San Juan de Fuca

Luis Ángel Rodríguez García

25-05-2022

Introduction

Transform the original data set into an array of dimension `c(n, d+1, r)`, where `n` is the time instant, `r` is the number of (latitude and longitude) coordinates and `d+1` are the Cartesian angles which in this cases is equal to 2.

```
# Long data for strait of Juan de Fuca
data("jdf")

# Obtain other data frame

# Transform into coordinates (value formed by lat and lon separated by a comma)
# Remove speed from the dataset
jdf_coor <- jdf %>%
  unite(col = "coordinates", lat, lon, sep = ",", remove = FALSE) %>%
  dplyr::select(-"speed")

# Transform the data frame into a 3-dimensional array
jdf_coor <- abind(split(jdf_coor, jdf_coor$coordinates), along = 3)

# Sum the number of NA's by each row
to_remove <- colSums(t(sapply(seq_len(dim(jdf_coor)[3]),
                              function(k) {
                                is.na(jdf_coor[, 5, k])
                              })))
# Indexes of the rows without NA's
index_to_not_remove <- to_remove == 0

# Select only those rows without NA's
jdf_coor <- jdf_coor[index_to_not_remove, , ]
```

Dimension reduction with psc-SNE

High-dimensional concentrations for a given perplexity

To apply psc-SNE, we first Cartesianize the coordinates to $(\mathbb{S}^1)^r$.

```
# Calculate the Cartesian angles
X_jdf <- sapply(seq_len(dim(jdf_coor)[3]), function(k) {
    DirStats::to_cir(as.numeric(jdf_coor[, 5, k]))
  }, simplify = "array")
dim(X_jdf)
```

```
## [1] 4976    2  105
```

The sample size is $n = 4976$ and the number of $\mathbb{S}^1$'s is $r = 105$. We obtain now the concentrations for the given perplexity in the high dimensional space $(\mathbb{S}^1)^{105}$, a time consuming process.

```
# Get the rho values for a specific perplexity
perplexity <- 30
rho_30 <- rho_optim_bst(x = X_jdf, perp_fixed = perplexity, num_cores = 7)
# Time difference of 1.590534 days
```

We save the data obtained so far to skip these computations in subsequent runs.

```
save(
  list = c("jdf_coor", "index_to_not_remove", "X_jdf", "perplexity", "rho_30"),
  file = paste(here("data-raw", "strait-juan-fuca"), "pscsne.rda", sep = "/")
)
```

Let's load the data objects generated in the previous chunk.

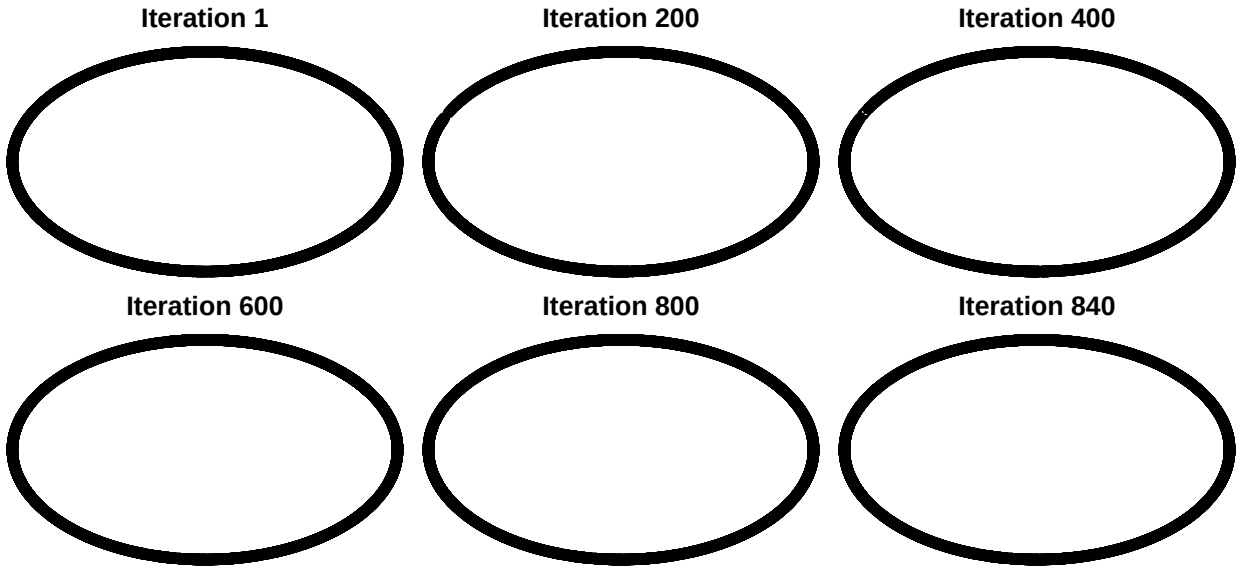
```
# Loading the psc-sne input object
load(paste(here("data-raw", "strait-juan-fuca"), "pscsne.rda", sep = "/"))
```

Reduction to $d = 1$

Now we are ready to reduce the dimension using the polyspherical Cauchy SNE. First, optimized rho is based on a perplexity of 30, $d = 1$ and initializing the resultant reduced data with optimized evenly space points:

```
res_pscsne_1_eq <- psc_sne(X = X_jdf, d = 1, rho_psc_list = rho_30,
                           init = "equispaced", maxit = 1e3)
```

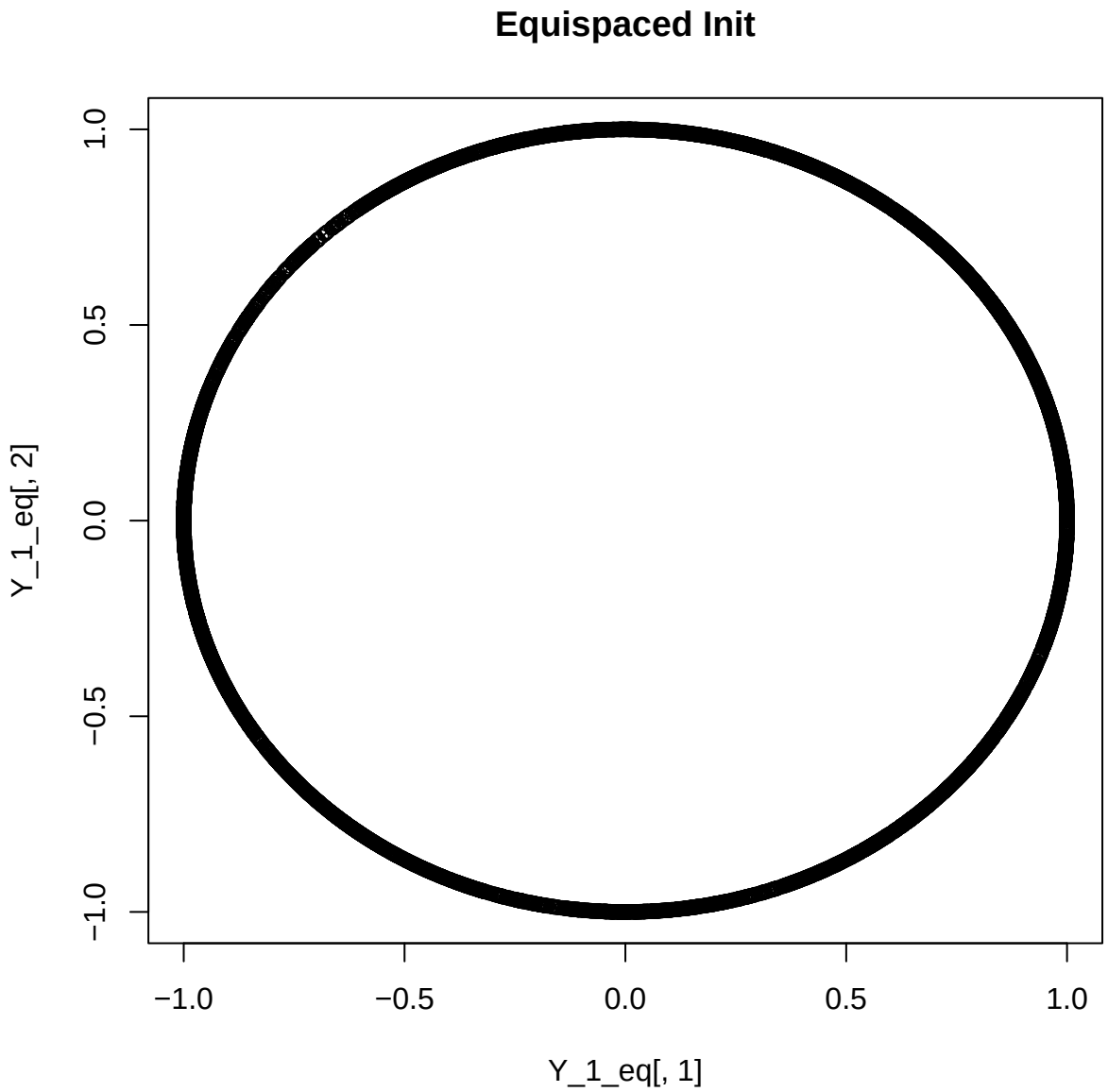
```
## It: 1 (best: 1); obj: 2.48e+01 (best: 2.48e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 3.1e-02; mom: 0.0e+00
## It: 100 (best: 100); obj: 2.09e+01 (best: 2.09e+01); abs: 7.8e-03; rel: 3.7e-04; norm: 1.4e-02; mom: 1.1e+00
## It: 200 (best: 200); obj: 3.76e+00 (best: 3.76e+00); abs: 1.7e-05; rel: 4.4e-06; norm: 2.0e-04; mom: 4.3e-02
## It: 300 (best: 300); obj: 3.75e+00 (best: 3.75e+00); abs: 1.2e-06; rel: 3.2e-07; norm: 3.5e-05; mom: 2.7e-02
## It: 400 (best: 400); obj: 3.75e+00 (best: 3.75e+00); abs: 5.5e-05; rel: 1.5e-05; norm: 2.4e-04; mom: 1.8e-01
## It: 500 (best: 500); obj: 3.74e+00 (best: 3.74e+00); abs: 9.2e-05; rel: 2.5e-05; norm: 3.5e-04; mom: 2.3e-01
## It: 600 (best: 600); obj: 3.74e+00 (best: 3.74e+00); abs: 2.5e-07; rel: 6.8e-08; norm: 1.6e-05; mom: 1.4e-02
## It: 700 (best: 700); obj: 3.74e+00 (best: 3.74e+00); abs: 7.3e-07; rel: 1.9e-07; norm: 2.7e-05; mom: 2.2e-02
## It: 800 (best: 800); obj: 3.74e+00 (best: 3.74e+00); abs: 7.3e-09; rel: 2.0e-09; norm: 2.6e-06; mom: 2.3e-03
## It: 840 (best: 840); obj: 3.74e+00 (best: 3.74e+00); abs: 1.0e-09; rel: 2.8e-10; norm: 9.8e-07; mom: 8.8e-04
## CONVERGENCE!
```




```
Y_1_eq <- res_pscsne_1_eq$best_Y
```

Let's see these results on the circle:

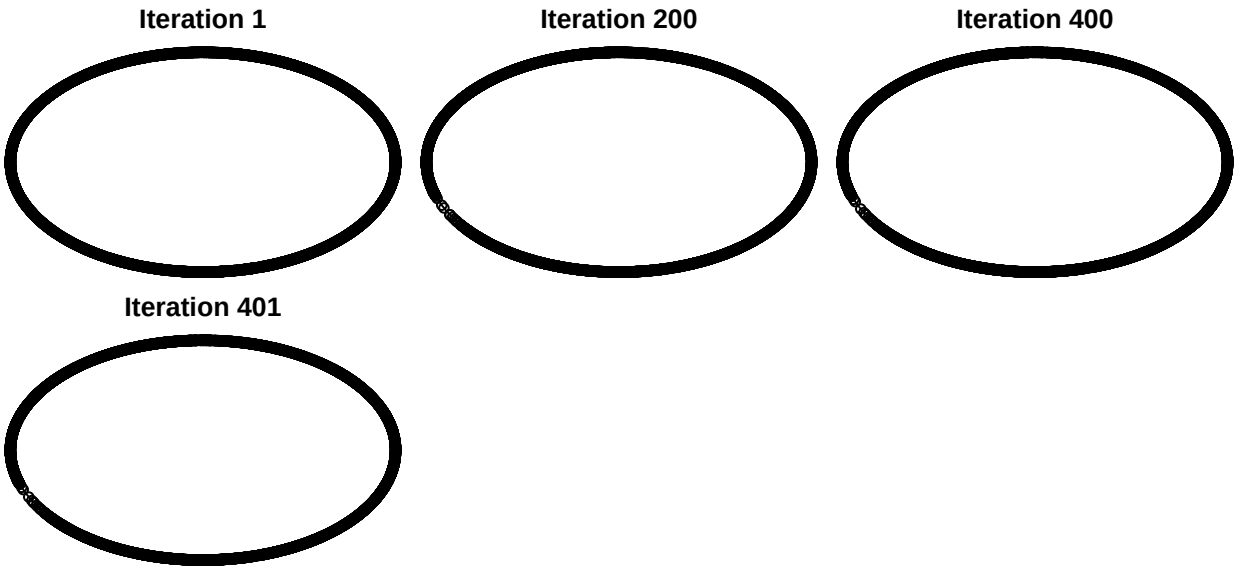
```
plot(Y_1_eq[, 1], Y_1_eq[, 2], xlim = c(-1, 1), ylim = c(-1, 1),
     main = "Equispaced Init")
th <- seq(0, 2 * pi, length.out = 100)
polygon(x = cos(th), y = sin(th))
```



To validate the previous solution we initialize the reduced dimension data with a uniform distribution instead.

```
set.seed(42)
res_pscsne_1_unif <- psc_sne(X = X_jdf, d = 1, rho_psc_list = rho_30,
                             init = "random", maxit = 1e3)
```

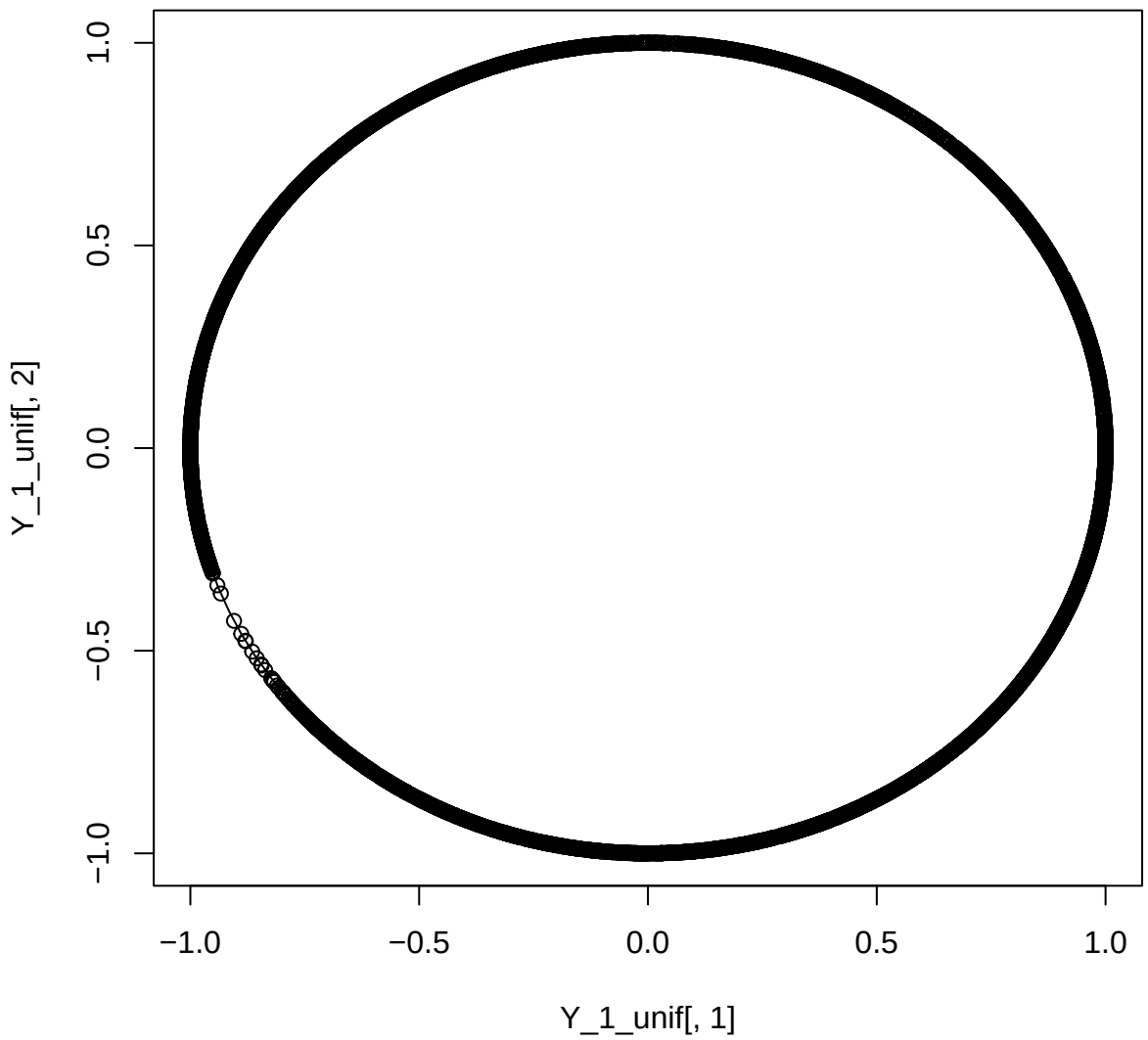
```
## It: 1 (best: 1); obj: 2.54e+01 (best: 2.54e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 2.4e-02; mom: 0.0e+00
## It: 100 (best: 100); obj: 2.08e+01 (best: 2.08e+01); abs: 4.2e-04; rel: 2.0e-05; norm: 1.4e-02; mom: 9.1e-01
## It: 200 (best: 200); obj: 3.76e+00 (best: 3.76e+00); abs: 2.2e-06; rel: 5.9e-07; norm: 7.4e-05; mom: 1.5e-02
## It: 300 (best: 300); obj: 3.76e+00 (best: 3.76e+00); abs: 7.2e-08; rel: 1.9e-08; norm: 8.3e-06; mom: 7.0e-03
## It: 400 (best: 400); obj: 3.76e+00 (best: 3.76e+00); abs: 1.1e-09; rel: 3.0e-10; norm: 1.0e-06; mom: 9.4e-04
## It: 401 (best: 401); obj: 3.76e+00 (best: 3.76e+00); abs: 1.1e-09; rel: 2.8e-10; norm: 9.8e-07; mom: 9.1e-04
## CONVERGENCE!
```




```
Y_1_unif <- res_pscsne_1_unif$best_Y
```

Let's see these results on the circle:

```
plot(Y_1_unif[, 1], Y_1_unif[, 2], xlim = c(-1, 1), ylim = c(-1, 1),
     main = "Uniform Init")
polygon(x = cos(th), y = sin(th))
```



As we can see in the iteration status printing, the `equispaced` initialization obtains slightly better results due to the smaller value of the objective function (3.74 compared to 3.76).

We can not get so many insights from the previous plots, since all the points are around the circumference.

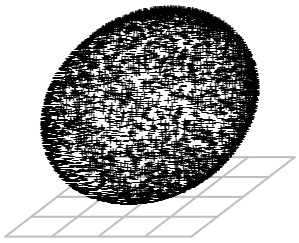
To $d = 2$

Let's see what happens when data is reduced onto the sphere, $d = 2$, of radius 1, with an optimized rho based on a perplexity of 30 and initialize the resultant reduced dimension data with optimized evenly space points:

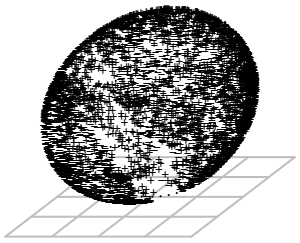
```
res_pscsne_2_eq <- psc_sne(X = X_jdf, d = 2, rho_psc_list = rho_30,
                           init = "equispaced", maxit = 1e3)
```

```
## It: 1 (best: 1); obj: 2.61e+01 (best: 2.61e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 6.3e-02; mom: 0.0e+00
## It: 100 (best: 24); obj: 1.86e+01 (best: 1.82e+01); abs: 1.5e-03; rel: 7.9e-05; norm: 1.5e-01; mom: 8.8e+00
## It: 200 (best: 200); obj: 2.89e+00 (best: 2.89e+00); abs: 1.7e-05; rel: 6.0e-06; norm: 2.1e-04; mom: 4.1e-02
## It: 300 (best: 300); obj: 2.88e+00 (best: 2.88e+00); abs: 8.0e-06; rel: 2.8e-06; norm: 9.0e-05; mom: 7.9e-02
## It: 400 (best: 400); obj: 2.88e+00 (best: 2.88e+00); abs: 4.9e-08; rel: 1.7e-08; norm: 7.0e-06; mom: 5.6e-03
## It: 500 (best: 500); obj: 2.88e+00 (best: 2.88e+00); abs: 1.4e-09; rel: 4.9e-10; norm: 1.2e-06; mom: 9.9e-04
## It: 512 (best: 512); obj: 2.88e+00 (best: 2.88e+00); abs: 1.0e-09; rel: 3.5e-10; norm: 9.9e-07; mom: 8.5e-04
## CONVERGENCE!
```

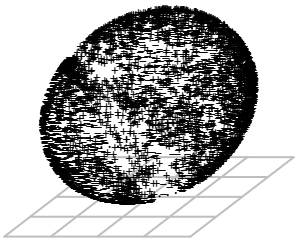
Iteration 1



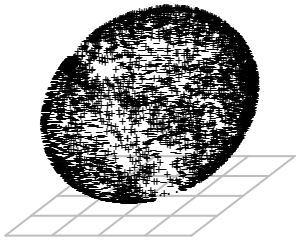
Iteration 200



Iteration 400



Iteration 512

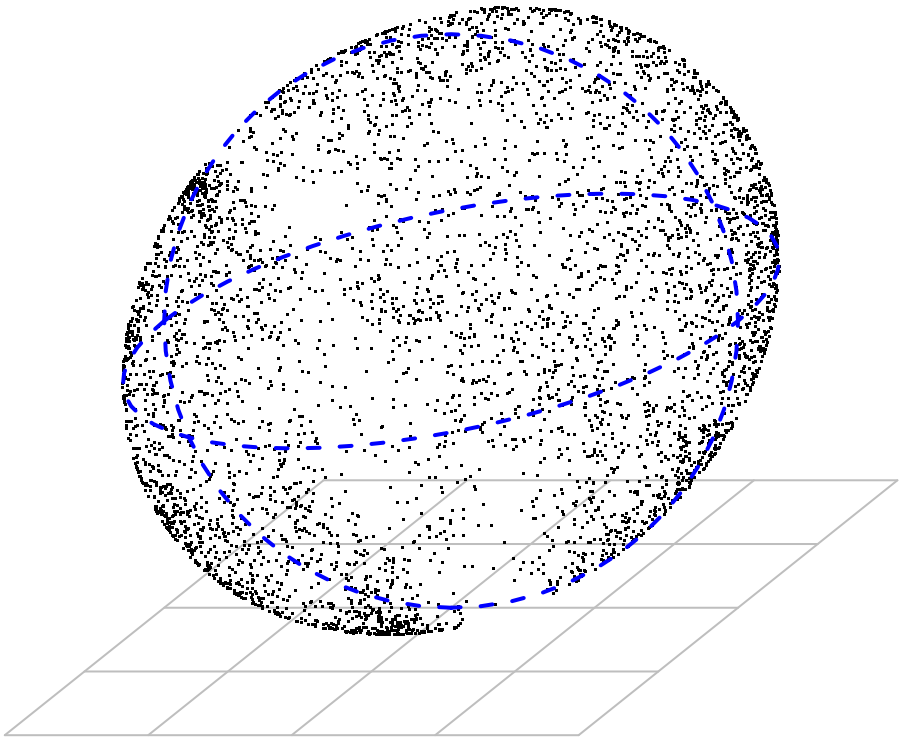


```
Y_2_eq <- res_pscsne_2_eq$best_Y
```

And plot the results onto the sphere:

```
seq_rad <- seq(-pi, pi, by = pi / 30)
meridian <- do.call(rbind, lapply(seq_rad, function(i) c(0, i)))
equator <- do.call(rbind, lapply(seq_rad, function(i) c(i, pi/2)))

sd3 <- scatterplot3d::scatterplot3d(
  Y_2_eq, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", axis = FALSE, main = "Equispaced Init",
  pch = "."
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 2, col = "blue", lwd = 2)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 2, col = "blue", lwd = 2)
```

Equispaced Init

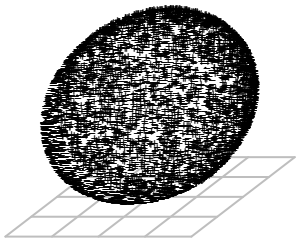


Let's do the same but initializing the reduced dimension data with a uniform distribution. Afterwards, it will be nice to compare both results.

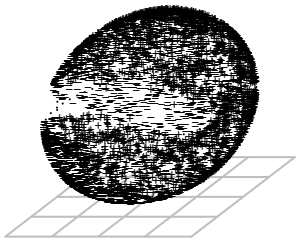
```
set.seed(42)
res_pscsne_2_unif <- psc_sne(X = X_jdf, d = 2, rho_psc_list = rho_30,
                             init = "random", maxit = 1e3)
```

```
## It: 1 (best: 1); obj: 2.69e+01 (best: 2.69e+01); abs: 0.0e+00; rel: 0.0e+00; norm: 4.9e-02; mom: 0.0e+00
## It: 100 (best: 29); obj: 1.85e+01 (best: 1.85e+01); abs: 6.4e-04; rel: 3.4e-05; norm: 1.6e-01; mom: 8.9e+00
## It: 200 (best: 200); obj: 2.87e+00 (best: 2.87e+00); abs: 3.7e-06; rel: 1.3e-06; norm: 9.5e-05; mom: 2.0e-02
## It: 300 (best: 300); obj: 2.87e+00 (best: 2.87e+00); abs: 2.2e-07; rel: 7.6e-08; norm: 1.4e-05; mom: 1.3e-02
## It: 400 (best: 400); obj: 2.87e+00 (best: 2.87e+00); abs: 8.1e-08; rel: 2.8e-08; norm: 9.2e-06; mom: 6.9e-03
## It: 461 (best: 461); obj: 2.87e+00 (best: 2.87e+00); abs: 1.3e-09; rel: 4.4e-10; norm: 9.8e-07; mom: 1.2e-03
## CONVERGENCE!
```

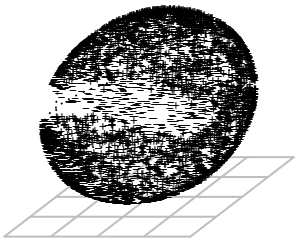
Iteration 1



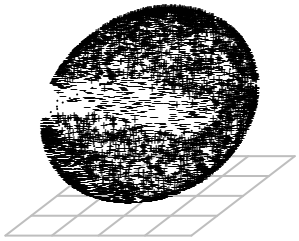
Iteration 200



Iteration 400



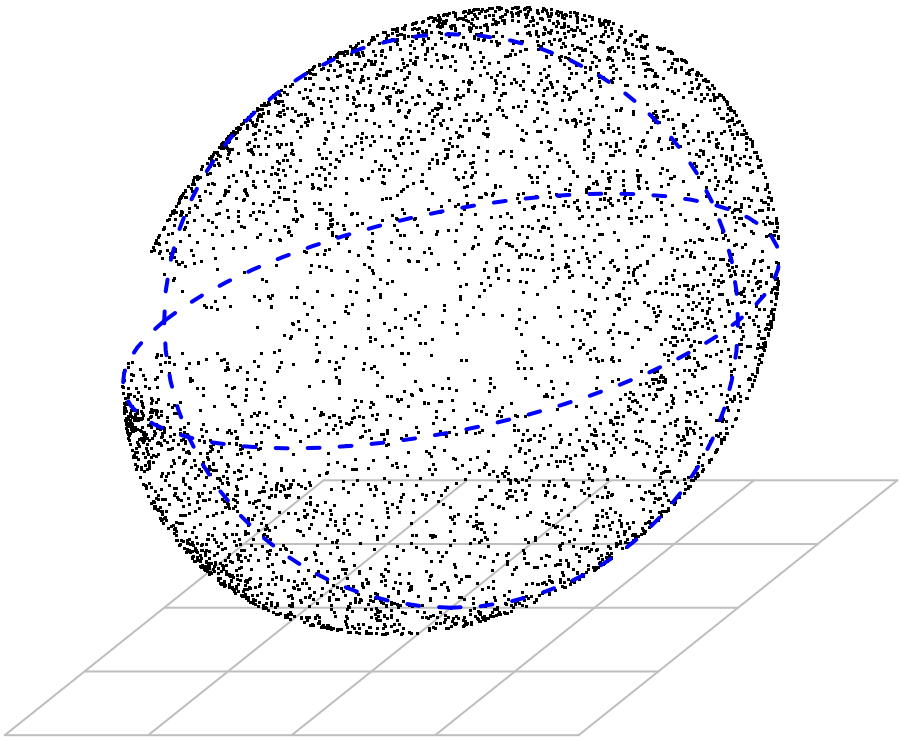
Iteration 461



```
Y_2_unif <- res_pscsne_2_unif$best_Y
```

And plot the results onto the sphere:

```
sd3 <- scatterplot3d::scatterplot3d(
  Y_2_unif, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  xlab = "", ylab = "", zlab = "", axis = FALSE, main = "Uniform Init",
  pch = "."
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 2, col = "blue", lwd = 2)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 2, col = "blue", lwd = 2)
```



As it did not occur for $d = 1$, results are more satisfactory when the initialization of the reduced dimension data points are randomly spread than when they are based on evenly optimized equidistant points. It is easy to see this little difference comparing the objective function value of the `random` initialization (2.87) and the value of the `equispaced` (2.88).

For the following analysis, we are going to consider the object with the lowest objective function for both cases, $d = 1$ and $d = 2$, then the objects `Y_1_eq` and `Y_2_unif`, respectively.

Analysis of the results

Resultant points onto the sphere ($d = 2$) suggest there is a well-defined belt that separates observations. That suggests that there is a potential difference between sea currents directions. Nevertheless, as one could see in the above plot, there is some continuity so the belt is not completely around the sphere. That could be informing the difference between "East" and "West" flows and also on the existence of other intermediate regimes. Let's see what are some rows located in both sides of the belt, to figure out why they are well separated.

First, the kernel mean shift is applied for data on to the circumference.

```
res_kms_dir_Y_1 <- kms_dir(data = Y_1_eq)
```

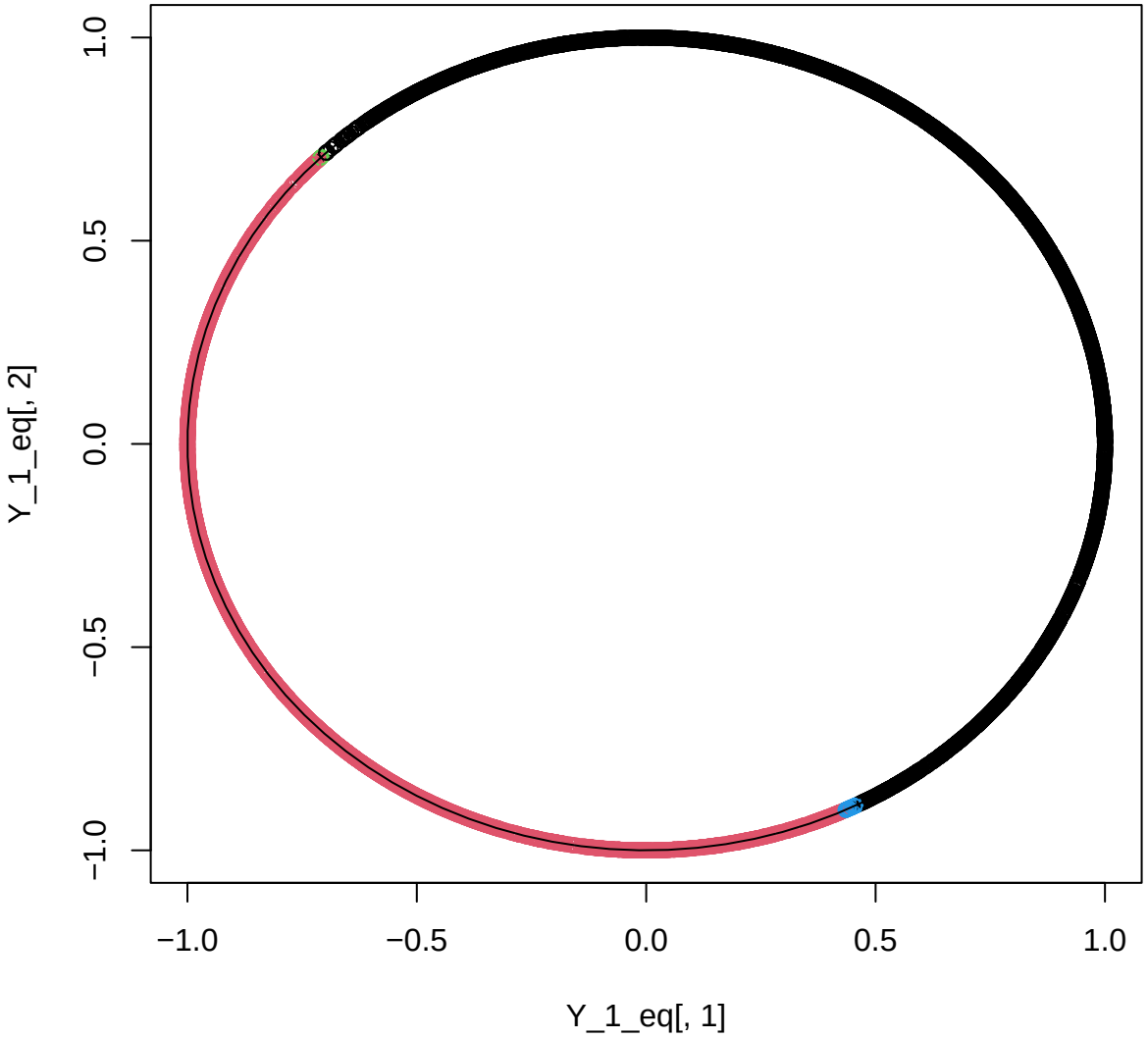
```
##   |                                                                      |
```

```
colors_1d <- res_kms_dir_Y_1$cluster
sprintf("Number of clusters (d = 1) %d", max(colors_1d))
```

```
## [1] "Number of clusters (d = 1) 4"
```

The visualization of the clusters associated visualized in the circumference:

```
plot(Y_1_eq[, 1], Y_1_eq[, 2], col = colors_1d, xlim = c(-1, 1),
     ylim = c(-1, 1))
polygon(x = cos(th), y = sin(th))
```



Once we have got these results, we can conclude that this could be a potential solution since it is identifying two principal directions (west and east) and other two (south and north) that are less frequent.

We do the same for the reduced data on the sphere $d = 2$:

```
res_kms_dir_Y_2 <- kms_dir(data = Y_2_unif)
```

```
##   |                                                                      |
```

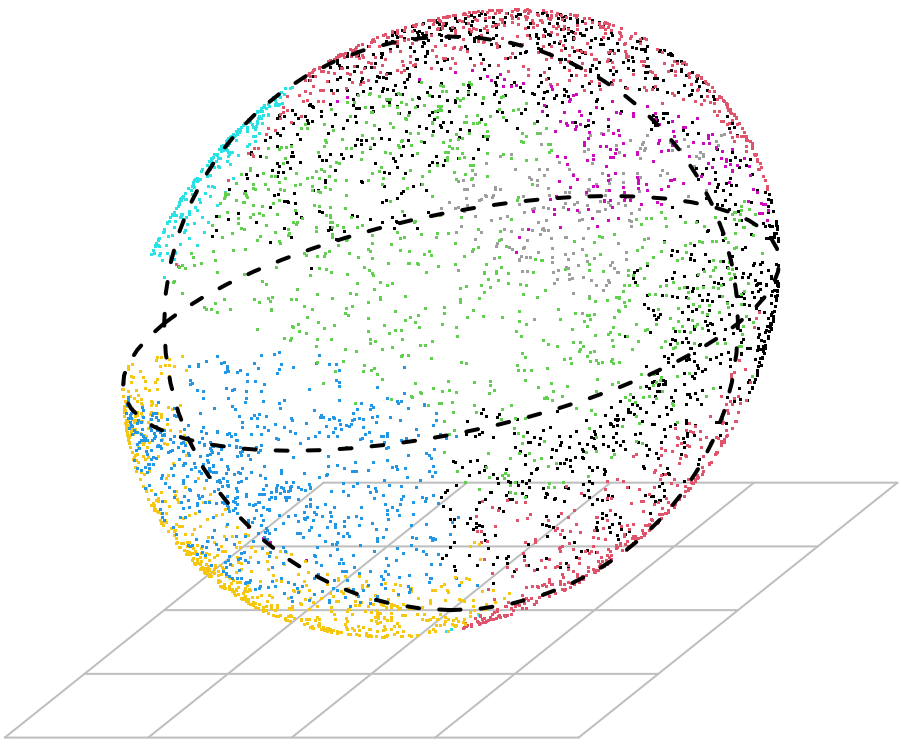
```
colors_2d <- res_kms_dir_Y_2$cluster
sprintf("Number of clusters (d = 2) %d", max(colors_2d))
```

```
## [1] "Number of clusters (d = 2) 19"
```

Let's see the results on the sphere:

```
sd3 <- scatterplot3d::scatterplot3d(
  Y_2_unif, xlim = c(-1, 1), ylim = c(-1, 1), zlim = c(-1, 1),
  color = colors_2d, xlab = "", ylab = "", zlab = "", axis = FALSE,
```

```
  main = "kms clustering", pch = "."
)
sd3$points3d(DirStats::to_sph(th = meridian[, 1], ph = meridian[, 2]),
             type = "l", lty = 2, lwd = 2)
sd3$points3d(DirStats::to_sph(th = equator[, 1], ph = equator[, 2]),
             type = "l", lty = 2, lwd = 2)
```



Many clusters identified, that makes difficult the interpretation of each cluster. The results on the circumference are better in terms of understanding.

Dynamic visualization

Now, it is time to visualize how the current sea evolves for the different instants depending on the group assigned:

```
num_instants <- nrow(jdf_coor)
num_coordinates <- dim(jdf_coor)[3]

jdf_long_fmt_res <- NULL
for (k in 1:num_coordinates) {
  jdf_long_fmt_res <- rbind(jdf_long_fmt_res, jdf_coor[, , k])
}

# Delete the column coordinates
jdf_long_fmt_res <- jdf_long_fmt_res[, -1]

jdf_long_fmt_res <- data.frame(jdf_long_fmt_res) %>%
```

```
  mutate(
    time = as.POSIXct(time, origin = "UTC"),
    lat = as.numeric(lat),
    lon = as.numeric(lon),
    theta = as.numeric(theta)
  ) %>%
  arrange(time)

hours <- 3
instant_seq <- seq(
    from = as.POSIXct("2020-06-01 00:00:00", tz = "UTC"),
    to = as.POSIXct("2022-07-01 00:00:00", tz = "UTC"),
    by = paste(hours, "hours", sep = " ")
  )
instant_seq <- instant_seq[-length(instant_seq)]
instant_seq <- instant_seq[index_to_not_remove]

jdf_long_fmt_strtime <- jdf_long_fmt_res %>%
  mutate(time = as.character(time))

jdf_by_time_res <- abind(
  split(jdf_long_fmt_strtime, jdf_long_fmt_strtime$time), along = 3)

unique_lat <- unique(jdf_long_fmt_res$lat)
unique_lon <- unique(jdf_long_fmt_res$lon)
lat_length <- length(unique_lat)
lon_length <- length(unique_lon)
lat_values <- rev(sort(unique_lat))
lon_values <- sort(unique_lon)

r <- num_instants
n <- num_coordinates

jdf_by_time_latlon <- array(NA,
                            dim = c(lon_length, lat_length,
                                    dim(jdf_by_time_res)[3]))
kms_dir_colors_cir <- array(NA,
                            dim = c(lon_length, lat_length,
                                    dim(jdf_by_time_res)[3]))
kms_dir_colors_sph <- array(NA,
                            dim = c(lon_length, lat_length,
                                    dim(jdf_by_time_res)[3]))

for (k in seq_len(r)) {
  for (i in seq_len(n)) {
    val <- jdf_by_time_res[i, , k]
    lat <- as.numeric(val["lat"])
    lon <- as.numeric(val["lon"])
    index_lat <- which(round(lat_values, digits = 5) == lat)
    index_lon <- which(round(lon_values, digits = 4) == lon)
    theta <- as.numeric(val["theta"])
    jdf_by_time_latlon[index_lon, index_lat, k] <- theta
    kms_dir_colors_cir[index_lon, index_lat, k] <- res_kms_dir_Y_1$cluster[k]
    kms_dir_colors_sph[index_lon, index_lat, k] <- res_kms_dir_Y_2$cluster[k]
  }
}
```

```
colnames(jdf_by_time_latlon) <- lat_values
rownames(jdf_by_time_latlon) <- lon_values

plot_vfield <- function(k, colors, colors_by_time_latlon) {

  loc <- "Juan de Fuca"
  tim_i <- format(instant_seq[k], "%Y-%m-%d %H")
  direction <- colors[k]
  title <- paste(loc, tim_i, "\n", direction)
  OceanView::quiver2D(cos(jdf_by_time_latlon[, , k]),
                      sin(jdf_by_time_latlon[, , k]),
                      x = lon_values,
                      y = lat_values,
                      col = colors_by_time_latlon[, , k], clim = c(0, 1),
                      xlim = c(min(lon_values - 0.015),
                               max(lon_values + 0.015)),
                      ylim = c(min(lat_values - 0.015),
                               max(lat_values + 0.015)),
                      main = title, colkey = FALSE, xlab = "Longitude",
                      ylab = "Latitude")
  points(expand.grid(lon_values, lat_values), pch = 16, cex = 0.5)

}
```

Let's create some videos with the clusters identified by the `kms_dir` in previous sections. First, for those clusters from the scores on the circumference and lastly on the sphere.

```
animation::saveVideo(expr = {
  animation::ani.options(interval = 0.4, ani.res = 1080, ani.width = 1080,
                         ani.height = 1080)
  times_seq <- seq_len(length(instant_seq))
  for (i in times_seq) {
    plot_vfield(i, colors = colors_1d, colors_by_time_latlon = kms_dir_colors_cir)
  }
}, video.name = paste(here("data-raw", "strait-juan-fuca"), "jdfkmscir.mp4",
                      sep = "/"),
    ffmpeg = "/opt/homebrew/opt/ffmpeg@2.8/bin/ffmpeg")
```

The video is stored here: https://www.dropbox.com/s/1a6p8yvhz3kvpwy/jdfkmscir.mp4?dl=0.

Next thing to do is generate the video on the sphere $d = 2$.

```
animation::saveVideo(expr = {
  animation::ani.options(interval = 0.4, ani.res = 1080, ani.width = 1080,
                         ani.height = 1080)
  times_seq <- seq_len(length(instant_seq))
  for (i in times_seq) {
    plot_vfield(i, colors = colors_2d, colors_by_time_latlon = kms_dir_colors_sph)
  }
}, video.name = paste(here("data-raw", "strait-juan-fuca"), "jdfkmssph.mp4",
                      sep = "/"),
ffmpeg = "/opt/homebrew/opt/ffmpeg@2.8/bin/ffmpeg")
```

The video recorded is stored here: https://www.dropbox.com/s/hpk4fxr6u8n7wnf/jdfkmssph.mp4?dl=0.

Now that we have the vector field, we can place each of them on the map associated to the Juan de Fuca area. The regimens correspond to those identified with the `kms_dir` and the dimension $d = 1$ since results are more satisfactory and less cumbersome.

```
jdf_long_fmt_res_uv <- jdf_long_fmt_res %>%
  mutate(
    u = cos(theta),
    v = sin(theta),
    time = as.character(time),
    color = rep(colors, each = dim(jdf_coor)[3])
  )

jdf_by_time_uv <- abind(split(jdf_long_fmt_res_uv, jdf_long_fmt_res_uv$time),
                        along = 3)
jdf_by_time_uv <- jdf_by_time_uv[, -c(3, 4), ]

(map <<- get_map(c(left = min(lon_values) - 0.1,
                   bottom = min(lat_values) - 0.1,
                   right = max(lon_values) + 0.1,
                   top = max(lat_values) + 0.1)))

dput(map, file = "myMaps")

plot_ggvfield <- function(i, df_jdf_uv, colors) {
  loc <- "Juan de Fuca"
  tim_i <- format(instant_seq[i], "%Y-%m-%d %H")
  direction <- colors[i]
  title <- paste(loc, tim_i, "\n", direction)
  ggmap(dget(file = "myMaps"), extent = "panel", geom = "blank", zoom = 0.9,
        maptype = "toner-lite") +
    geom_quiver(mapping = aes(u = u, v = v), color = direction,
                data = df_jdf_uv[[i]], center = TRUE, show.legend = FALSE) +
    ggtitle(title)

  ggsave(paste(here("data-raw", "strait-juan-fuca", "img"),
               paste0("map", i, ".png"), sep = "/"))
}

times_seq <- seq_len(dim(jdf_by_time_uv)[3])

df_jdf_uv <- lapply(times_seq, function(i) {
  data.frame(jdf_by_time_uv[, , i]) %>%
    mutate(
      lon = as.numeric(lon),
      lat = as.numeric(lat),
      u = as.numeric(u),
      v = as.numeric(v)
    ) %>%
    select(lon, lat, u, v)
})

for(i in times_seq) {
  plot_ggvfield(i, df_jdf_uv = df_jdf_uv, colors = colors_1d)
}

# ffmpeg -y -r 1/0.4 -i map%d.png juandefucamap.mp4
```

The video is stored here: https://www.dropbox.com/s/67jesivddi1d1w3/juandefucamap.mp4?dl=0.

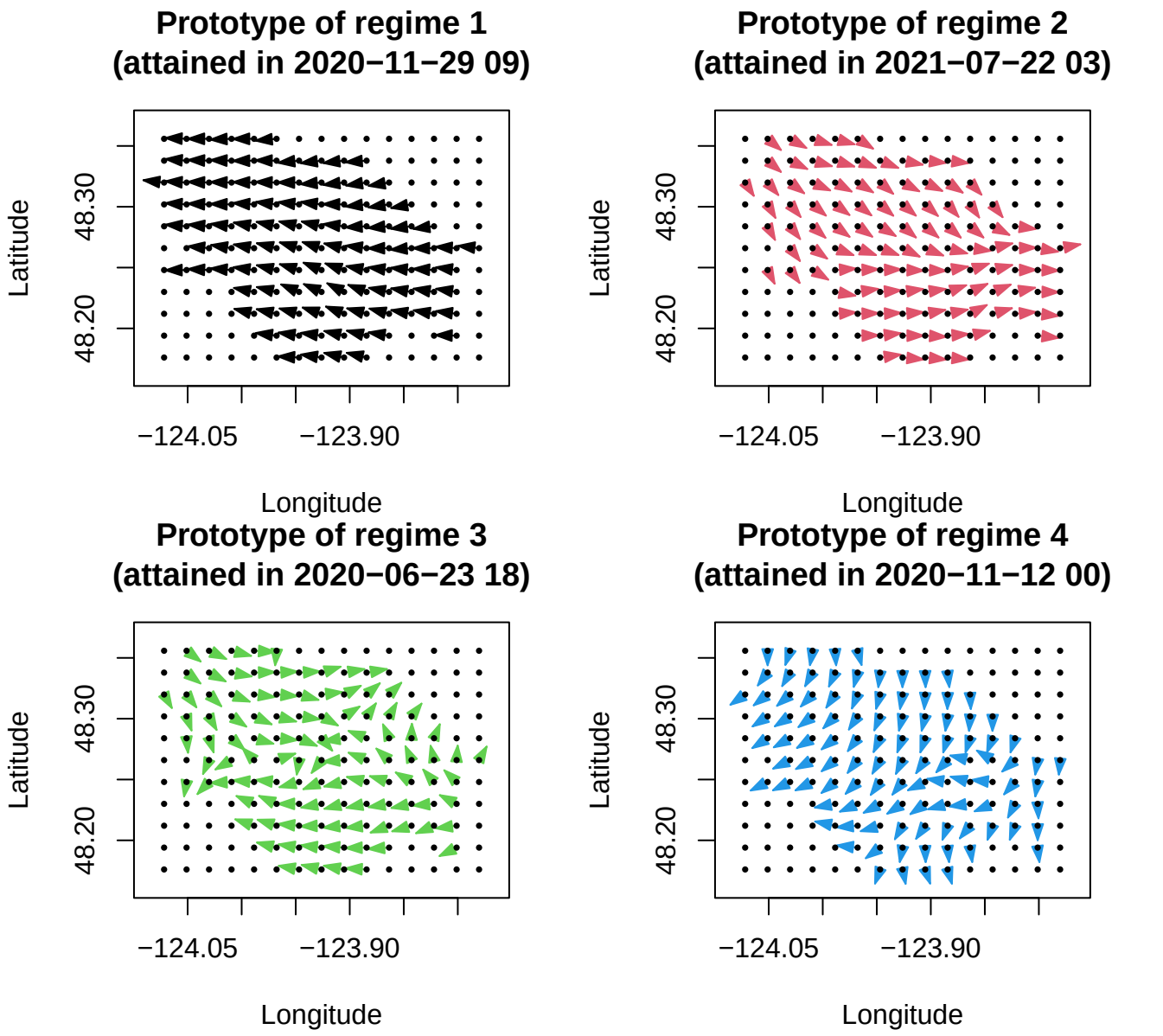
Static visualization

Another possible visualization is to produce a static view of the vector fields with the observations that are more prototypical of different regimes. For that, we first retrieve the $\mathbf{m}_1, \ldots, \mathbf{m}_s \in \mathbb{S}^d$ modes in the psc-SNE scores. Then, we look for the observations $\mathbf{x}_1, \ldots, \mathbf{x}_s \in (\mathbb{S}^1)^r$ whose psc-SNE scores are closer to $\mathbf{m}_1, \ldots, \mathbf{m}_s \in \mathbb{S}^d$ with a numerical search. Then, we plot the s vector fields.

```
# Get prototypical observations of the vector field regimes
proto_obs <- function(X, Y, modes) {

  protos_X <- array(dim = c(nrow(modes), dim(X)[2:3]))
  protos_ind <- numeric(nrow(modes))
  for (i in seq_len(nrow(modes))) {

    protos_ind[i] <- which.min(rowSums(t(t(Y) - modes[i, ])^2))
    protos_X[i, , ] <- X[protos_ind[i], , ]

  }
  return(list("protos_X" = protos_X, "protos_ind" = protos_ind))

}

# Get regimes for d = 1, 2
proto_1 <- proto_obs(X = X_jdf, Y = Y_1_eq, modes = res_kms_dir_Y_1$modes)
proto_2 <- proto_obs(X = X_jdf, Y = Y_2_unif, modes = res_kms_dir_Y_2$modes)

k_proto_1 <- proto_1$protos_ind
k_proto_2 <- proto_2$protos_ind

# Plotting function
plot_vfield2 <- function(k, reg, col = 1) {

  loc <- paste("Prototype of regime", reg)
  tim_i <- format(instant_seq[k], "%Y-%m-%d %H")
  title <- paste0(loc, "\n(attained in ", tim_i, ")")
  OceanView::quiver2D(cos(jdf_by_time_latlon[, , k]),
                      sin(jdf_by_time_latlon[, , k]),
                      x = lon_values,
                      y = lat_values,
                      col = col, clim = c(0, 1),
                      xlim = c(min(lon_values - 0.015),
                               max(lon_values + 0.015)),
                      ylim = c(min(lat_values - 0.015),
                               max(lat_values + 0.015)),
                      main = title, colkey = FALSE, xlab = "Longitude",
                      ylab = "Latitude")
  points(expand.grid(lon_values, lat_values), pch = 16, cex = 0.5)

}
```

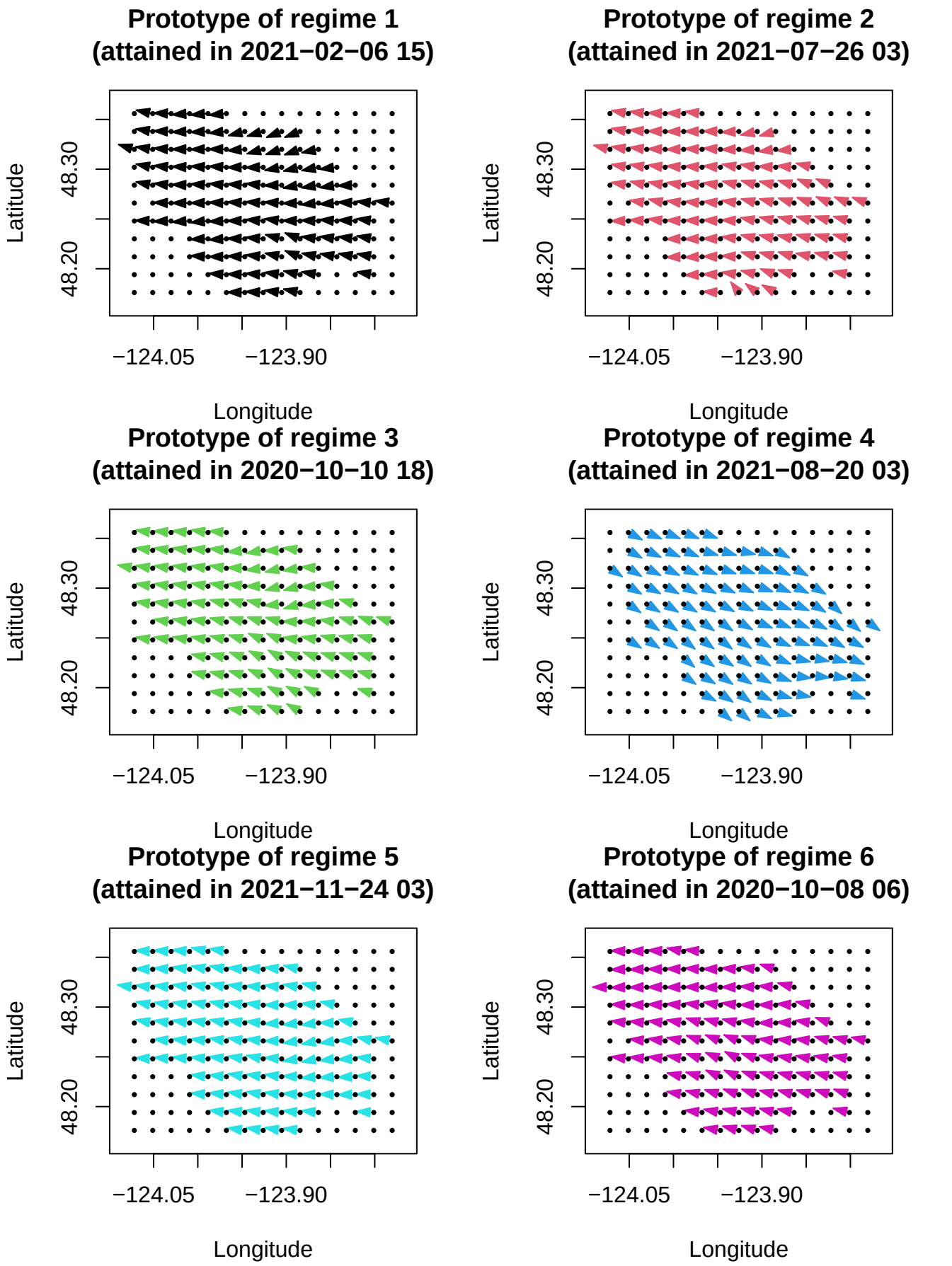
The instants that are closely related to the regimens are going to be shown. First, for $d = 1$:

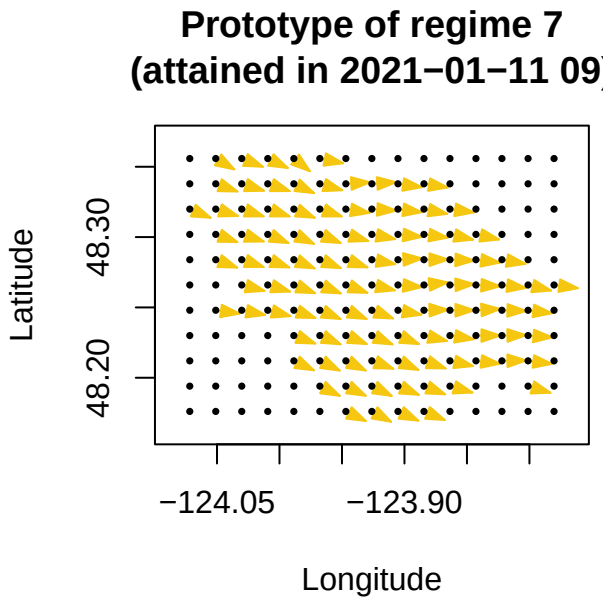
```
# Plots
old_par <- par(mfrow = c(1, 2))
for (reg in seq_along(k_proto_1)) {
  plot_vfield2(k = k_proto_1[reg], reg = reg, col = reg)
}
```

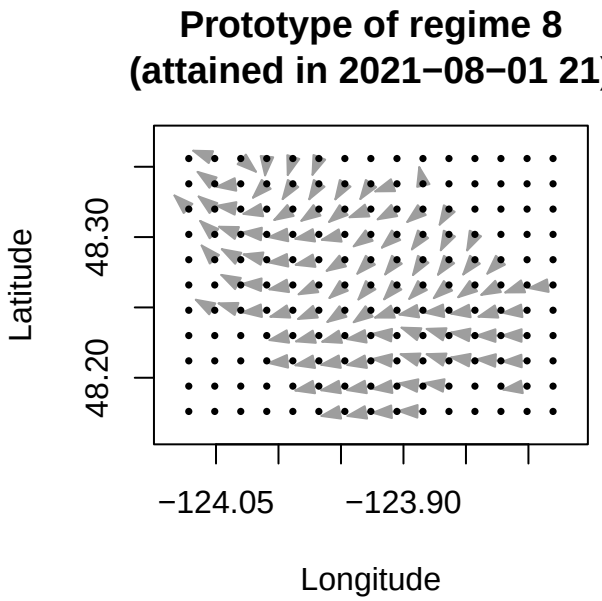


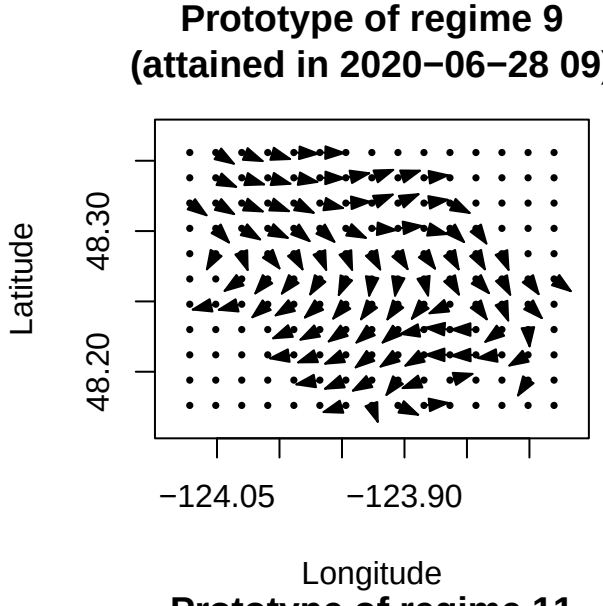
Now, it is time to see what are the closer instants for $d = 2$:

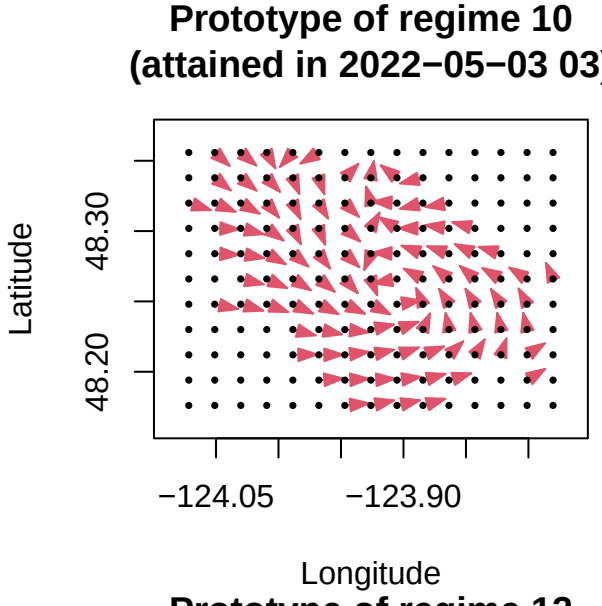
```
par(mfrow = c(1, 2))
for (reg in seq_along(k_proto_2)) {
  plot_vfield2(k = k_proto_2[reg], reg = reg, col = reg)
}
```

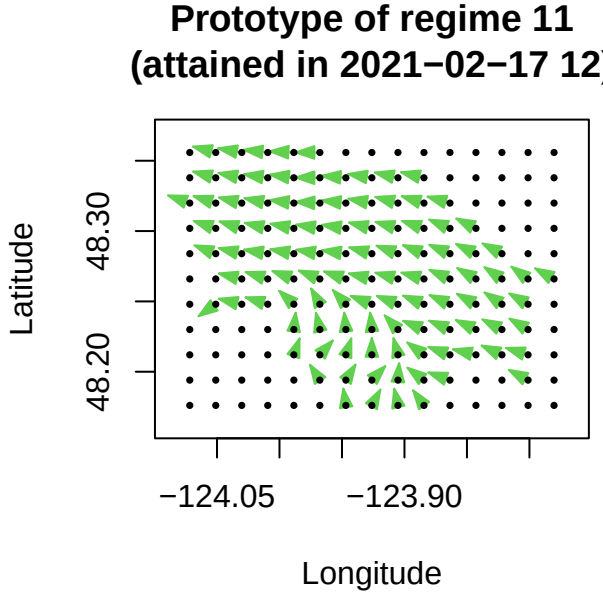

Prototype of regime 1
(attained in 2021−02−06 15)
Prototype of regime 2
(attained in 2021−07−26 03)
Prototype of regime 3
(attained in 2020−10−10 18)
Prototype of regime 4
(attained in 2021−08−20 03)
Prototype of regime 5
(attained in 2021−11−24 03)
Prototype of regime 6
(attained in 2020−10−08 06)
Latitude
48.30
48.20
−124.05
−123.90
Longitude

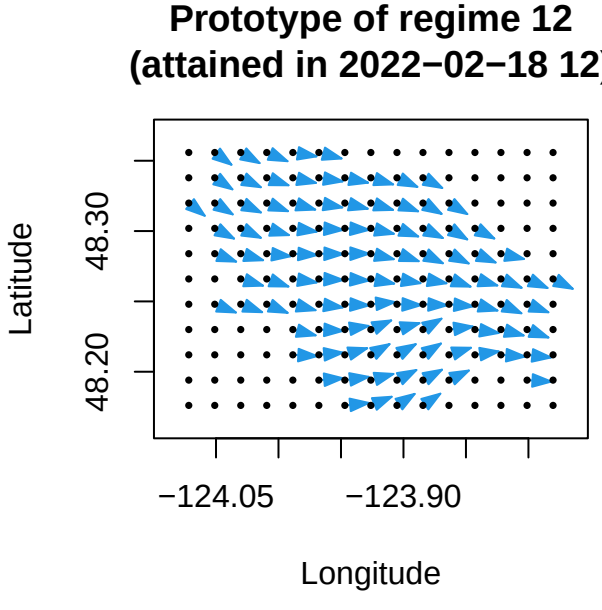

Prototype of regime 7
(attained in 2021−01−11 09)
Latitude
48.30
48.20
−124.05
−123.90
Longitude

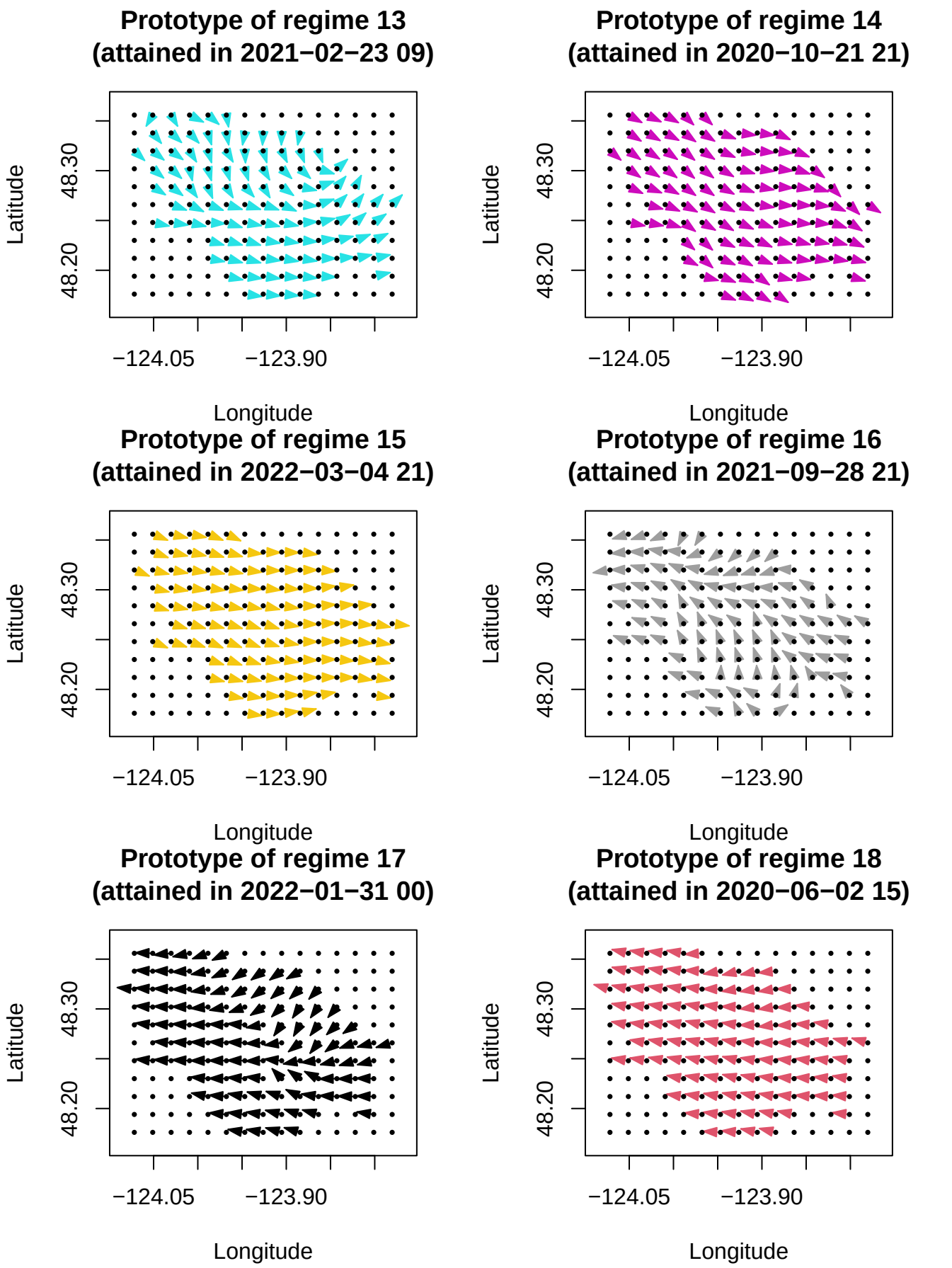

Prototype of regime 8
(attained in 2021−08−01 21)
Latitude
48.30
48.20
−124.05
−123.90
Longitude

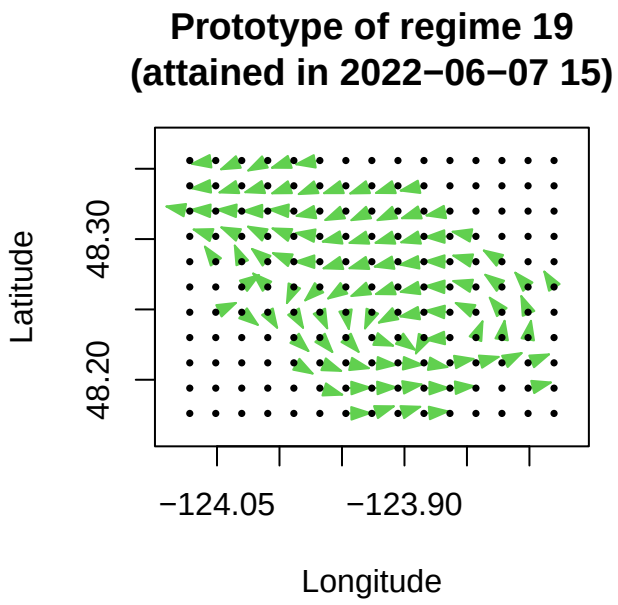

Prototype of regime 9
(attained in 2020−06−28 09)
Latitude
48.30
48.20
−124.05
−123.90
Longitude


Prototype of regime 10
(attained in 2022−05−03 03)
Latitude
48.30
48.20
−124.05
−123.90
Longitude


Prototype of regime 11
(attained in 2021−02−17 12)
Latitude
48.30
48.20
−124.05
−123.90
Longitude


Prototype of regime 12
(attained in 2022−02−18 12)
Latitude
48.30
48.20
−124.05
−123.90
Longitude


Prototype of regime 13
(attained in 2021−02−23 09)
Prototype of regime 14
(attained in 2020−10−21 21)
Prototype of regime 15
(attained in 2022−03−04 21)
Prototype of regime 16
(attained in 2021−09−28 21)
Prototype of regime 17
(attained in 2022−01−31 00)
Prototype of regime 18
(attained in 2020−06−02 15)
Latitude
48.30
48.20
−124.05
−123.90
Longitude


Prototype of regime 19
(attained in 2022−06−07 15)
Latitude
48.30
48.20
−124.05
−123.90
Longitude