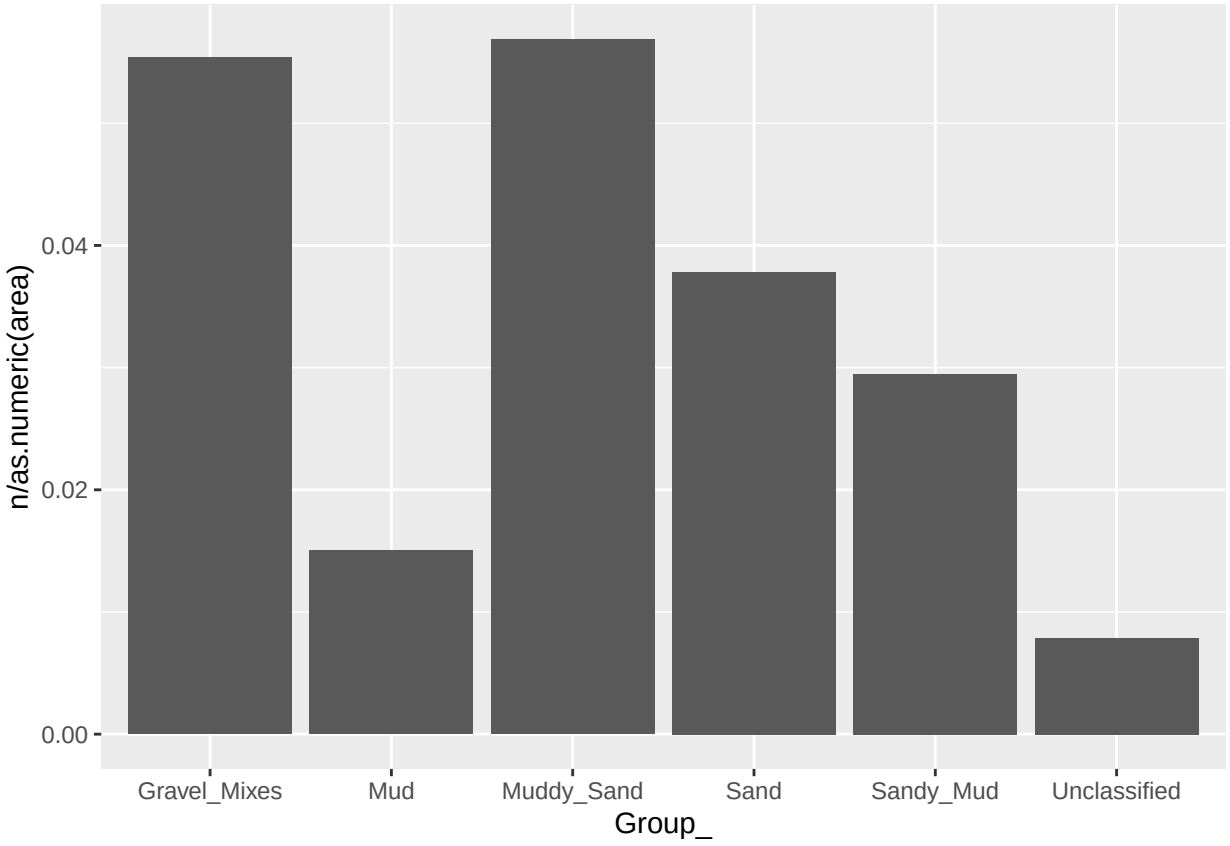
```
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar

## `summarise()` ungrouping output (override with `.groups` argument)

## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
```

```r
pip_scaled <- pip %>%
  group_by(Group_, transmitter) %>%
  summarize(n = n()) %>%
  left_join(area_agg, by = 'Group_')
```

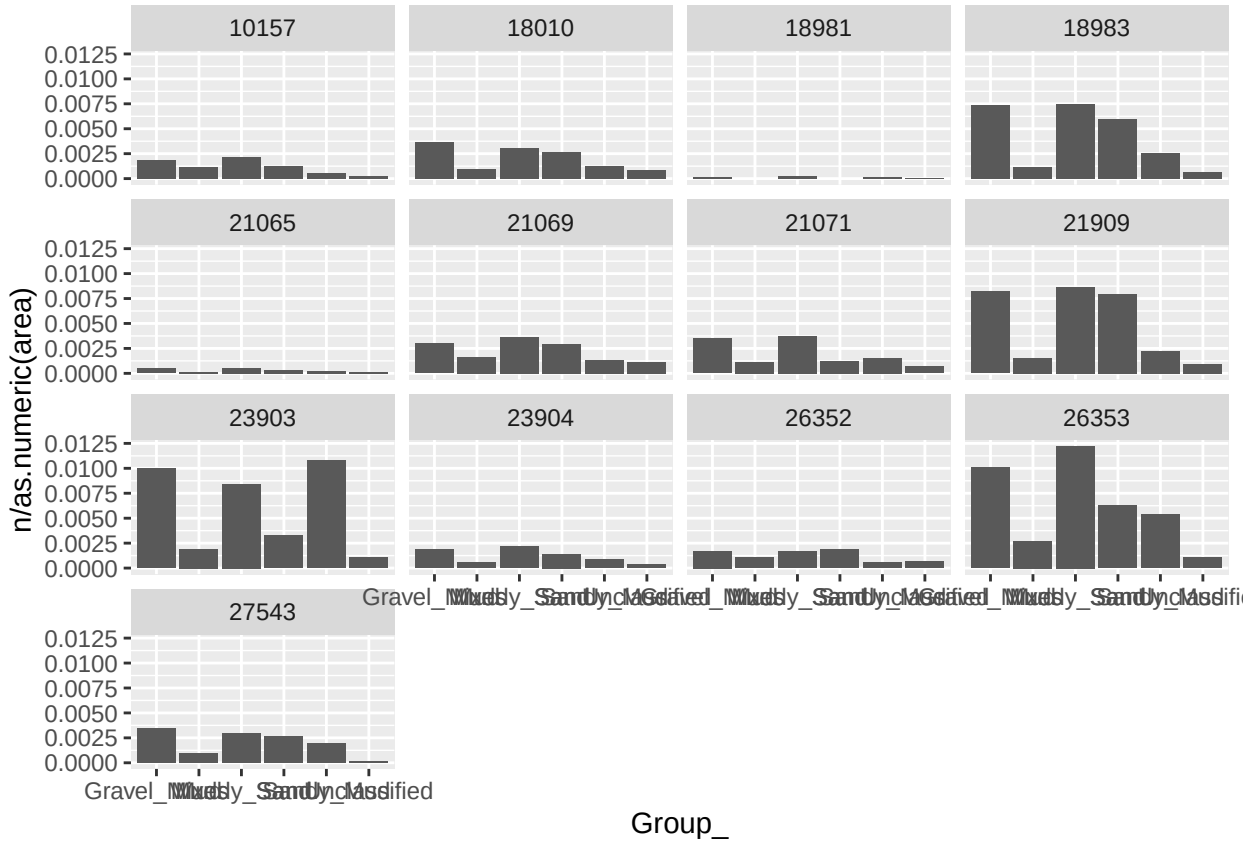
```
## `summarise()` regrouping output by 'Group_' (override with `.groups` argument)
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
```

```
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
## although coordinates are longitude/latitude, st_union assumes that they are planar
```

```
ggplot(data = pip_scaled) +
  geom_col(aes(x = Group_, y = n / as.numeric(area)))
```




```
ggplot(data = pip_scaled) +
  geom_col(aes(x = Group_, y = n / as.numeric(area))) +
  facet_wrap(~ transmitter)
```


10157
18010
18981
18983
21065
21069
21071
21909
23903
23904
26352
26353
27543
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
n/as.numeric(area)
Group_