ETAS model example - Colfiorito

Francesco Serafini

07/03/2022

Contents

Dataset

In this example, I use the Colfiorito dataset ranging from 1997 to 1998 (included). The first step would be to load the data and plot it.

```
# load the data
load('colfiorito.97_99.Rds')
# to plot correctly
# load italy map
it.map <- ne_countries(country = 'Italy', returnclass = "sf", scale = 'medium')
# extract crs
italy.crs <- crs(it.map)
# create sp object
colfiorito.sp.97_99 <- colfiorito.ds.97_99
coordinates(colfiorito.sp.97_99) <- c('Lon', 'Lat')
# create sf object
colfiorito.sf.97_99 <- st_as_sf(colfiorito.sp.97_99)
# project sf object
st_crs(colfiorito.sf.97_99) <- italy.crs#"+proj=longlat +datum=WGS84 +no_defs"#"+proj=utm +zone=33"
colfiorito.sf.97_99 <- st_transform(colfiorito.sf.97_99, italy.crs)

# create Polygon representing study region
```

```
x.lims <- c(12, 13.5)
y.lims <- c(42, 44)
x_coords <- c(x.lims[1],x.lims[2],x.lims[2],x.lims[1],x.lims[1])
y_coords <- c(y.lims[1],y.lims[1],y.lims[2],y.lims[2],y.lims[1])
poly1 <- sp::Polygon(cbind(x_coords,y_coords))
bdy <- sp::Polygons(list(poly1), ID = "A")
bdy <- sp::SpatialPolygons(list(bdy))
# create sf object
bdy.sf <- st_as_sf(bdy)
st_crs(bdy.sf) <- italy.crs
bdy.sf <- st_transform(bdy.sf, italy.crs)

# plots
# map with Italy
pl.it <- ggplot() +
  geom_sf(data = it.map, fill=alpha("lightgrey", 0), color = 'green') +
  geom_sf(data = bdy.sf) +
  geom_sf(data = colfiorito.sf.97_99, size = 0.2)
# zoom on the sequence
pl.seq <- ggplot() +
  geom_sf(data = bdy.sf) +
  geom_sf(data = colfiorito.sf.97_99[order(colfiorito.sf.97_99$Mw),], shape = 21, mapping = aes(fill = M
  geom_sf(data = colfiorito.sf.97_99[colfiorito.sf.97_99$Mw >= 5,], shape = 24, fill = 'red', size = 2)
  scale_fill_viridis()
# time vs magnitude plot
pl.mag <- ggplot(colfiorito.ds.97_99, aes(x = time_date, y = Mw)) +
  geom_point()
# histogram of times
pl.hist <- ggplot(colfiorito.ds.97_99, aes(x = time_date,)) +
  geom_histogram(bins = 50)
# actual plot
multiplot(pl.it, pl.seq, pl.mag, pl.hist,
          layout = matrix(1:4, ncol = 2, byrow = TRUE))
```

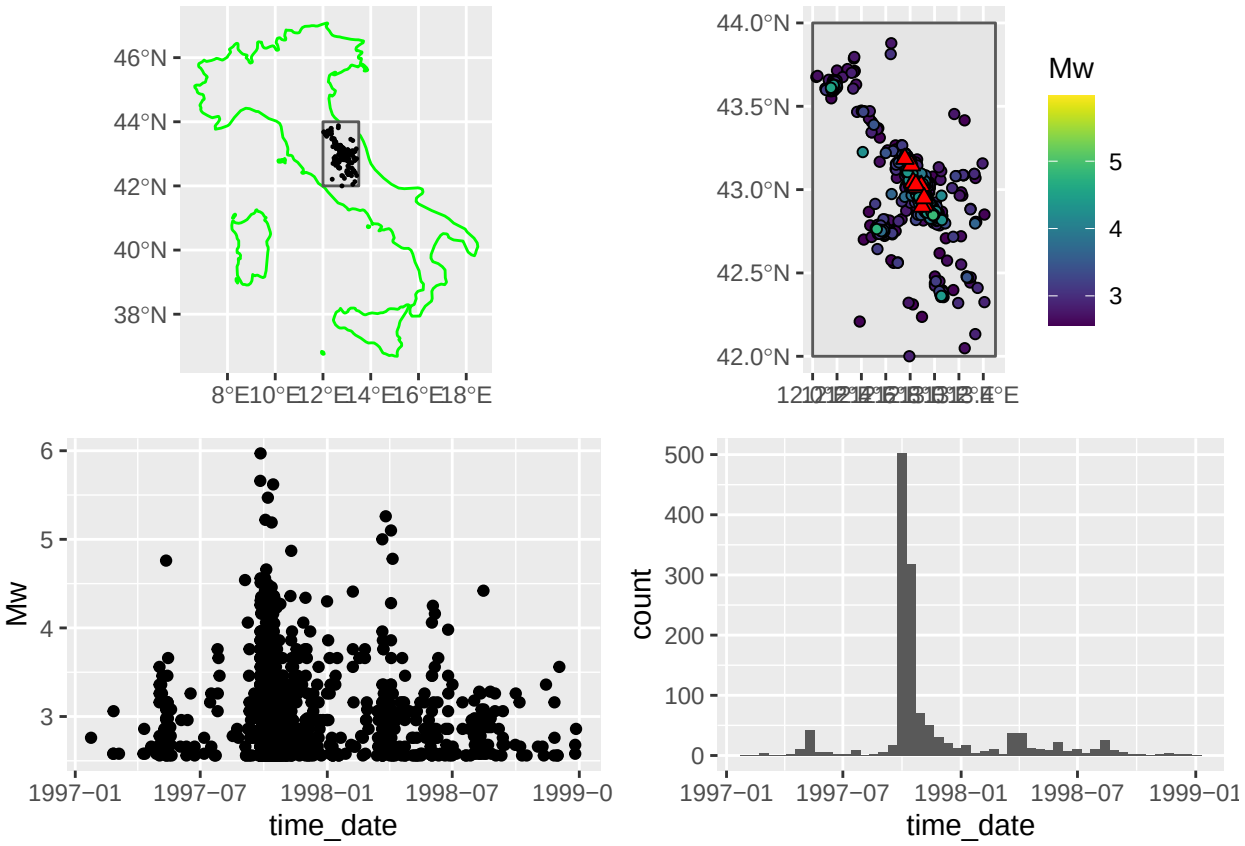


Figure 1: Colfiorito sequence 1997-1998

Next step is to create a data.frame storing the recorded events in the correct format

```
# preparing df for inlabru to be used later on
# starting date of the sequence (I have chosen this one because the first event is isolated in time)
start.date <- min(colfiorito.ds.97_99$time_date[-1])

# create data.frame (variable names are mandatory)
df.bru <- data.frame(x = colfiorito.ds.97_99$Lon, # x for longitude
                     y = colfiorito.ds.97_99$Lat, # y for latitude
                     ts = as.numeric(difftime(colfiorito.ds.97_99$time_date,
                                              start.date,
                                              units = 'days')), # ts for time (in secs, hours, days)
                     mags = colfiorito.ds.97_99$Mw) # mags for magnitudes
# remove the first two rows (first has negative time and the second has time equal 0)
df.bru <- df.bru[-c(1,2),]
```

Model for Background field

We are considering the background rate as given by:

$$\mu(\mathbf{s}) = \mu u(\mathbf{s})$$

Where $\mu \geq 0$ can be seen as the usual background rate, while we refer to $u(\mathbf{s}) \geq 0$ as the background field,

it represents the variation of the background field depending on location and it is normalized such that $\int_W u(\mathbf{s})d\mathbf{s} = 1$. Our goal now, is to estimate the logairthm of the background field, namely $\log u(\mathbf{s})$

The first thing to do is to fit a model for the log background field, which in this case is a simple model with just an intercept and a spatially varying SPDE random effect. The background field may include covariates (as in Kirsty's model). In general, however the model is, a mesh and and the value of the log-intensity at the mesh nodes.

```
# contruct the mesh
mesh_col <- inla.mesh.2d(boundary = bdy, max.edge = 0.05)

# transform data in SpatialPointsDataFrame
df.bru.pp <- df.bru
coordinates(df.bru.pp) <- c('x', 'y')

# set the spde object and the components
spde.o <- inla.spde2.pcmatern(mesh_col,
                              prior.sigma = c(0.1, 0.01),
                              prior.range = c(0.01, 0.01))
cmp <- coordinates ~ field(coordinates, model = spde.o) + Intercept(1)

# fit the model - takes around 50 secs
bru.bkg <- lgcp(components = cmp,
                data = df.bru.pp,
                samplers = bdy,
                domain = list(coordinates = mesh_col),
                options = list(inla.mode = 'experimental',
                               control.inla = list(int.strategy = "eb")))
```

```
# value of the bkg field at pixel coordinates (for plotting only)
pred.bkg <- predict(bru.bkg, pixels(mesh_col), ~ field + Intercept)
# plot log-intensity
ggplot() + gg(pred.bkg['median']) + gg(df.bru.pp) + scale_fill_viridis()
```

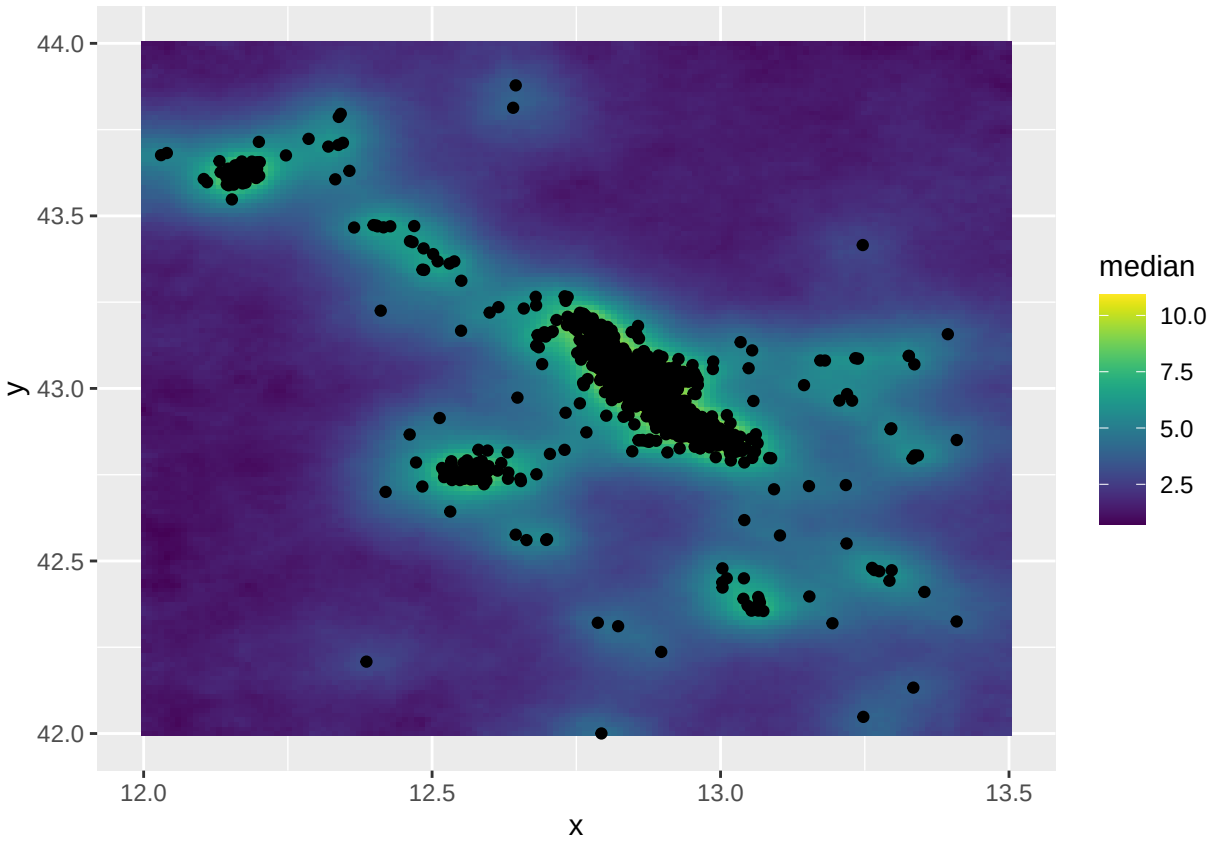


Figure 2: Estimated background field

In the next steps, we are going to use the median of the posterior log-intensity field as background field estimate. We now have to normalize the estimates to make them integrate to one.

```
# background rate at the mesh points
pred.bkg.mesh <- predict(bru.bkg, mesh_col, ~ field + Intercept)

# approximate integral
integ.field <- sum(ipoints(mesh_col)$weight*exp(pred.bkg.mesh$median))

# normalized background rate.
bkg.field <- pred.bkg.mesh$median - log(integ.field)

# create projection to find value of bkg field at the obs points
proj <- inla.mesh.project(mesh_col, cbind(df.bru$x, df.bru$y))

# add a bkg column to the data
df.bru$bkg <- as.numeric(proj$A %*% bkg.field)
```

We will consider the background field as given to estimate the ETAS parameters, however, in a forecasting framework, for each simulated catalogue we can extract a different background field from the posterior. Below three examples.

```
sample.bkg <- generate(bru.bkg, pixels(mesh_col), ~ field + Intercept, 3)
pix <- pixels(mesh_col)
pix$samp1 <- sample.bkg[,1]
```

```
pix$samp2 <- sample.bkg[,2]
pix$samp3 <- sample.bkg[,3]

lims <- range(sample.bkg)

pl1 <- ggplot() + gg(pix['samp1']) + scale_fill_viridis(limits = lims) + theme(legend.position = 'none')
pl2 <- ggplot() + gg(pix['samp2']) + scale_fill_viridis(limits = lims) + theme(legend.position = 'none')
pl3 <- ggplot() + gg(pix['samp3']) + scale_fill_viridis(limits = lims) + theme(legend.position = 'none')
multiplot(pl1, pl2, pl3, cols = 3)
```

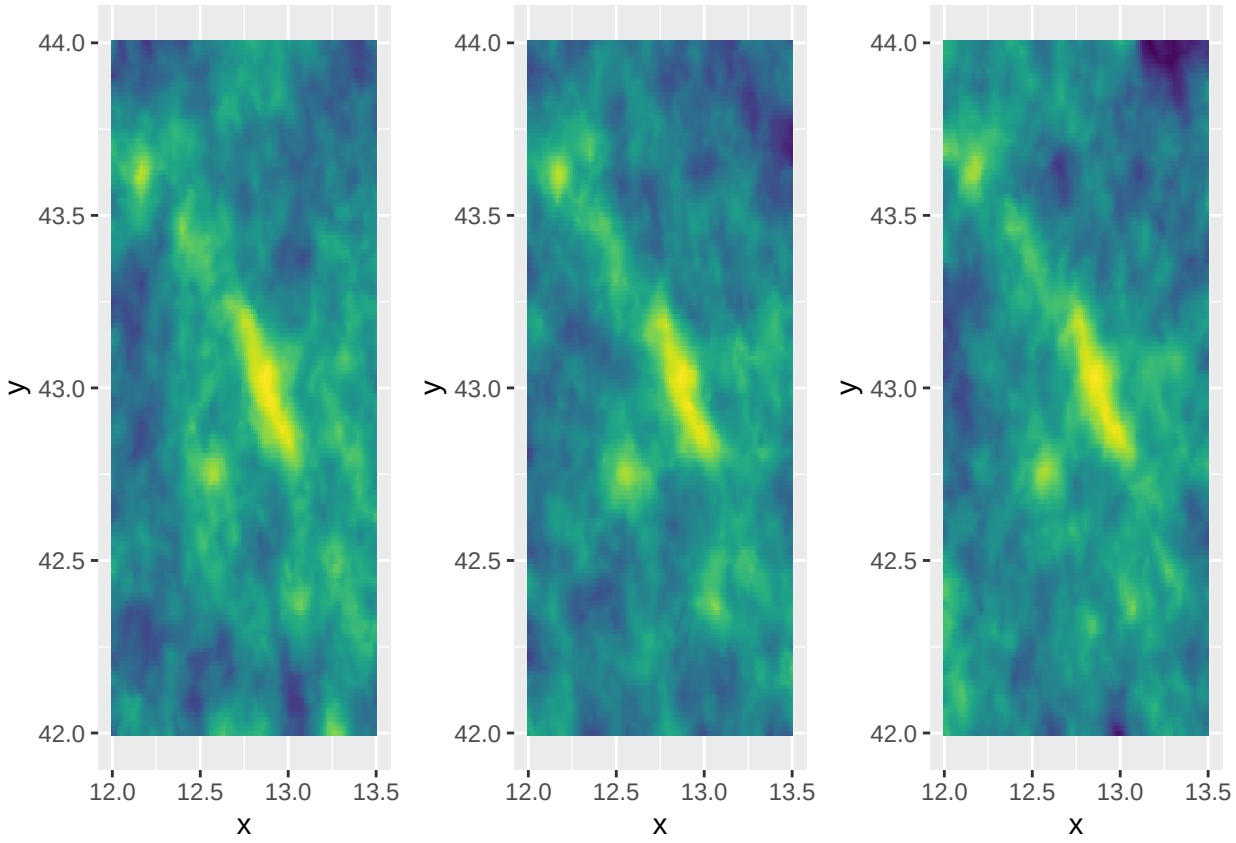


Figure 3: Sample from the posterior of the background field

Fit the model knowing Σ

In this first example, we are going to consider the covariance matrix Σ defining the spatial triggering function as fixed. To have a realistic Σ matrix we are going to consider

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{pmatrix}$$

where $\sigma_1 = \sigma_2 = \text{Med}(d_{min})$ it is equal to the median of the minimum distance between points in the observed sequence, $\rho = \text{corr}(\mathbf{s}_x, \mathbf{s}_y)$ the correlation coefficient of the observed points' coordinates. In this case,

```
# vector of minimum distances
min.dists <- sapply(1:nrow(df.bru), \(idx)
                    min(sqrt(colSums((c(df.bru$x[idx], df.bru$y[idx]) -
```

```
                          rbind(df.bru$x[-idx], df.bru$y[-idx]))^2)))
                    )

# set sigma.1 sigma.2 and rho
sigma.1 <- median(min.dists)
sigma.2 <- sigma.1
ro_ <- cor(df.bru$x, df.bru$y)

# set Sigma
Sigma.p <- matrix(c(sigma.1^2, sigma.1*sigma.2*ro_, sigma.1*sigma.2*ro_, sigma.2^2),
                  byrow = TRUE, ncol = 2)
# set T1 and T2
T1 = 0
T2 = max(df.bru$ts) + 0.1
```

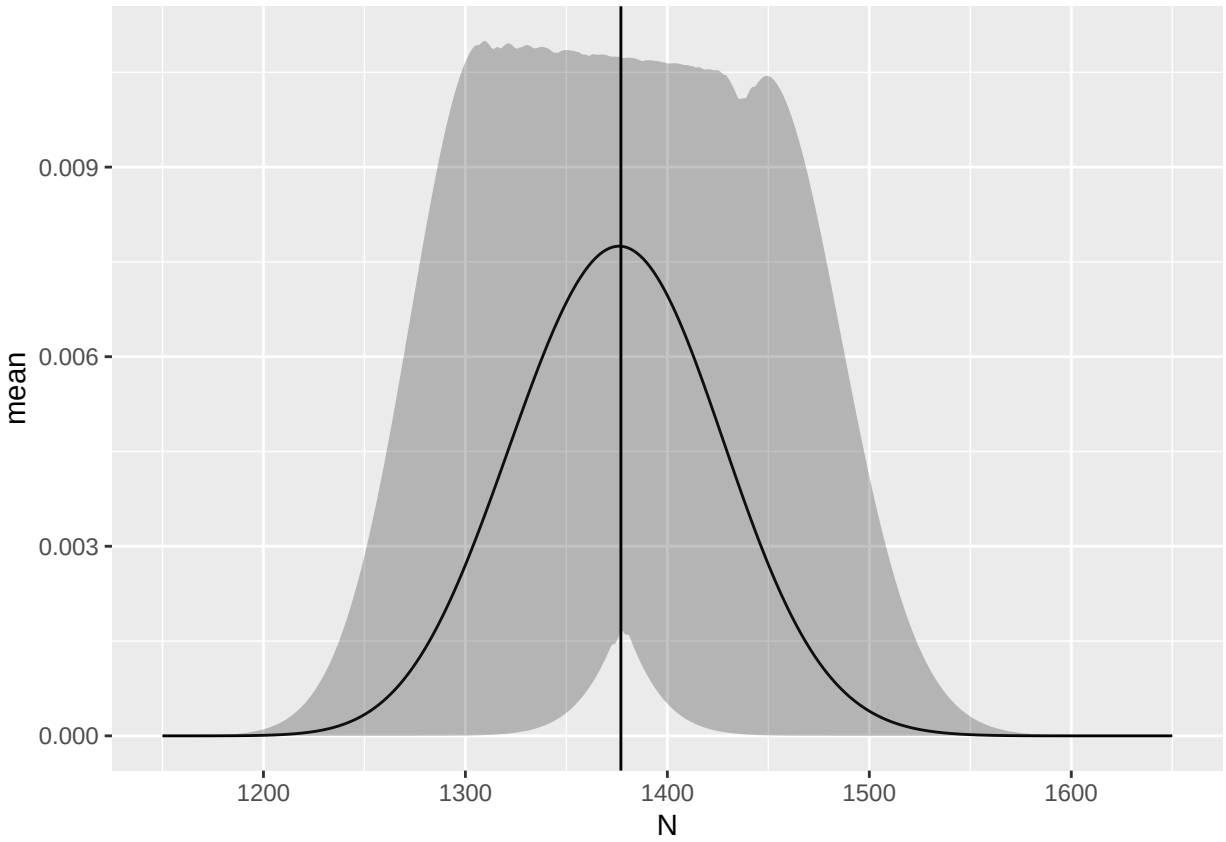
```
# fit the model
# details on data
# df.bru has to be a data.frame with columns
#     x - longitude
#     y - latitude
#     ts - time (between T1, T2, excluded)
#     mags - magnitudes (> M0)
#     bkg - value of the log-background field
# if one of those is missing it will return an error.
# takes around 13 mins
col.fit_sigma <- ETAS.fit.B_bkg(sample.s = df.bru, # data.frame representing data points
                                N.breaks.min = 5,  # minimum number of bins (Integral decomposition)
                                max.length = (T2 - T1)/30,  # maximum length of a time bin
                                Sigma = Sigma.p, # Sigma matrix for spatial trig function
                                prior.mean = c(0,-1,0,0,-1),  # prior mean for thetas of temporal ETAS
                                prior.prec = c(2,2,2,2,2), # prior precision for thetas of temporal ETAS
                                M0 = 2.5, # magnitude of completeness
                                T1 = T1, T2 = T2, # extremes of time interval
                                bdy = bdy, # SpatialPolygon representing study region (has to be squared)
                                bru.opt = # bru options
                                  list(inla.mode = 'experimental', # inla mode
                                       bru_max_iter = 40, # max number of iterations
                                       bru_verbose = 3), # verbose changes what inlabru prints
                                bin.strat = 'alt') # strategy for the bins (recommend to use alt when T2>
save(col.fit_sigma, file = 'col.fit_sigma.Rds')
```

Exploring the solution

Below the distribution of the number of expected points. I have written a function Lambda_ which takes as input also a `part` parameter indicating if the function returns the total number of events, the background number of events or the triggered number of events. This will be useful later on analyzing the differences between models fitted on different data.

```
load('col.fit_sigma.Rds')

Lambda_ <- function(th.mu, th.K, th.alpha, th.c, th.p, Sigma, th, xh, yh, mh, M0, T1, T2, bdy,
                    part = 'total'){
  th_ <- c(0, th.K[1], th.alpha[1], th.c[1], th.p[1])
```

```
  if(part == 'total'){
    exp(th.mu[1])*(T2 - T1) + sum(exp(logLambda.h.vec(th.p = th_, th = th, xh = xh, yh = yh, mh = mh,
                                      M0 = M0, T1 = T1, T2 = T2, bdy = bdy, Sigma = Sigma)))
  }
  else if(part == 'background'){
    exp(th.mu[1])*(T2 - T1)
  }
  else if(part == 'triggered'){
    sum(exp(logLambda.h.vec(th.p = th_, th = th, xh = xh, yh = yh, mh = mh,
                                      M0 = M0, T1 = T1, T2 = T2, bdy = bdy, Sigma = Sigma)))
  }
  else{
    stop('Unknown part')
  }
}

N.distr <- predict(col.fit_sigma, df.bru,
                   ~ data.frame(N = 1150:1650,
                                pdf = dpois(1150:1650,
                                            lambda = Lambda_(th.mu = th.mu,
                                                             th.K = th.K,
                                                             th.alpha = th.alpha,
                                                             th.c = th.c,
                                                             th.p = th.pm1,
                                                             Sigma = Sigma.p,
                                                             M0 = 2.5, th = ts, xh = x, yh = y,
                                                             mh = mags, T1 = 0, T2 = T2, bdy = bdy))
                                )
                   )

plot(N.distr) +
  geom_vline(xintercept = nrow(df.bru))
```



Below the code to extract a sample from the posterior of the parameters and use it to generate a catalogue.

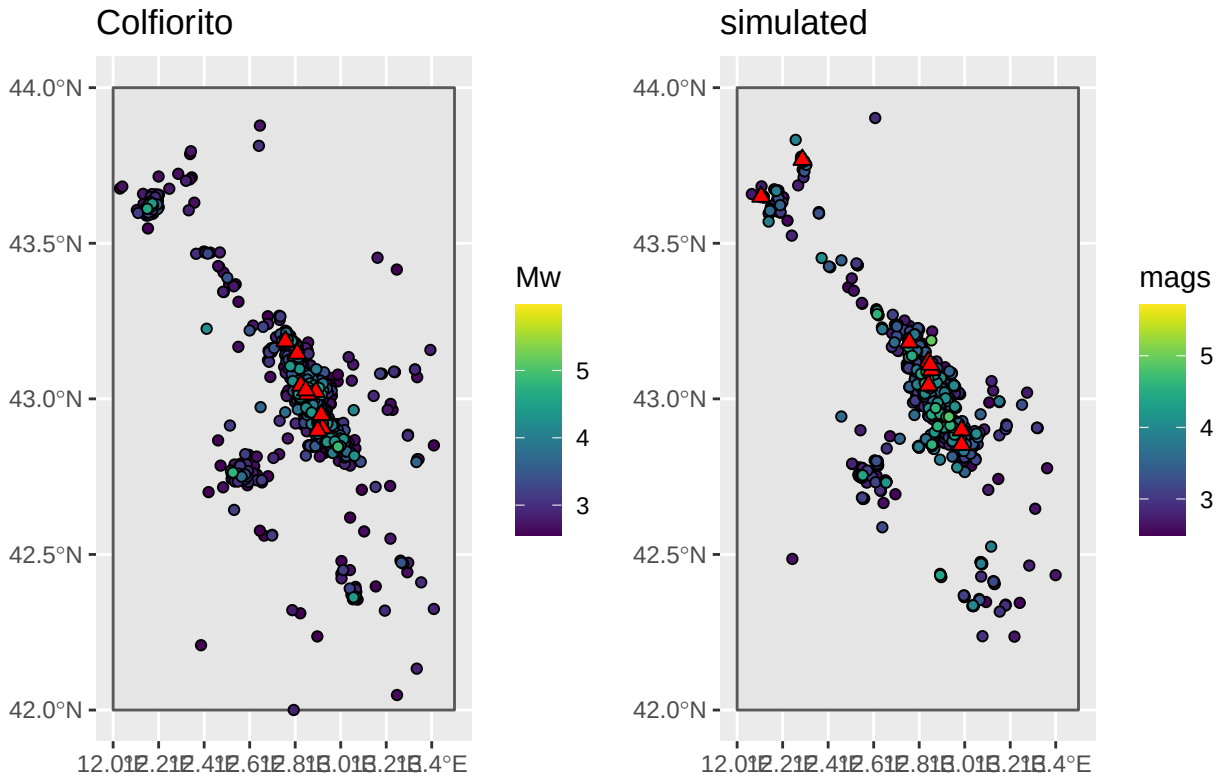
```
# generate posterior samples of the parameters
params <- generate(col.fit_sigma, df.bru, ~ c(th.mu, th.K, th.alpha, th.c, th.pm1),
                   n.samples =  1) # number of samples from the posterior

# estimate beta for GR law
M0 = 2.5 # mag of completeness
beta.p_ <- 1/mean(df.bru$mags - M0) # classic beta (GR law) estimator

# generate catalogue
# it returns a list of Generations
sim.cat <- sample.ETAS(th.p = as.numeric(params),
                       beta.p = beta.p_,
                       M0 = M0,
                       T1 = T1,
                       T2 = T2,
                       bdy = bdy,
                       Sigma = Sigma.p,
                       loglambda.bkg = bkg.field,
                       mesh.bkg = mesh_col,crs_obj = italy.crs)
# merge different generations and arrange events by time
sim.cat <- bind_rows(sim.cat) %>%
  arrange(ts)
# have a look
head(sim.cat)
```

```
##             x        y        ts     mags gen
```

```
## ...1  12.89204 42.95150 0.1575851 4.450043   1
## ...2  12.94173 42.92475 1.4558881 3.606295   1
## ...3  12.90008 42.97181 1.8134063 2.812335   1
## x...4 12.90601 42.96990 2.4520173 2.569306   2
## ...5  12.90554 42.97032 2.9663204 3.071196   3
## ...6  12.89005 42.94672 5.0583626 2.649117   1
```

For plotting it is convenient to transform the catalogue in an `sf` object. The easiest way (for me) to do that is to transform the `data.frame` in a `SpatialPointsDataFrame` and then, turn it in a `sf` object.

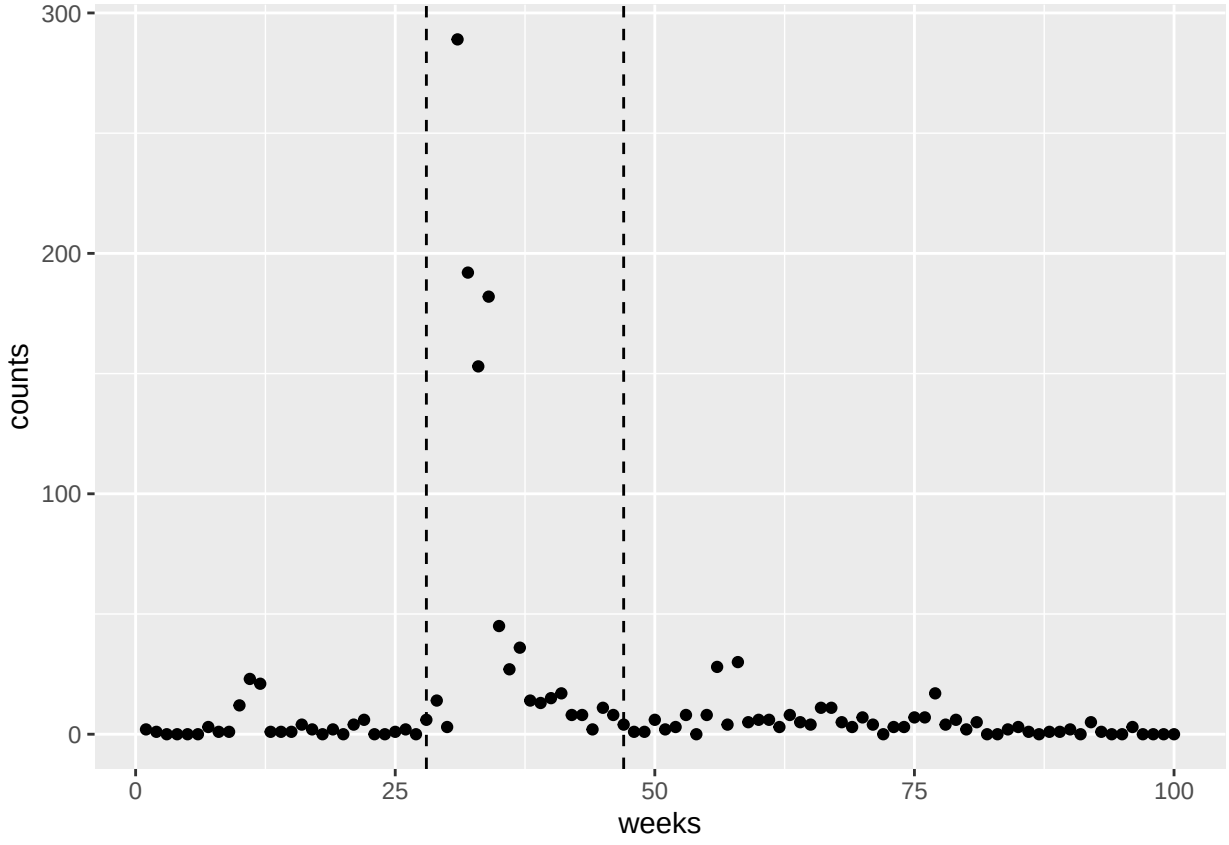
```
# create a copy of the data.frame
sim.cat.sp <- sim.cat
# turn it into SpatialPointsDataFrame
coordinates(sim.cat.sp) <- c('x', 'y')
# turn it into sf object
sim.cat.sf <- st_as_sf(sim.cat.sp)
# set crs
st_crs(sim.cat.sf) <- italy.crs
# project
sim.cat.sf <- st_transform(sim.cat.sf, italy.crs)

# plot the sequence
pl.seq2 <- ggplot() +
  geom_sf(data = bdy.sf) +
  geom_sf(data = sim.cat.sf[order(sim.cat.sf$mags),], shape = 21, mapping = aes(fill = mags)) +
  geom_sf(data = sim.cat.sf[sim.cat.sf$mags >= 5,], shape = 24, fill = 'red', size = 2) +
  scale_fill_viridis() +
  labs(title = 'simulated')
# compare with Colfiorito sequence
multiplot(pl.seq + labs(title = 'Colfiorito'), pl.seq2, cols = 2)
```



Weekly retrospective forecast

This example is more similar to an actual forecasting experiment, the only difference is that is retrospective, meaning that the parameters are optimized on the same data that will be used for testing. Here, we present a weekly experiment, in which we take 20 weeks from 03/09/1997 to 17/01/1998. The period is highlighted below by vertical dashed line.

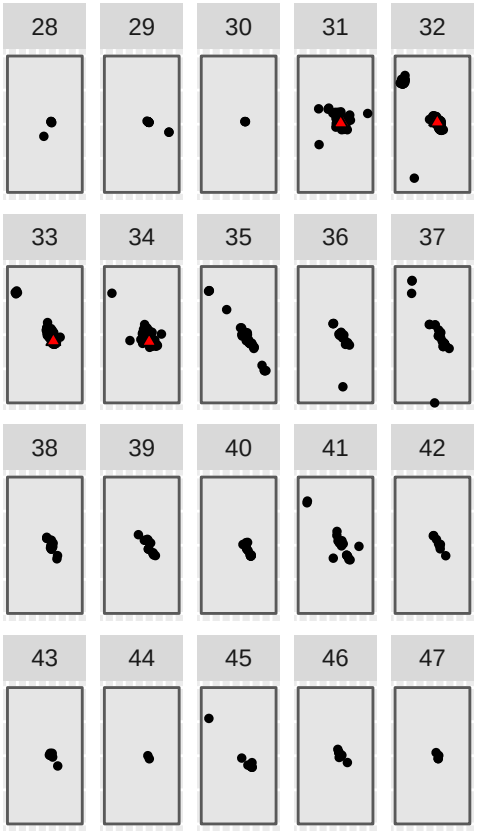
```
# divide the first 100 weeks of data
list.weeks <- foreach(i = 1:100) %do% {
  df.bru[df.bru$ts >= (i-1)*7 & df.bru$ts <= (i)*7,]
}
# number of events per week
N.weeks <- unlist(lapply(list.weeks, nrow))
# plot
ggplot() +
  geom_point(aes(x = 1:length(N.weeks), y = N.weeks)) +
  xlab('weeks') +
  ylab('counts') +
  geom_vline(xintercept = c(28,47), linetype = 2)
```



Below a plot of the observations per each week, red triangle indicates events with magnitude greater than 5.

```
# select weeks to show
weeks.to.fore <- list.weeks[28:47]
# create a data.frame with all the observations and a column that indicate the week
df.plot <- foreach(i = 1:length(weeks.to.fore), .combine = rbind) %do% {
  data.frame(x = weeks.to.fore[[i]]$x,
             y = weeks.to.fore[[i]]$y,
             mags = weeks.to.fore[[i]]$mags,
```

```
            week = 28 + (i - 1))
}

# turn it in an sf object
df.plot.sp <- df.plot
coordinates(df.plot.sp) <- c('x', 'y')
df.plot.sf <- st_as_sf(df.plot.sp)
st_crs(df.plot.sf) <- italy.crs
df.plot.sf <- st_transform(df.plot.sf, italy.crs)

# plot it
ggplot() +
  geom_sf(data = bdy.sf) +
  geom_sf(data = df.plot.sf, size = 1) +
  geom_sf(data = df.plot.sf[df.plot.sf$mags >= 5, ], shape = 24, fill = 'red', size = 1.5) +
  facet_wrap(facets = vars(week)) +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank())
```



Below the function to generate the forecast for a given week. The function takes in input the fitted model to generate the parameters, `total.data` is the entire history of events as `data.frame`, `weeks.data` is a list containing the observations for each week, `week.idx` is an index indicating the week we want to forecast. The others parameters are parameters needed for the simulation, the covariance matrix of the spatial triggering function, the background field, the mesh, the parameter of the GR law, the number of samples and the crs object. For the present case, they are set to default to the values retrieved before, in this way we don't need to specify them explicitly every time.

```
fore.week <- function(fit_, total.data, weeks.data, week.idx,
                      bdy_ = bdy, Sigma_ = Sigma.p,
```

```
                      bkg.field_ = bkg.field, mesh_ = mesh_col,
                      beta.p = beta.p_,
                      n.samp = 1000, crs_ = italy.crs){

  # set extreme of the interval we want to forecast
  T1.wf <- min(weeks.data[[week.idx]]$ts)
  T2.wf <- T1.wf + 7
  # select all the data recorded before the forecasting period
  past.week.data <- total.data[total.data$ts < T1.wf, ]
  # sample from the posterior of the parameters
  # it generate a matrix with nrow = 5 (number of parameters) and ncol = number of samples
  sample.param <- generate(fit_, data.frame(x = 0, y = 0, ts = 0, mags = 0),
                           ~ c(th.mu, th.K, th.alpha, th.c, th.pm1), n.samples = n.samp)

  # initialize list of samples
  list.sim.cat <- list()
  # for each combination of parameters from the posterior
  for(i in 1:ncol(sample.param)){
    # set parameters
    th_ <- sample.param[,i]
    # sample
    ss <- sample.ETAS(th.p = th_,
                      beta.p = beta.p,
                      M0 = 2.5,
                      T1 = T1.wf, T2 = T2.wf,
                      loglambda.bkg = bkg.field_, mesh.bkg = mesh_, crs_obj = italy.crs,
                      bdy = bdy_, Sigma = Sigma_, Ht = past.week.data)
    # combine generations and arrange by time
    ss <- bind_rows(ss) %>%
      arrange(ts)
    # store
    list.sim.cat[[i]] = ss
    # print completeness
    if( (i/ncol(sample.param)) %% 0.25 == 0){
      print(paste0('completed : ', i/ncol(sample.param)))
    }
  }
  # return the list of samples
  list.sim.cat
}
```

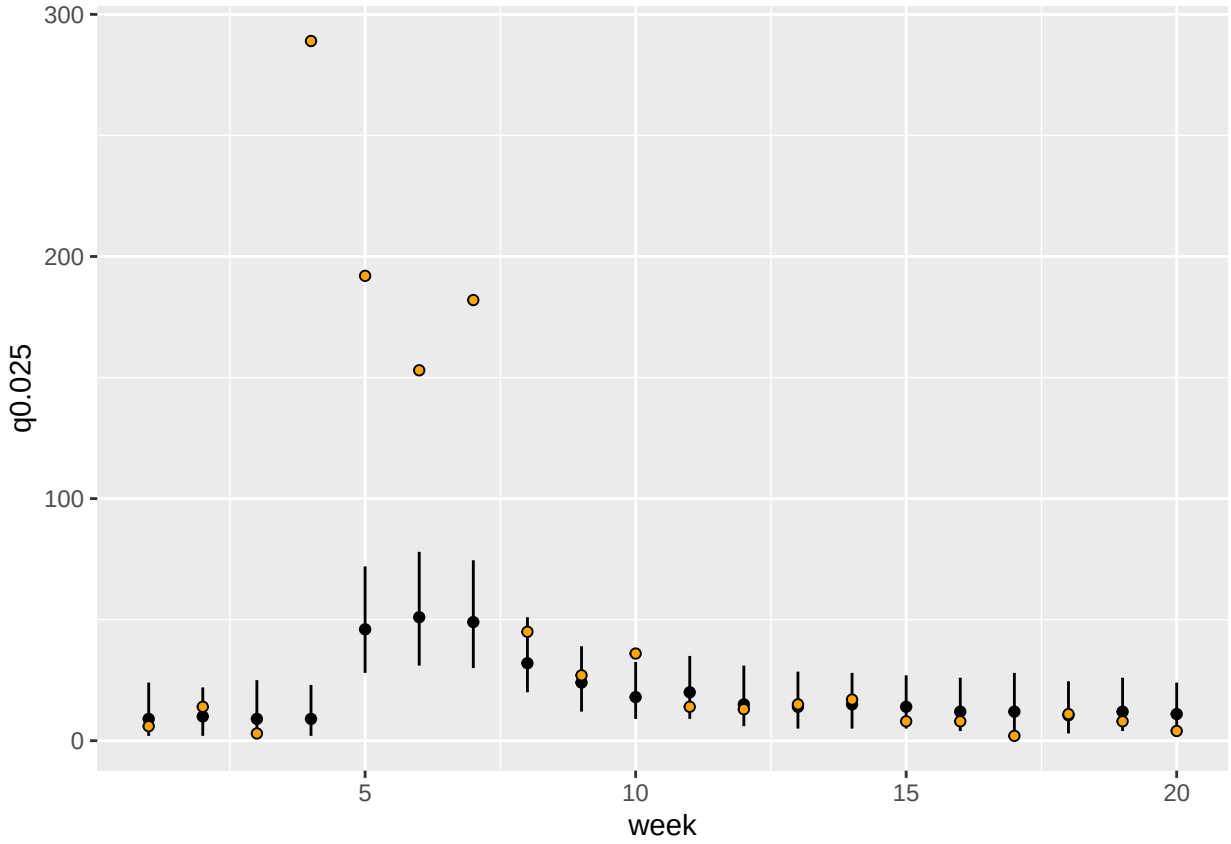
To obtain forecasts for a set of weeks is sufficient to set a We just need to iterate over the weeks we want to predict.

```
for(ww in 1:length(weeks.to.fore)){
  print(ww)
  # forecast a week produce a list of 500 simulated catalogues as data.frames
  ll <- fore.week(fit_ = col.fit_sigma, total.data = df.bru, weeks.data = weeks.to.fore,
                  week.idx = ww, n.samp = 500)
  # store them
  save(ll, file = paste0('fore_weeks/fore_week_', ww, '.Rds'))
}
```

Weekly forecast - Number of events

Below code to extract informations from the lists of simulated catalogues we have generated before. First of all, we are interested in the number of events per week. To extract them, we just load the simulated catalogues for each week and counts the number of rows for each element of the list. Notice that, we have saved all the forecasts with the same name, therefore when we load a new week it overwrites the old one saving memory (we don't need them loaded all together). We will produce a `data.frame` with weeks as rows and columns reporting the $0.025, 0.5, 0.975$ quantiles of the number of events in the simulated catalogues.

```
# extract informations on Number of events
# take the file names in the folder
files.names <- list.files(path = 'fore_weeks/')
# initialize matrix to store information on the number of events per week
N.matrix <- matrix(NA, ncol = 5, nrow = length(files.names))
# set colnames
colnames(N.matrix) <- c('q0.025', 'q0.5', 'q0.975','N.obs','week')

# for each file (they may not be sorted correctly ex. week 10 comes before week 2)
for(idx in 1:length(files.names)){
  # extract forecast file name
  fore.name <- files.names[idx]
  # load the forecast
  load(paste0('fore_weeks/',fore.name))
  # extract which week the forecast refers to
  n.week <- as.numeric(gsub('.Rds', '', gsub('fore_week_', '', fore.name)))
  # extract number of observed event in that week
  N.obs <- nrow(weeks.to.fore[[n.week]])
  # counts number of observations per simulated catalogue
  N.sim <- sapply(ll, nrow)
  # store informations
  N.matrix[idx, ] <- c(quantile(N.sim, 0.025), quantile(N.sim, 0.5), quantile(N.sim, 0.975),
                       N.obs, n.week)
}

# transform matrix in data.frame and arrange by week
N.df <- data.frame(N.matrix) %>%
  arrange(week)

# plot
ggplot(N.df) +
  geom_segment(aes(x = week, xend = week, y = q0.025, yend = q0.975)) +
  geom_point(aes(x = week, y = q0.5)) +
  geom_point(aes(x = week, y = N.obs), shape = 21, fill = 'orange')
```



Weekly forecast - Number of events spatial distribution

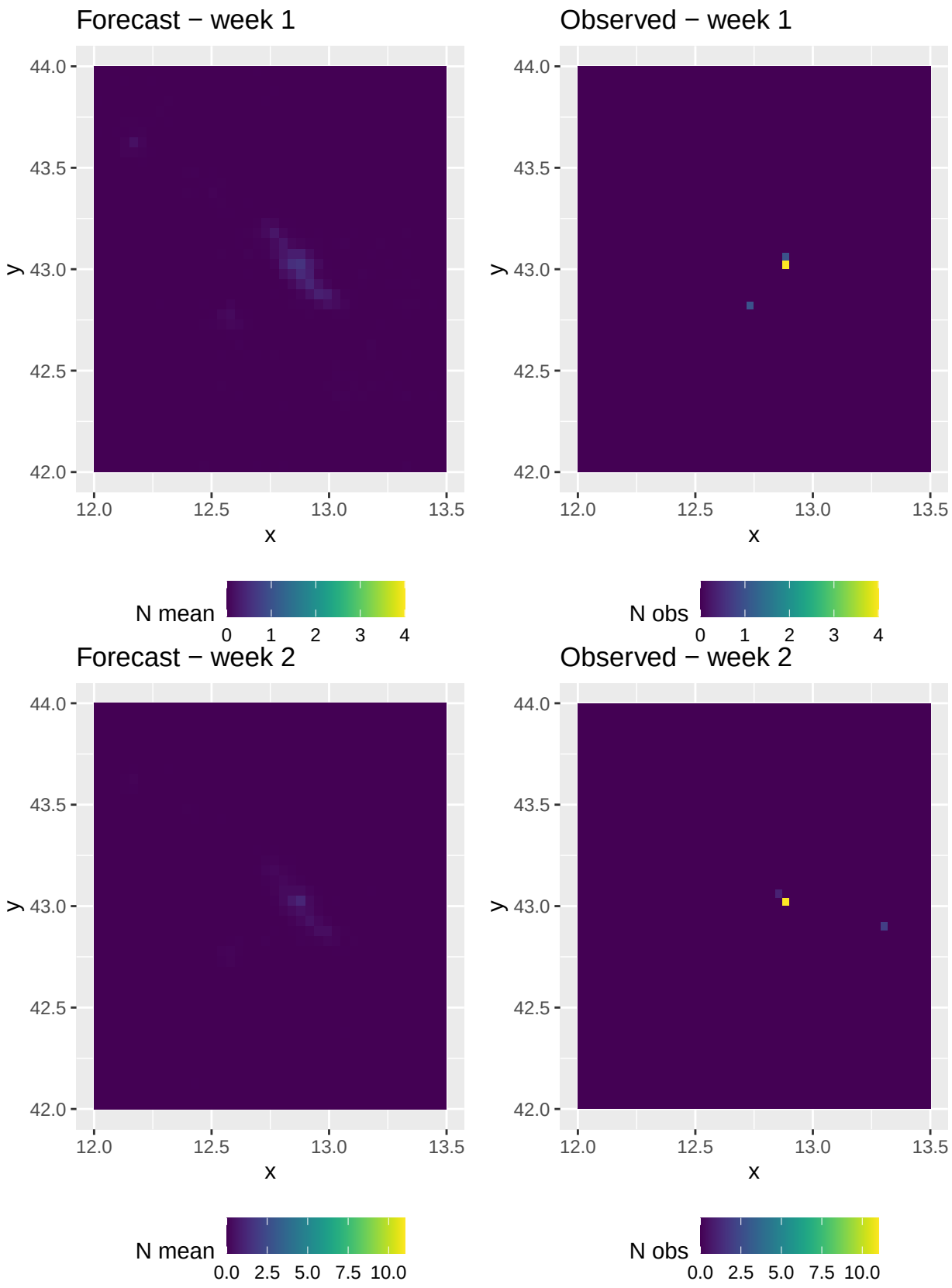
Below we check the spatial distribution of our forecast in terms of average number of events and probability of activity. This is to show a general technique that can be used to calculate other functionals (e.g, number of events quantiles, number of events variance, probability of events above a certain magnitude). First we are going to consider the number of events. For each week we are going to create a `raster` object covering the region of interest (below 40×40 pixels), each cell is initialized at 0. Then, for each simulated catalogue, we update the value of the cell with the number of simulated events in the cell. Last step, we divide the value of each cell for the number of simulated catalogues, to get the average number of simulated events per cell. We will produce a list of `SpatialPointsDataFrame`'s representing the spatial distribution of the average number of events per each week. This choice of output is mainly for plotting purposes, can be changed in something lighter.

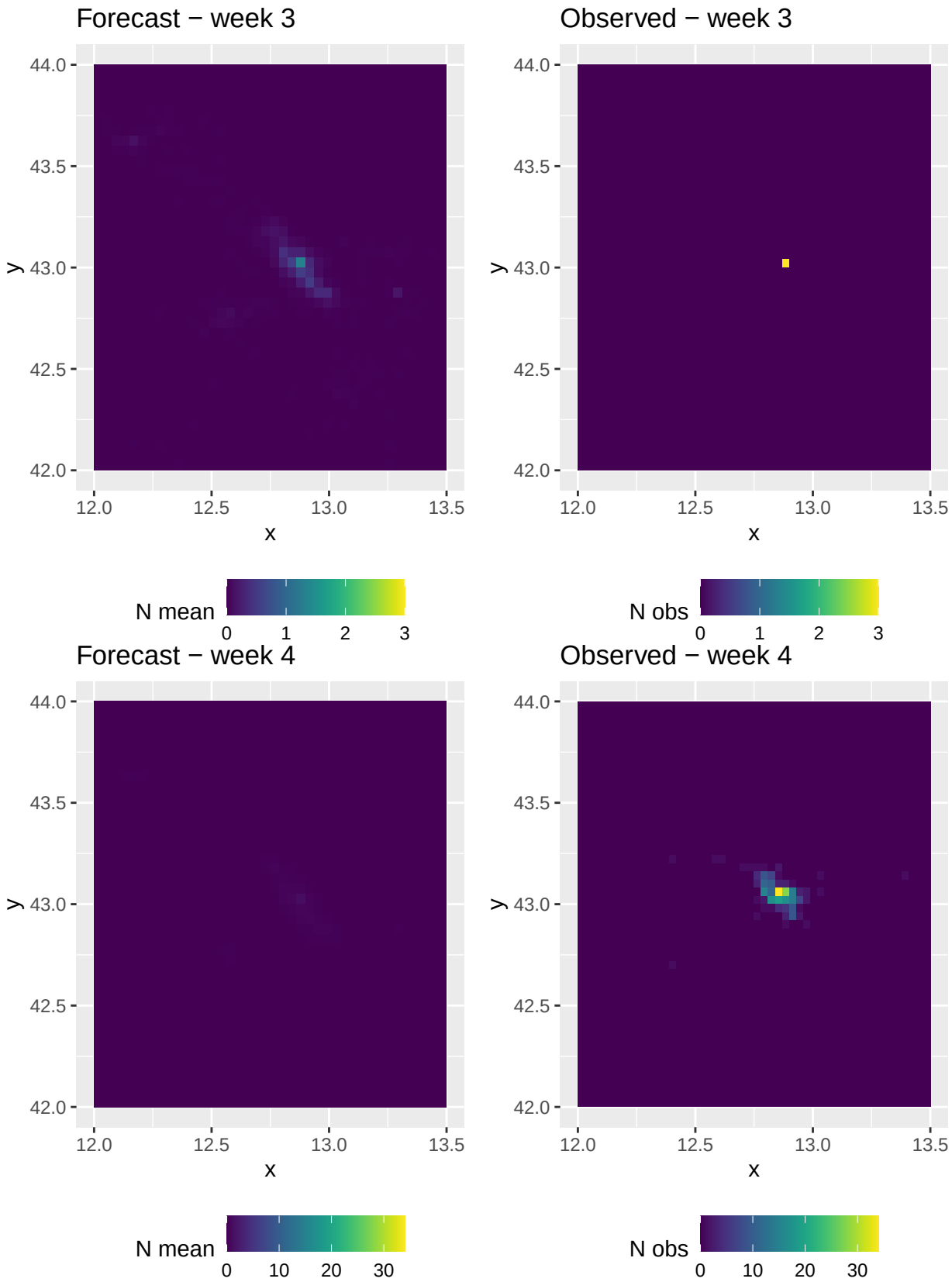
```
# spatial distribution
# take the file names in the folder
files.names <- list.files(path = 'fore_weeks/')
# initialize list of SpatialPixelsDataFrame
list.pixels <- list(rep(NA, length(files.names)))
# for each week
for(idx.file in 1:length(files.names)){
  # initialize raster
  raster.grid <- raster(bdy, nrows = 40, ncols = 40)
  # set value of the cells at 0
  raster.grid[] <- 0
  fore.name <- files.names[idx.file]
  load(paste0('fore_weeks/',fore.name))
  n.week <- as.numeric(gsub('.Rds', '', gsub('fore_week_', '', fore.name)))
```

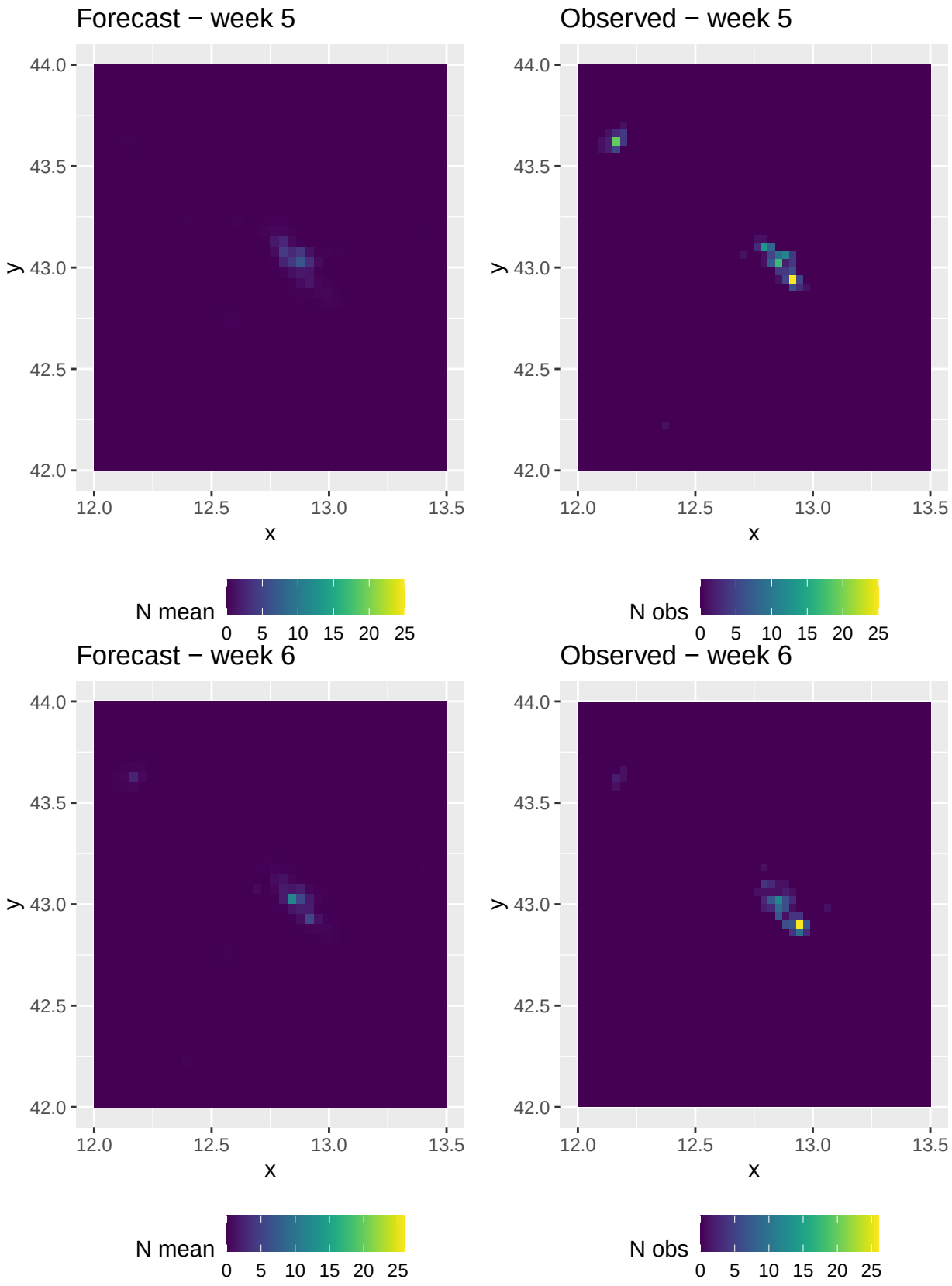
```
  # for each simulated catalogue
  for(idx.car in 1:length(ll)){
    # extract coordinates of the events
    sample.cat <- ll[[idx.car]][,c('x','y')]
    # this gives a table with names = index of cells with events in it and value = number of events
    counts_ <- table(cellFromXY(raster.grid, sample.cat))
    # update only the cells with events in them
    raster.grid[as.numeric(names(counts_))] <- raster.grid[as.numeric(names(counts_))] + counts_
  }
  # divide the cells' value for the number of simulated catalogues
  raster.grid[] <- raster.grid[]/length(ll)
  # store it as a SpatialPixelsDataFrame
  list.pixels[[n.week]] <- as(raster.grid, 'SpatialPixelsDataFrame')
}
# store it
save(list.pixels, file= 'list.pixels.Rds')
```

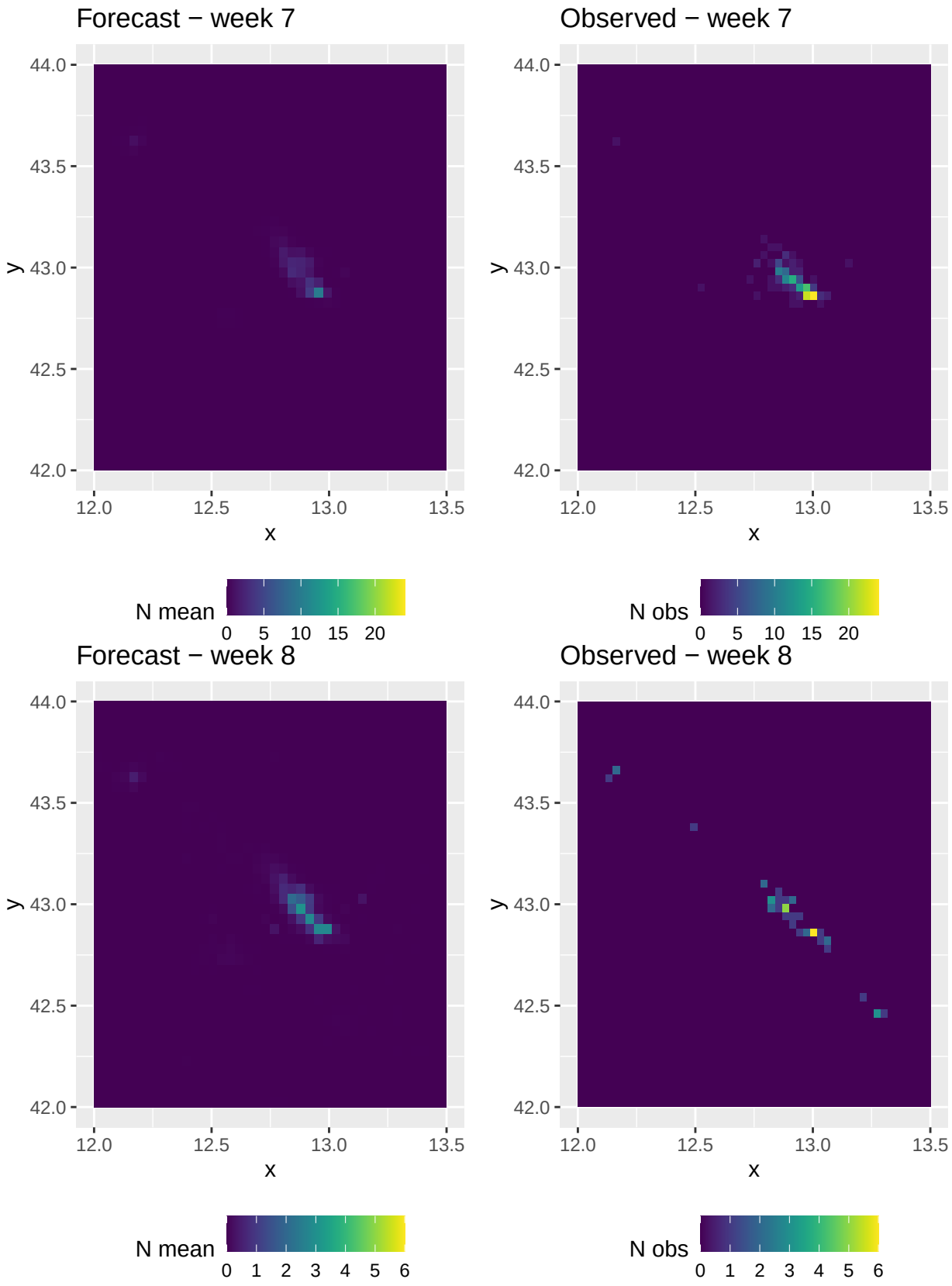
To plot them is sufficient to load the `SpatialPixelsDataFrame`s. Here we are going to compare them with the observed number of events per each week.

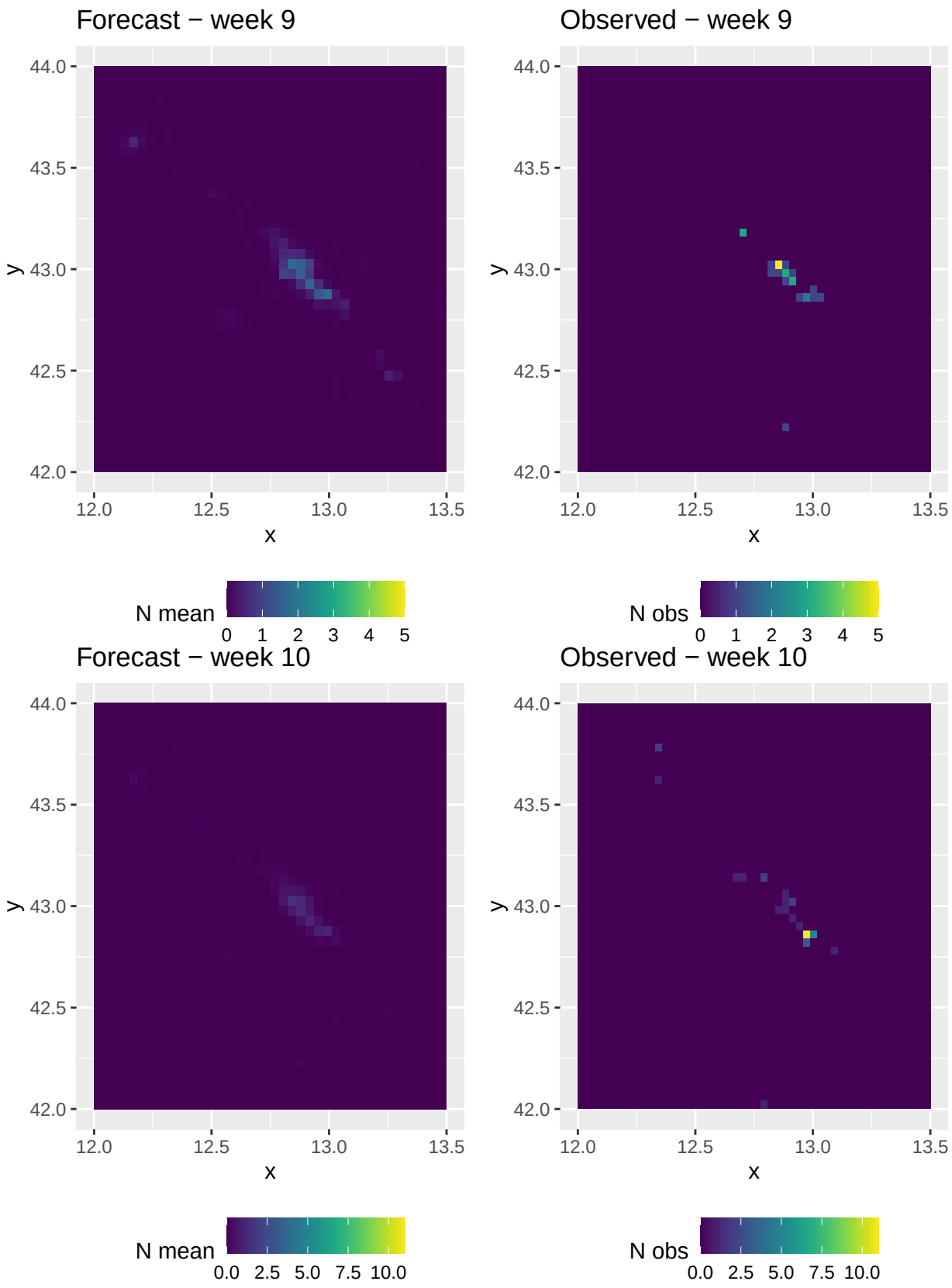
```
load('list.pixels.Rds')
# for each week
for(idxx in 1:length(weeks.to.fore)){
  # counts the number of observed events per cell as before
  week.data <- weeks.to.fore[[idxx]][,c('x', 'y')]
  raster.grid <- raster(bdy, nrows = 50, ncols = 50)
  raster.grid[] <- 0
  counts_ <- table(cellFromXY(raster.grid, week.data))
  raster.grid[as.numeric(names(counts_))] <- counts_
  week.sp <- as(raster.grid, 'SpatialPixelsDataFrame')
  # set scale limits from plotting
  N.lims <- c(min(c(week.sp$layer, list.pixels[[idxx]]$layer)),
              max(c(week.sp$layer, list.pixels[[idxx]]$layer)))

  pl.pred <- ggplot() +
    gg(list.pixels[[idxx]]) +
    labs(title = paste0('Forecast - week ', idxx), fill = 'N mean') +
    theme(legend.position = 'bottom') +
    scale_fill_viridis(limits = N.lims)


  pl.obs <- ggplot() +
    gg(week.sp) +
    labs(title = paste0('Observed - week ', idxx), fill = 'N obs') +
    theme(legend.position = 'bottom')+
    scale_fill_viridis(limits = N.lims)


  multiplot(pl.pred, pl.obs, cols = 2)

}
```

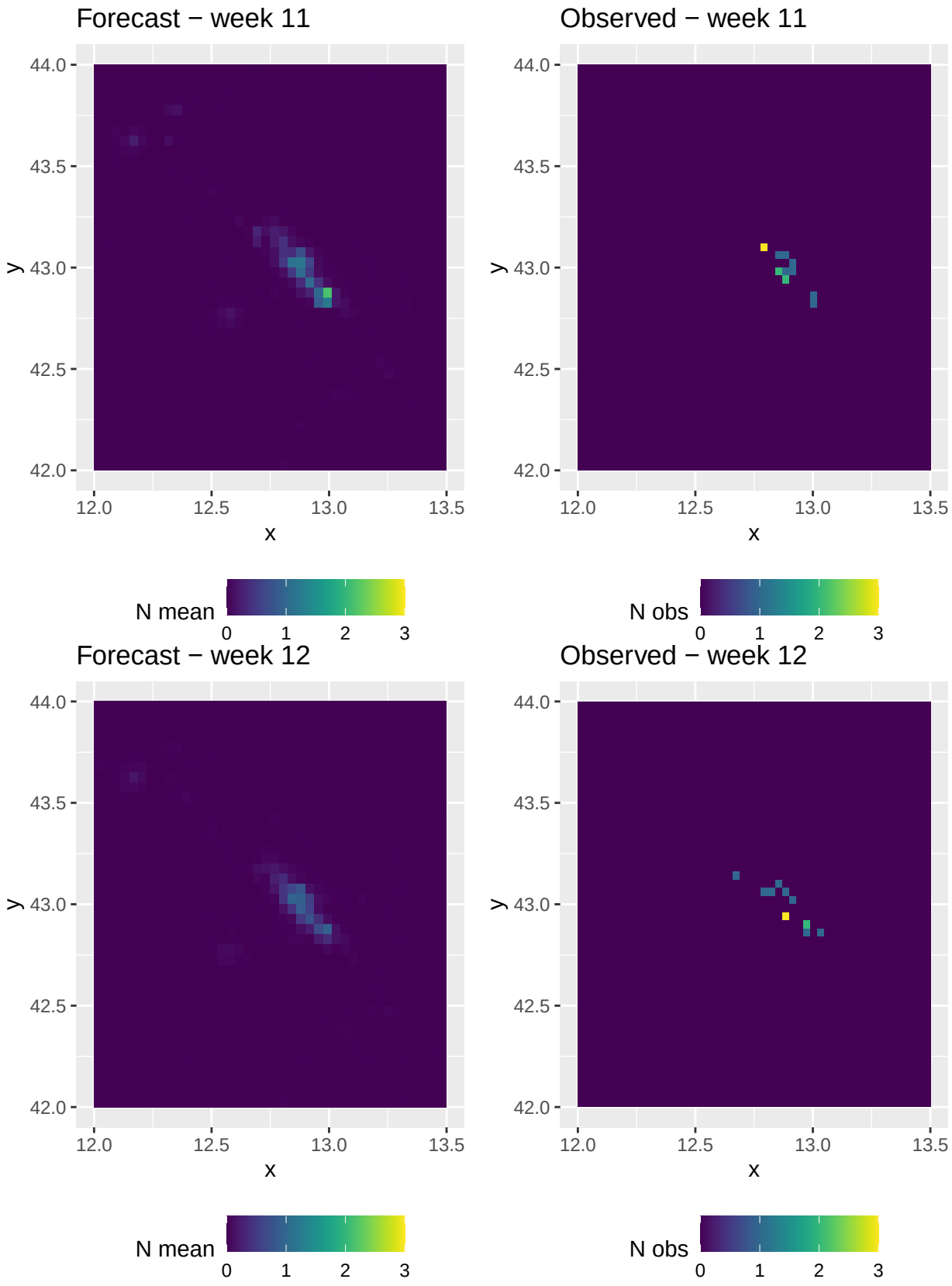

Forecast − week 1
Observed − week 1
Forecast − week 2
Observed − week 2
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
0 1 2 3 4
N obs
0 1 2 3 4
N mean
0.0 2.5 5.0 7.5 10.0
N obs
0.0 2.5 5.0 7.5 10.0

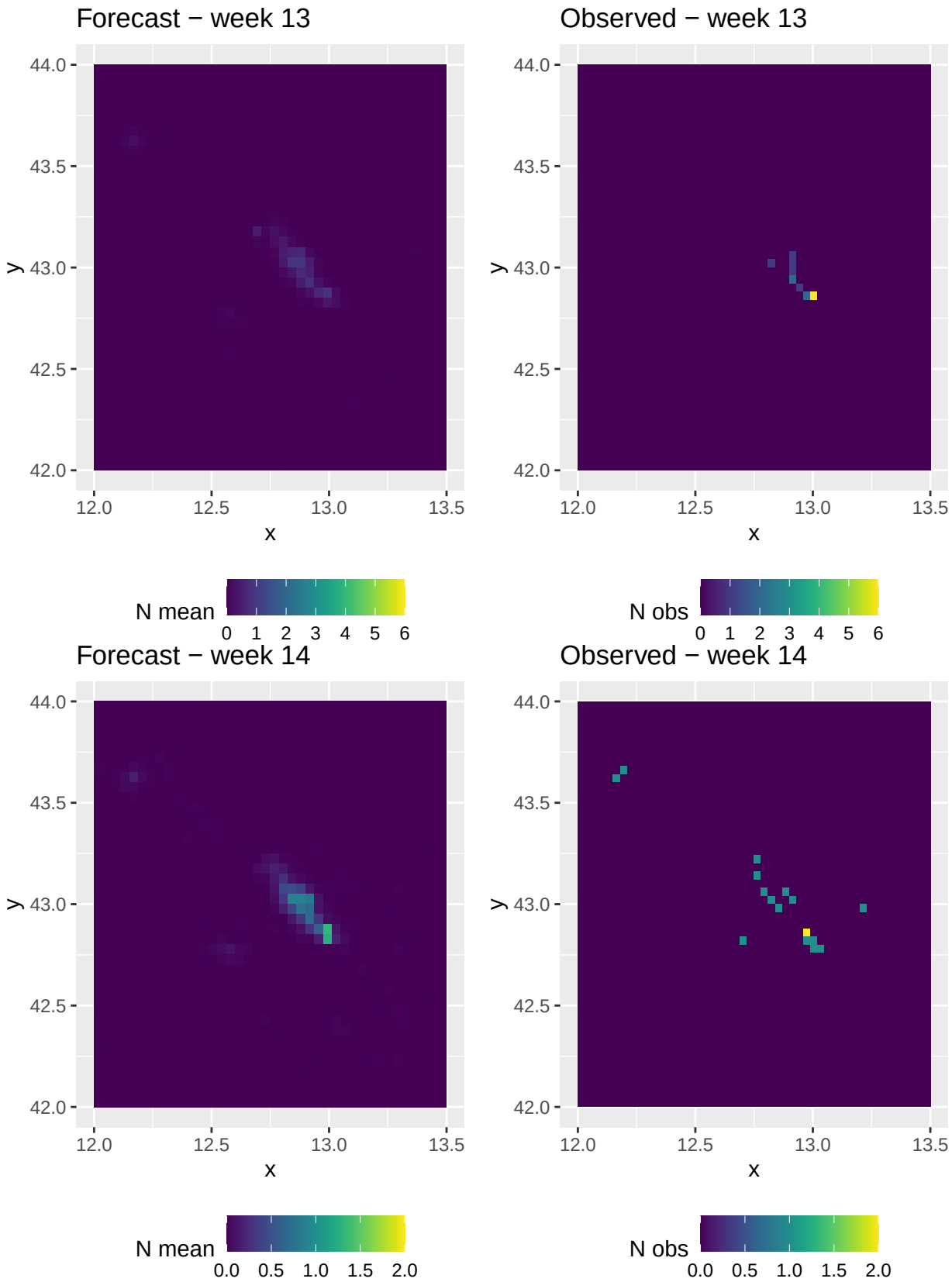

Forecast − week 3
Observed − week 3
Forecast − week 4
Observed − week 4
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0
1
2
3
10
20
30

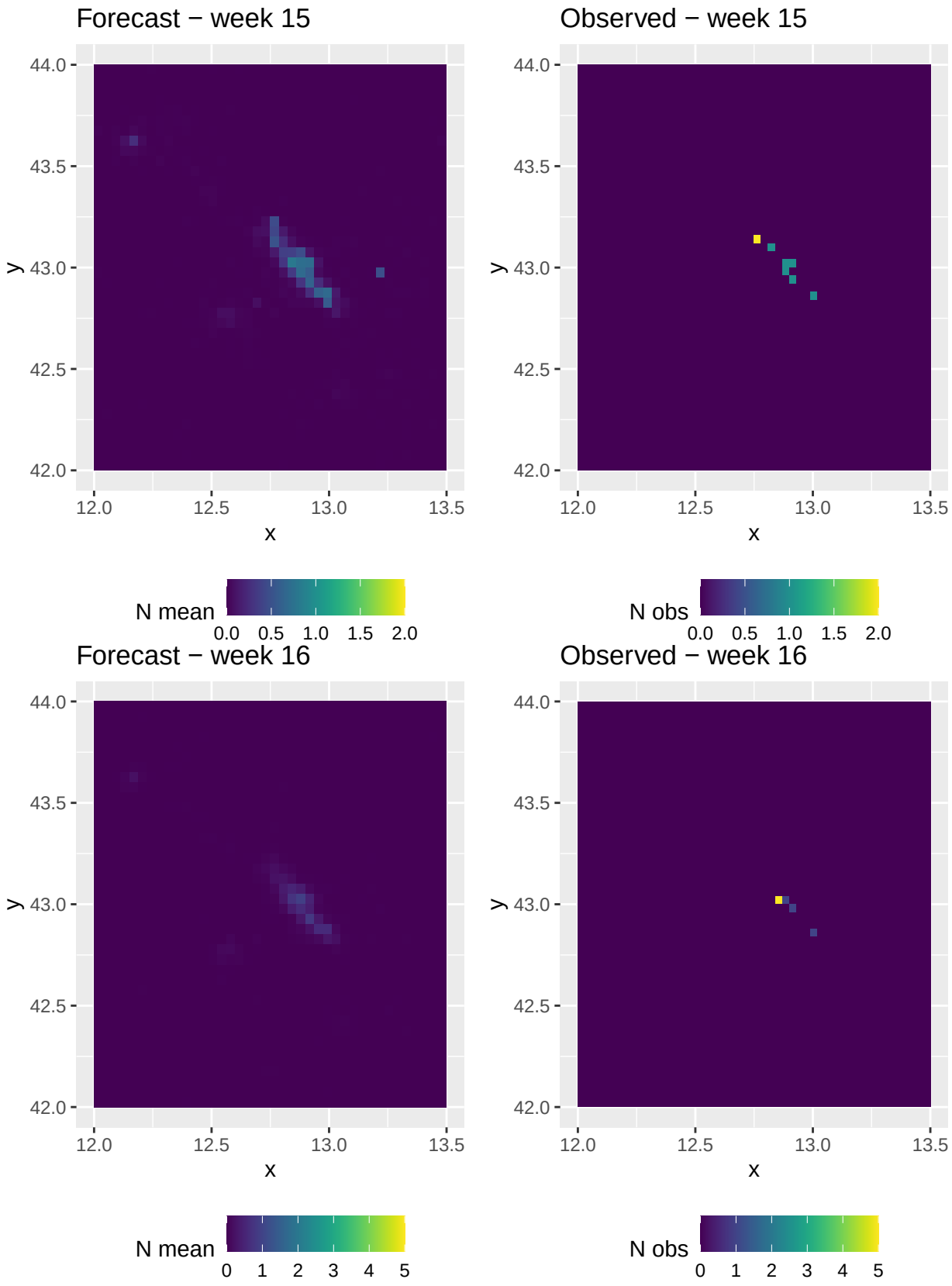

Forecast − week 5
Observed − week 5
Forecast − week 6
Observed − week 6
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0
5
10
15
20
25

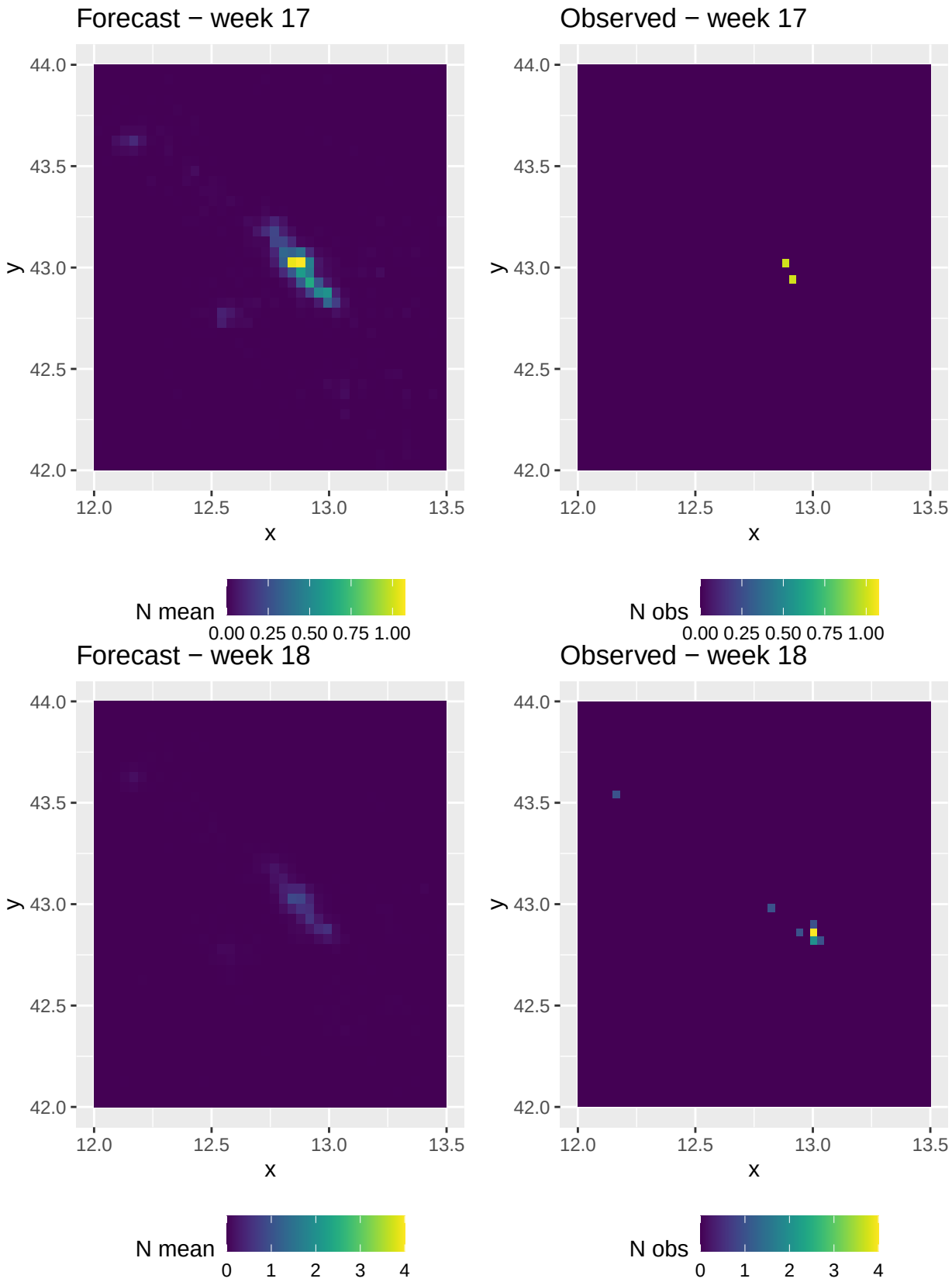

Forecast − week 7
Observed − week 7
Forecast − week 8
Observed − week 8
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0 5 10 15 20
0 1 2 3 4 5 6

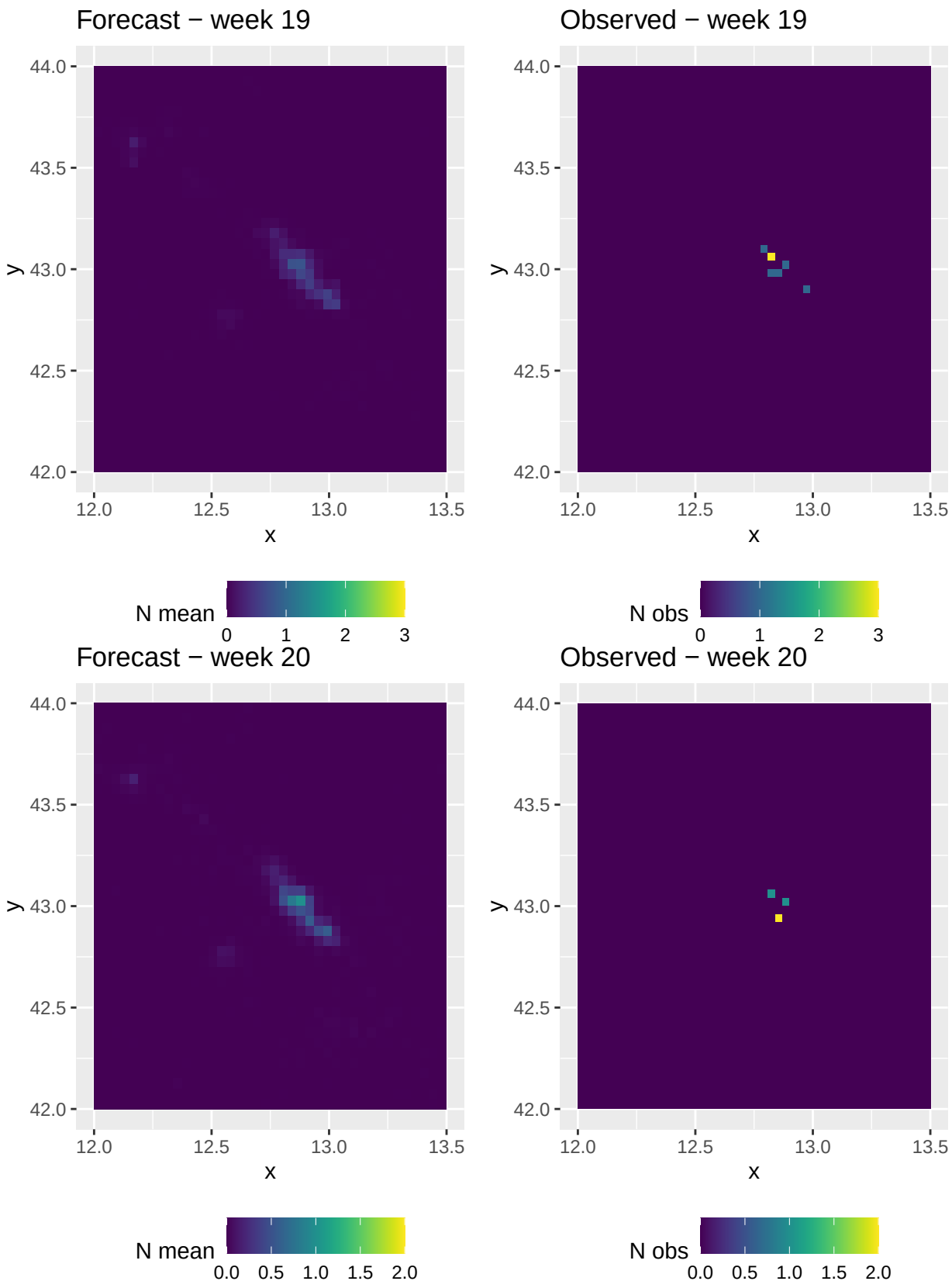

Forecast − week 9
Observed − week 9
Forecast − week 10
Observed − week 10
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
0 1 2 3 4 5
N obs
0 1 2 3 4 5
N mean
0.0 2.5 5.0 7.5 10.0
N obs
0.0 2.5 5.0 7.5 10.0


Forecast − week 11
Observed − week 11
Forecast − week 12
Observed − week 12
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0
1
2
3


Forecast − week 13
Observed − week 13
Forecast − week 14
Observed − week 14
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0 1 2 3 4 5 6
0.0 0.5 1.0 1.5 2.0


Forecast − week 15
Observed − week 15
Forecast − week 16
Observed − week 16
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
0.0 0.5 1.0 1.5 2.0
N obs
0.0 0.5 1.0 1.5 2.0
N mean
0 1 2 3 4 5
N obs
0 1 2 3 4 5


Forecast − week 17
Observed − week 17
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
0.00 0.25 0.50 0.75 1.00
N obs
0.00 0.25 0.50 0.75 1.00
Forecast − week 18
Observed − week 18
0
1
2
3
4


Forecast − week 19
Observed − week 19
Forecast − week 20
Observed − week 20
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
N mean
N obs
0
1
2
3
0.0
0.5
1.0
1.5
2.0

Weekly forecast - Probability of activity spatial distribution

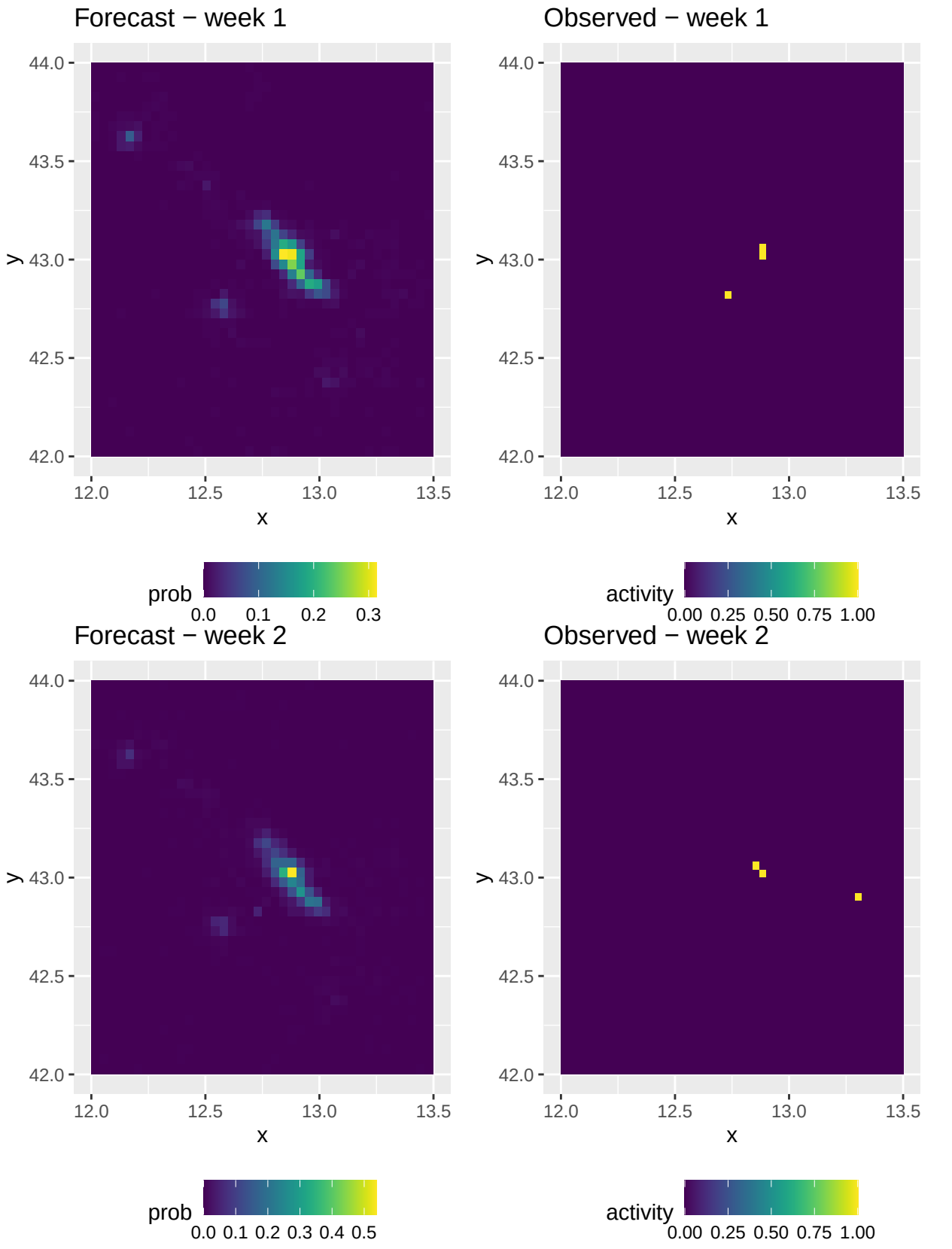
Below we can do the same for the probability of activity. Here the only difference is that instead of updating the cell value with the number of simulated events in the cell we only add +1. In principle, we could make a function out of this code that takes as input another function used to update the cell's value.

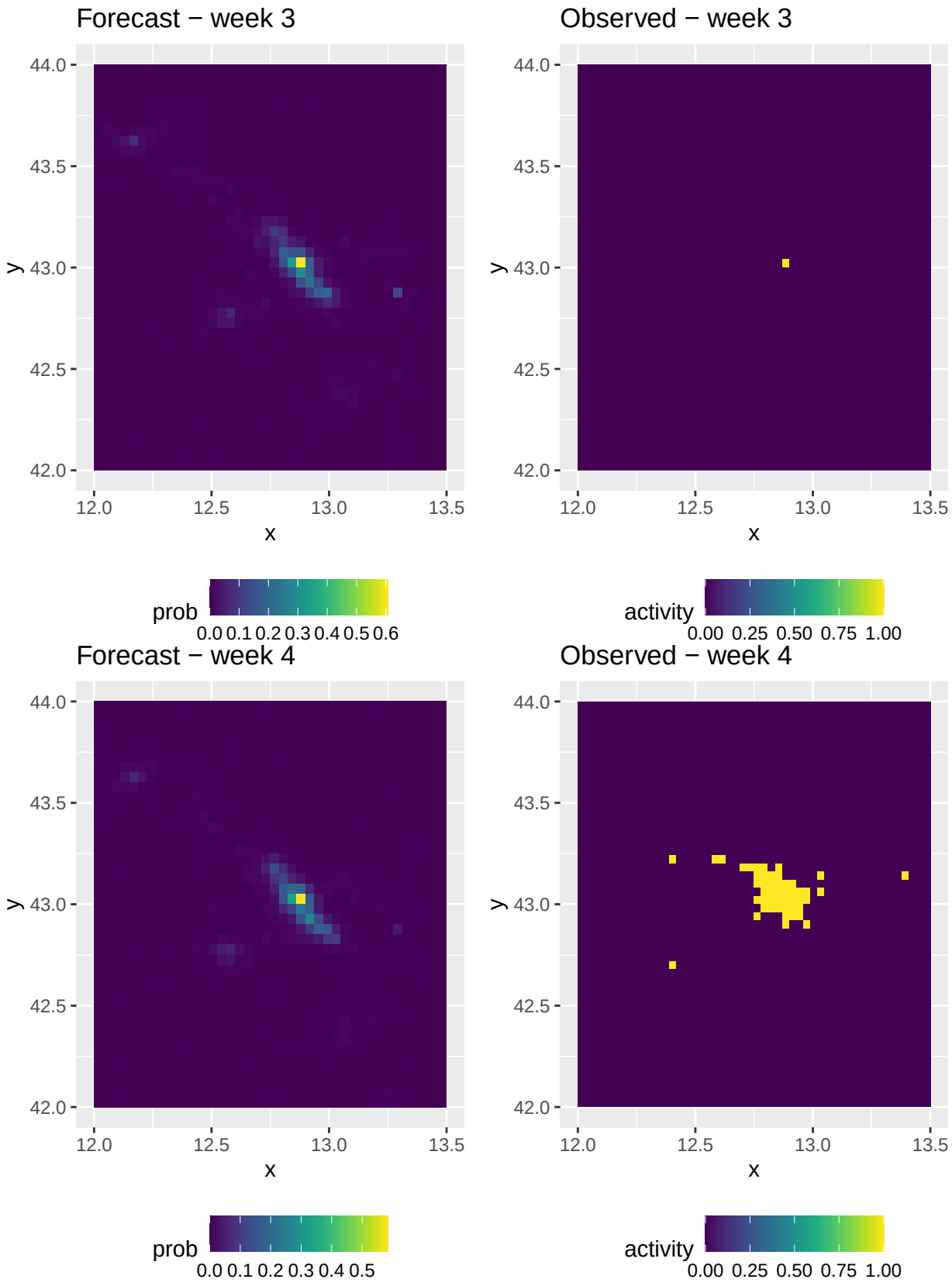
```
files.names <- list.files(path = 'fore_weeks/')
list.pixels.prob <- list(rep(NA, length(files.names)))
for(idx.file in 1:length(files.names)){
  raster.grid <- raster(bdy, nrows = 40, ncols = 40)
  raster.grid[] <- 0

  fore.name <- files.names[idx.file]
  load(paste0('fore_weeks/',fore.name))
  n.week <- as.numeric(gsub('.Rds', '', gsub('fore_week_', '', fore.name)))

  for(idx.car in 1:length(ll)){
    sample.cat <- ll[[idx.car]][,c('x','y')]
    counts_ <- table(cellFromXY(raster.grid, sample.cat))
    # the only difference from before.
    raster.grid[as.numeric(names(counts_))] <- raster.grid[as.numeric(names(counts_))] + 1
  }
  raster.grid[] <- raster.grid[]/length(ll)
  list.pixels.prob[[n.week]] <- as(raster.grid, 'SpatialPixelsDataFrame')
}
save(list.pixels.prob, file= 'list.pixels.prob.Rds')
```

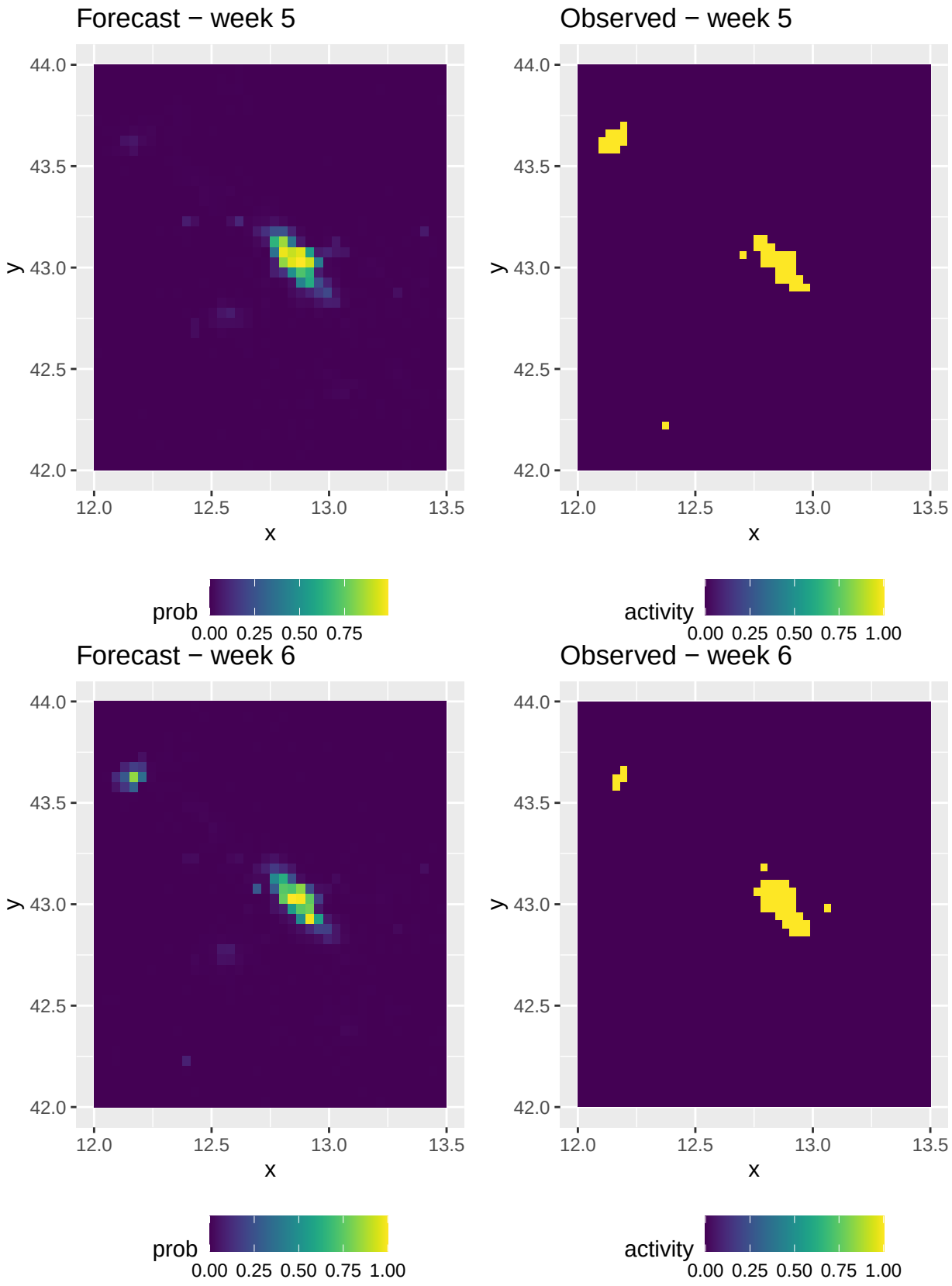
Also the plotting is the same

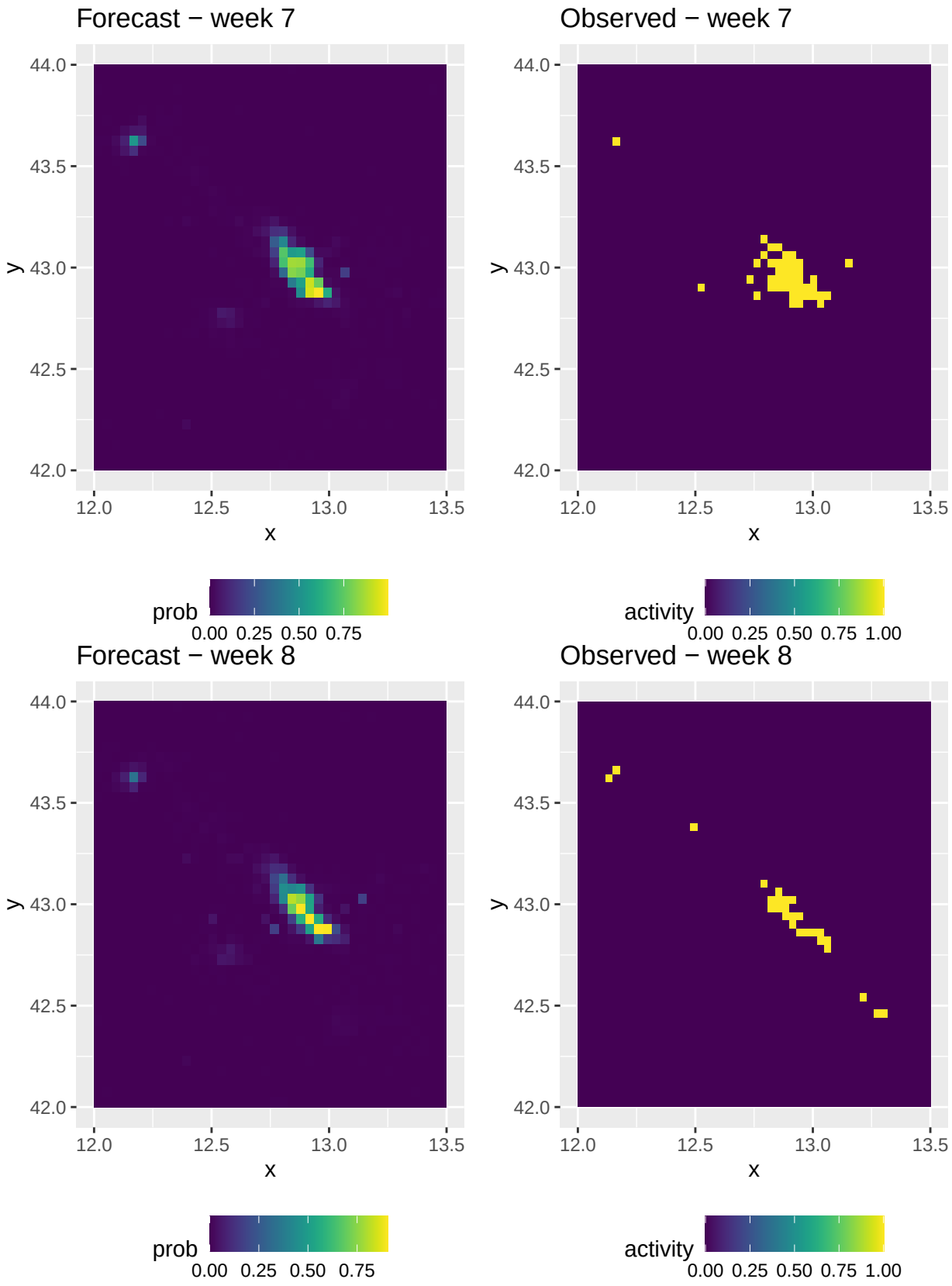
```
load('list.pixels.prob.Rds')
for(idxx in 1:length(weeks.to.fore)){
  week.data <- weeks.to.fore[[idxx]][,c('x', 'y')]
  raster.grid <- raster(bdy, nrows = 50, ncols = 50)
  raster.grid[] <- 0
  counts_ <- table(cellFromXY(raster.grid, week.data))
  # only difference from before
  raster.grid[as.numeric(names(counts_))] <- 1
  week.sp <- as(raster.grid, 'SpatialPixelsDataFrame')

  pl.pred <- ggplot() +
    gg(list.pixels.prob[[idxx]]) +
    labs(title = paste0('Forecast - week ', idxx), fill = 'prob') +
    theme(legend.position = 'bottom') +
    scale_fill_viridis()


  pl.obs <- ggplot() +
    gg(week.sp) +
    labs(title = paste0('Observed - week ', idxx), fill = 'activity') +
    theme(legend.position = 'bottom')+
    scale_fill_viridis()


  multiplot(pl.pred, pl.obs, cols = 2)
```

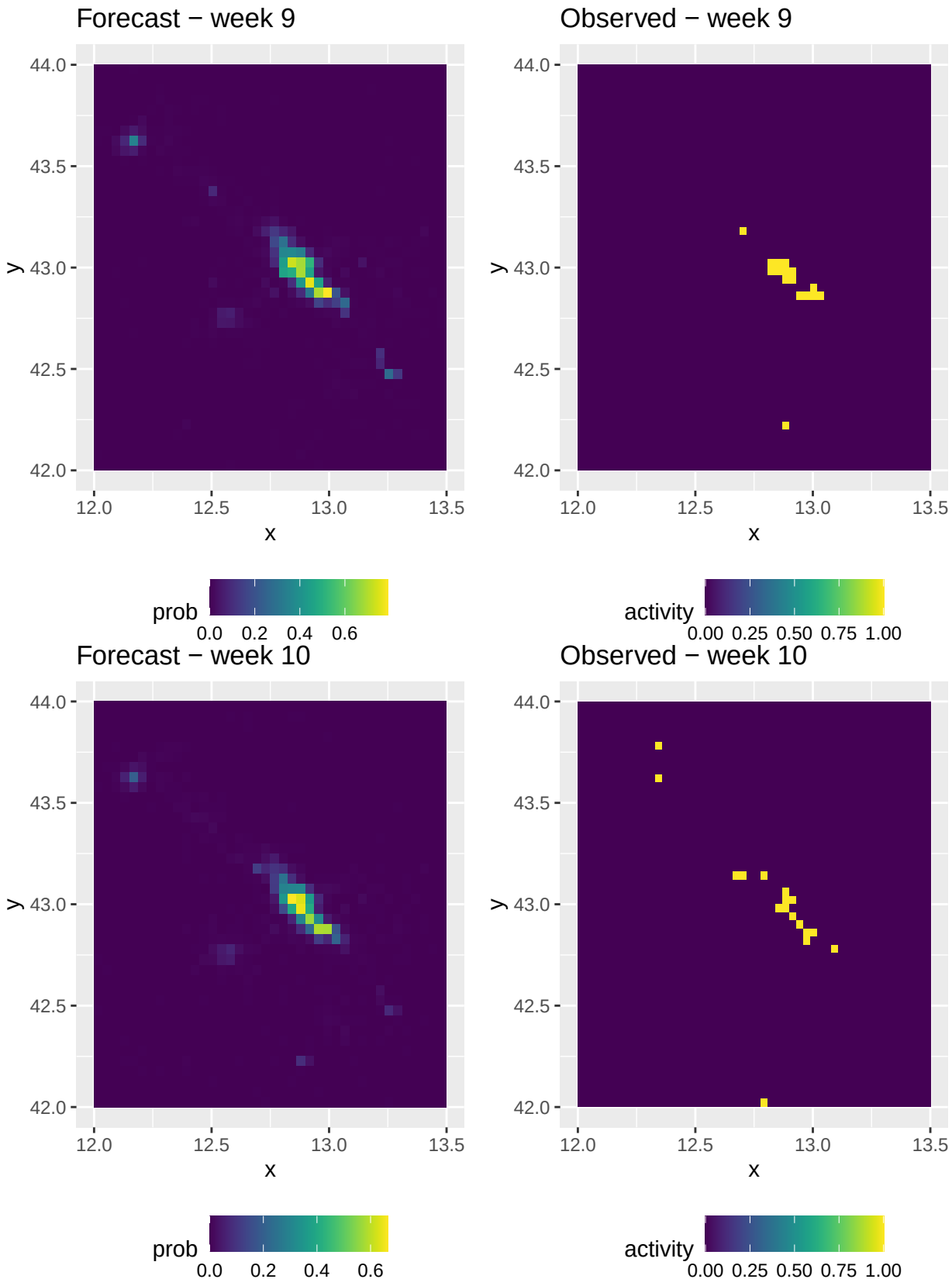
```
}
```

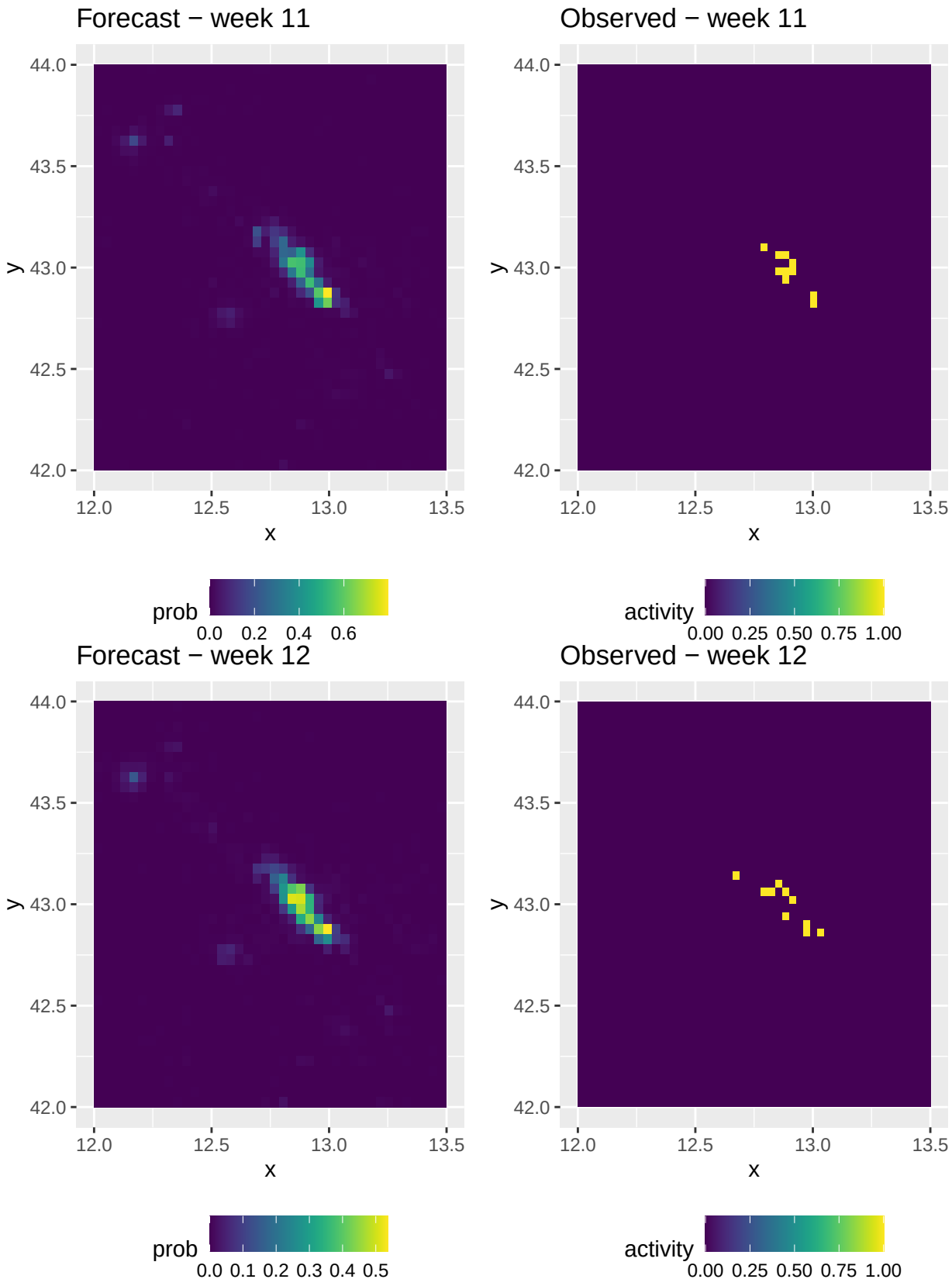

Forecast − week 1
Observed − week 1
Forecast − week 2
Observed − week 2
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3
0.0 0.1 0.2 0.3 0.4 0.5
activity
0.00 0.25 0.50 0.75 1.00

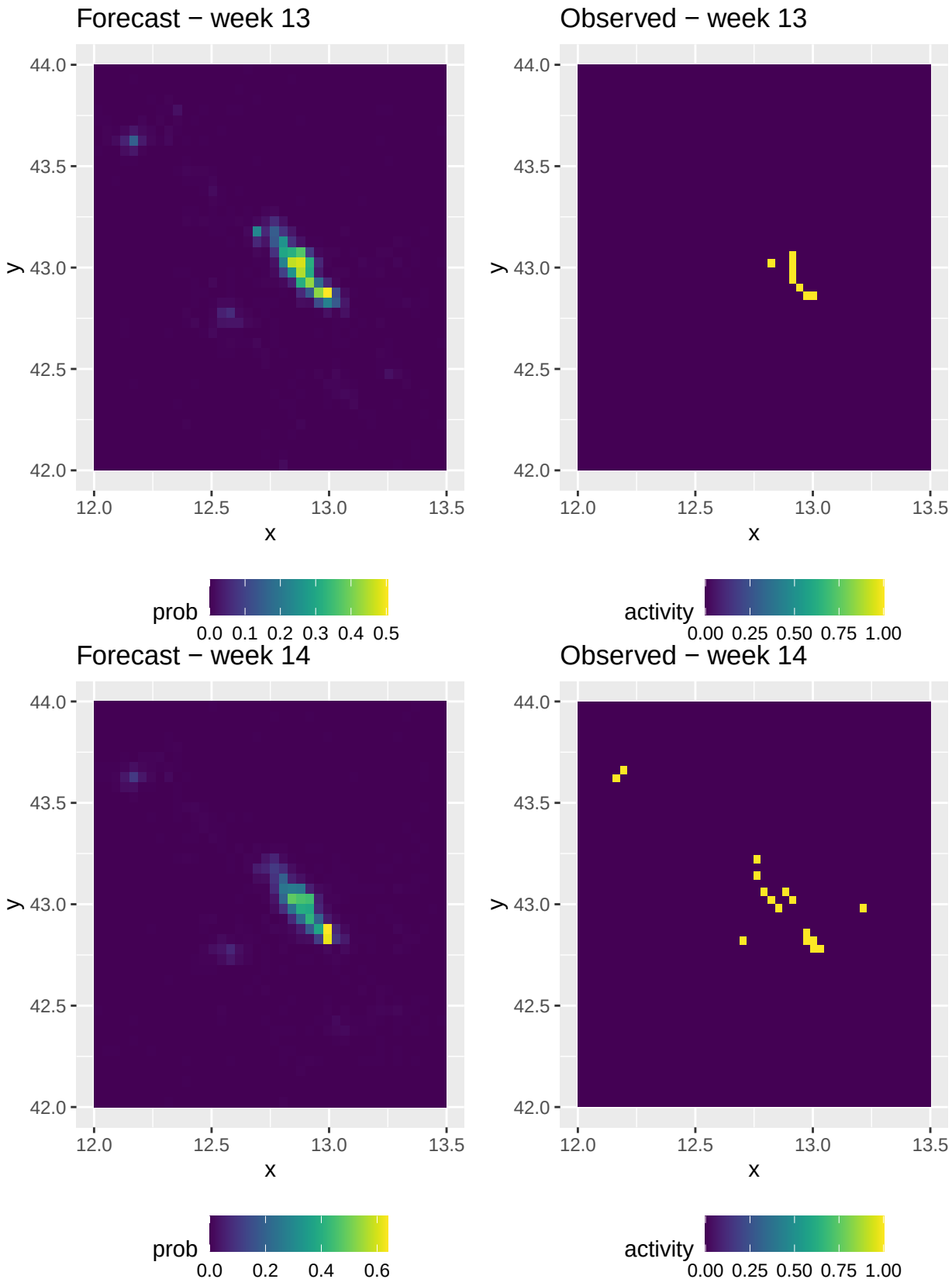

Forecast − week 3
Observed − week 3
Forecast − week 4
Observed − week 4
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3 0.4 0.5 0.6
activity
0.00 0.25 0.50 0.75 1.00
prob
0.0 0.1 0.2 0.3 0.4 0.5
activity
0.00 0.25 0.50 0.75 1.00

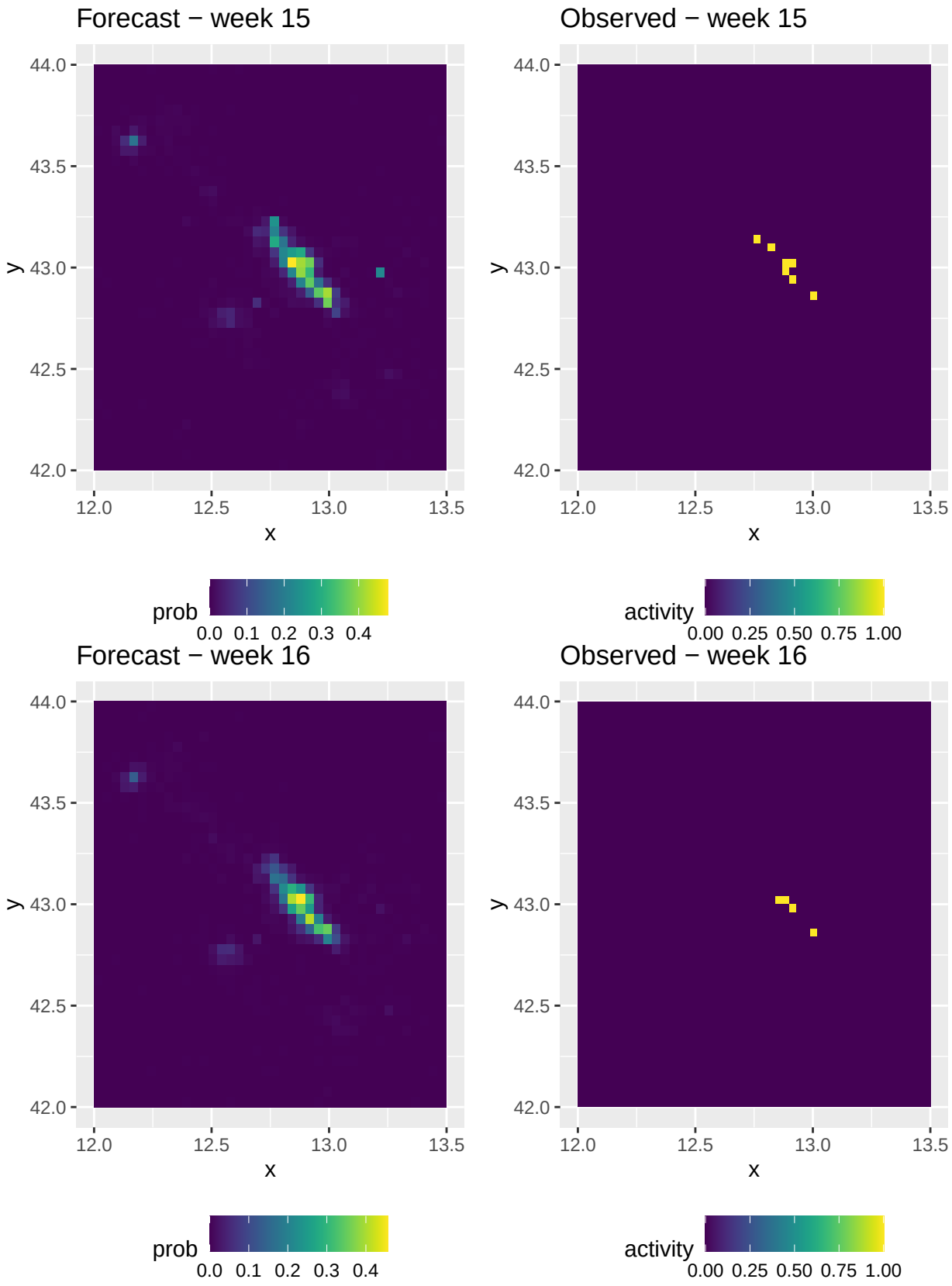

Forecast − week 5
Observed − week 5
Forecast − week 6
Observed − week 6
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.00 0.25 0.50 0.75
0.00 0.25 0.50 0.75 1.00
activity
0.00 0.25 0.50 0.75 1.00

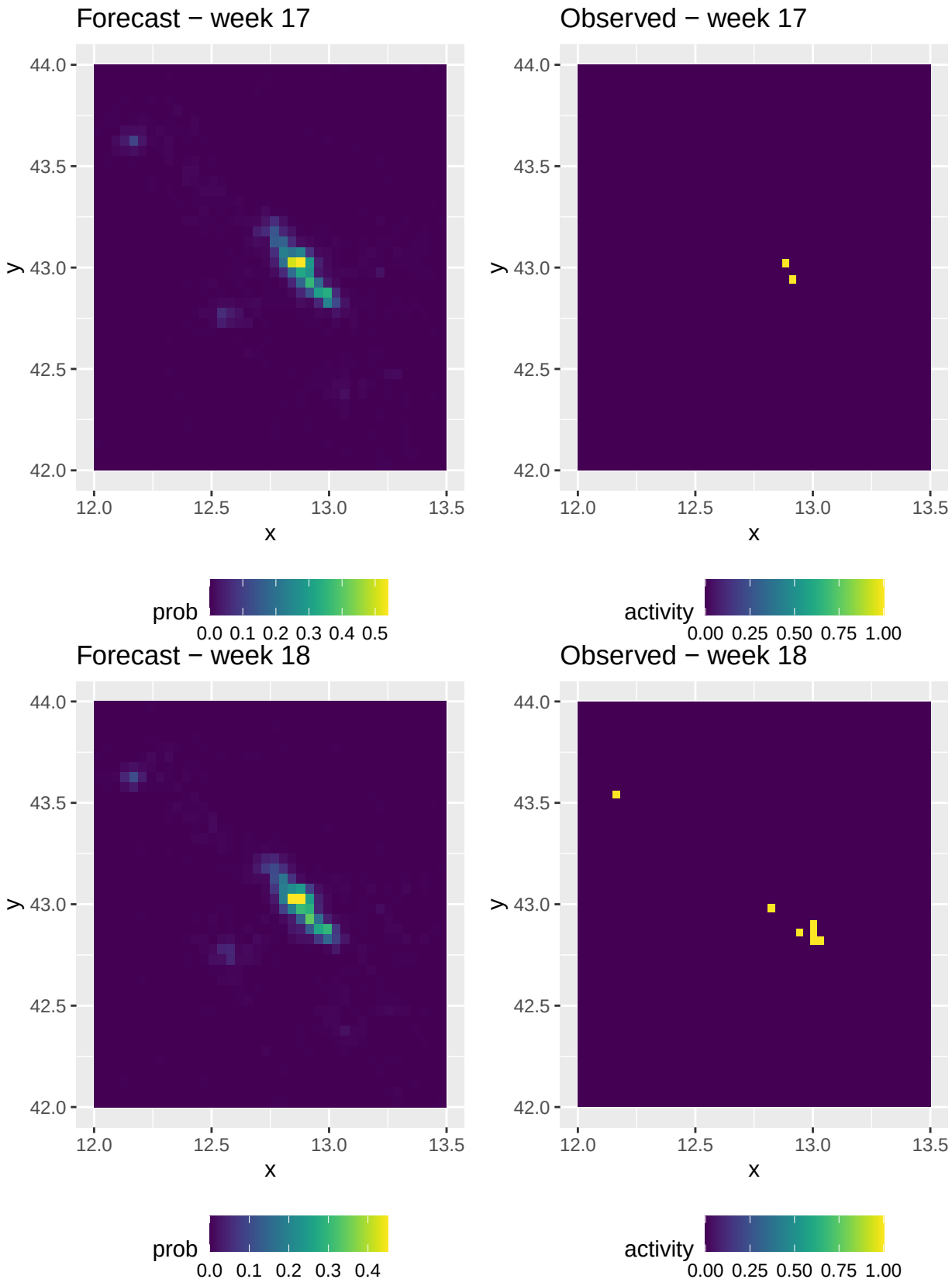

Forecast − week 7
Observed − week 7
Forecast − week 8
Observed − week 8
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.00 0.25 0.50 0.75
activity
0.00 0.25 0.50 0.75 1.00

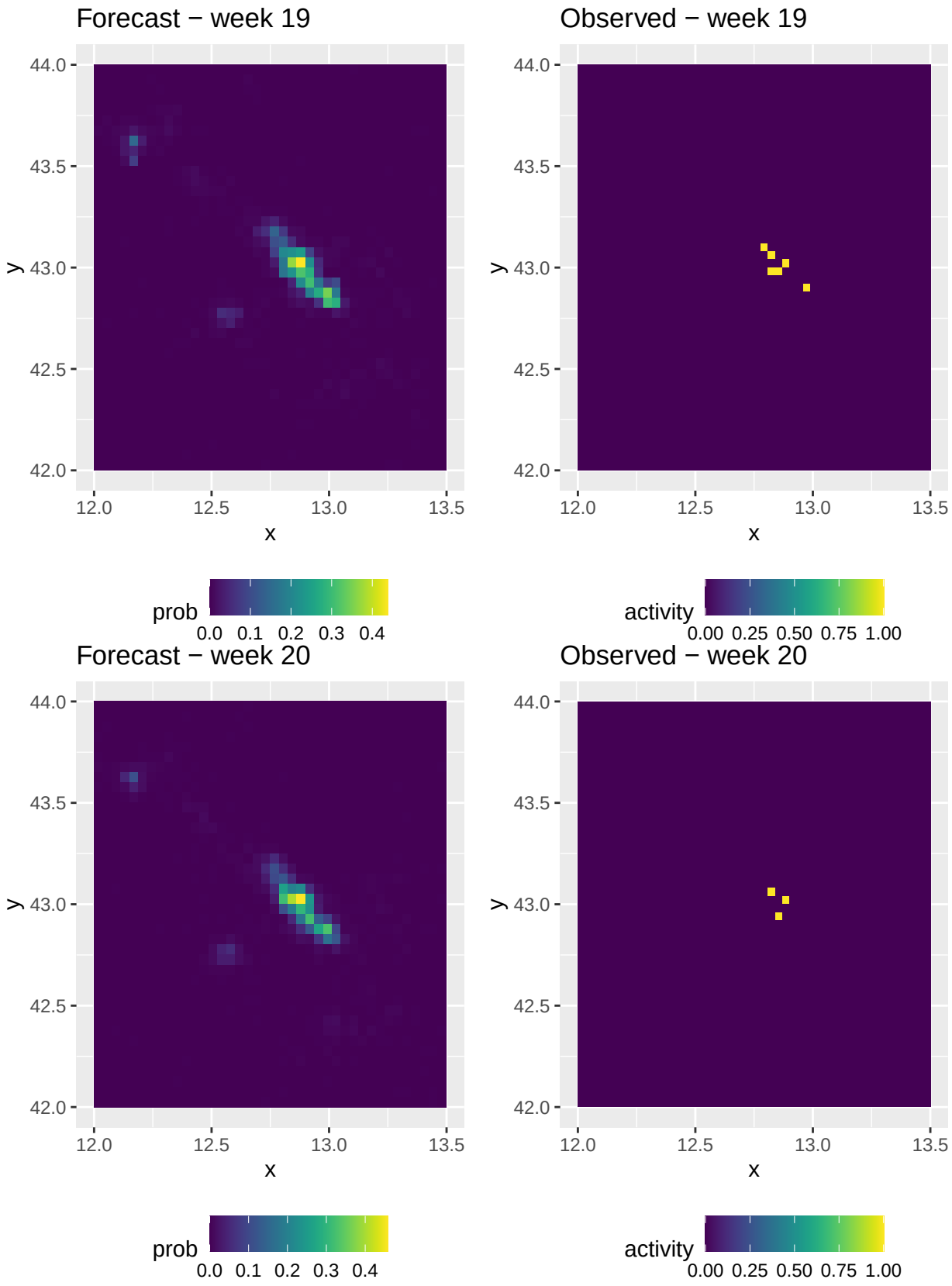

Forecast − week 9
Observed − week 9
Forecast − week 10
Observed − week 10
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.2 0.4 0.6
activity
0.00 0.25 0.50 0.75 1.00


Forecast − week 11
Observed − week 11
Forecast − week 12
Observed − week 12
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.2 0.4 0.6
activity
0.00 0.25 0.50 0.75 1.00
prob
0.0 0.1 0.2 0.3 0.4 0.5
activity
0.00 0.25 0.50 0.75 1.00


Forecast − week 13
Observed − week 13
Forecast − week 14
Observed − week 14
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3 0.4 0.5
0.0 0.2 0.4 0.6
activity
0.00 0.25 0.50 0.75 1.00


Forecast − week 15
Observed − week 15
Forecast − week 16
Observed − week 16
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3 0.4
activity
0.00 0.25 0.50 0.75 1.00


Forecast − week 17
Observed − week 17
Forecast − week 18
Observed − week 18
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3 0.4 0.5
activity
0.00 0.25 0.50 0.75 1.00
prob
0.0 0.1 0.2 0.3 0.4
activity
0.00 0.25 0.50 0.75 1.00


Forecast − week 19
Observed − week 19
Forecast − week 20
Observed − week 20
44.0
43.5
43.0
42.5
42.0
y
12.0
12.5
13.0
13.5
x
prob
0.0 0.1 0.2 0.3 0.4
activity
0.00 0.25 0.50 0.75 1.00

Model on partial data

The results, expecially regarding the number of points are not ideal. This may be due to the fact that we have fitted a model on a longer period than the one that we used to testing, and specifically, that we are testing the model only on the most active weeks. Therefore, it is interesting to see the differences if we fit the model only on the weeks used for testing.

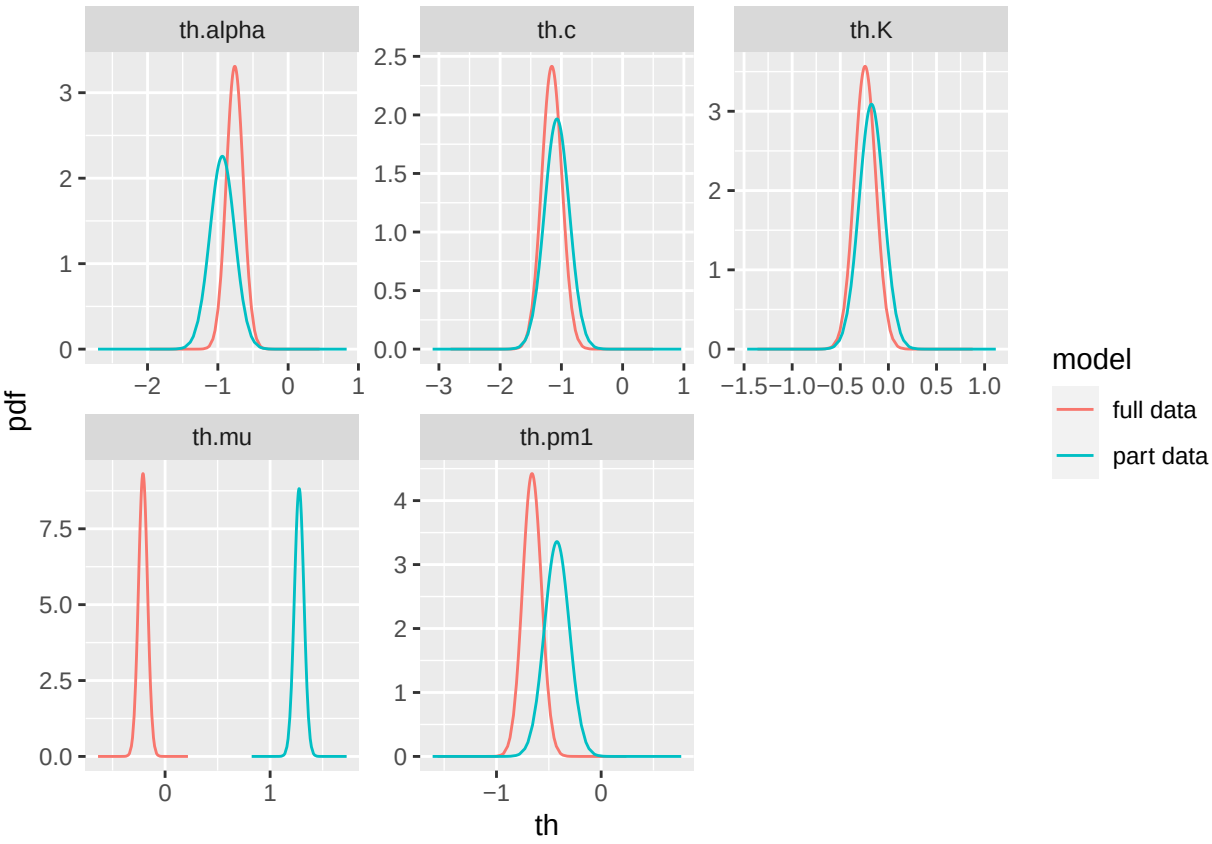
```
# set data and time interval
week.total <- bind_rows(weeks.to.fore)
T1.w <- min(week.total$ts) - 0.01
T2.w <- max(week.total$ts) + 0.01
```

```
# fit the model
col.fit_part <- ETAS.fit.B_bkg(sample.s = week.total, # data.frame representing data points
                               N.breaks.min = 5,  # minimum number of bins (Integral decomposition)
                               max.length = (T2 - T1)/30,  # maximum length of a time bin
                               Sigma = Sigma.p, # Sigma matrix for spatial trig function
                               prior.mean = c(0,-1,0,0,-1),  # prior mean for thetas of temporal ETAS
                               prior.prec = c(2,2,2,2,2), # prior precision for thetas of temporal ETAS
                               M0 = 2.5, # magnitude of completeness
                               T1 = T1.w, T2 = T2.w, # extremes of time interval
                               bdy = bdy, # SpatialPolygon representing study region (has to be squared)
                               bru.opt = # bru options
                                 list(inla.mode = 'experimental', # inla mode
                                      bru_max_iter = 40, # max number of iterations
                                      bru_verbose = 3), # verbose changes what inlabru prints
                               bin.strat = 'alt') # strategy for the bins (recommend to use alt when T2>
save(col.fit_part, file = 'col.fit_part.Rds')
```

Parameters' posterior comparison

Below a comparison of the posterior of the parameters, we can see clear differences in θ_μ indicating that the partial model has an higher background rate. The other parameters are similar, except for θ_α which is smaller for the partial model, meaning that we can expect slightly less triggered events.

```
load('col.fit_part.Rds')
# difference in the parameters' posterior
eff.names <- names(col.fit_part$marginals.fixed)
df.eff <- foreach(eff = eff.names, .combine = rbind) %do% {

  data.frame(th = c(col.fit_sigma$marginals.fixed[[eff]][,1],
                    col.fit_part$marginals.fixed[[eff]][,1]),
             pdf = c(col.fit_sigma$marginals.fixed[[eff]][,2],
                     col.fit_part$marginals.fixed[[eff]][,2]),
             model = c(rep('full data', nrow(col.fit_sigma$marginals.fixed[[eff]])),
                       rep('part data', nrow(col.fit_part$marginals.fixed[[eff]]))),
             eff = eff
             )
}


ggplot(df.eff, aes(x = th, y = pdf, color = model)) +
  geom_line() +
  facet_wrap(facets = vars(eff), scales = 'free')
```



Number of events comparison

Below a comparison of the posterior distribution of the number of points which has been decomposed in background component and triggered component. We can see a clear difference in the background part and a smaller difference in the triggered part.

```
# retrieve distribution of the total number of point
N.obs.weeks <- nrow(week.total)
N.distr.full <- predict(col.fit_sigma, df.bru[df.bru$ts < T2.w,],
                        ~ data.frame(N = 600:850,
                             pdf = dpois(600:850,
                                         lambda = Lambda_(th.mu = th.mu,
                                                          th.K = th.K,
                                                          th.alpha = th.alpha,
                                                          th.c = th.c,
                                                          th.p = th.pm1,
                                                          Sigma = Sigma.p,
                                                          M0 = 2.5, th = ts, xh = x, yh = y,
                                                          mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                             ))

N.distr.part <- predict(col.fit_part, df.bru[df.bru$ts < T2.w,],
                        ~ data.frame(N = 850:1200,
                             pdf = dpois(850:1200,
                                         lambda = Lambda_(th.mu = th.mu,
                                                          th.K = th.K,
                                                          th.alpha = th.alpha,
```

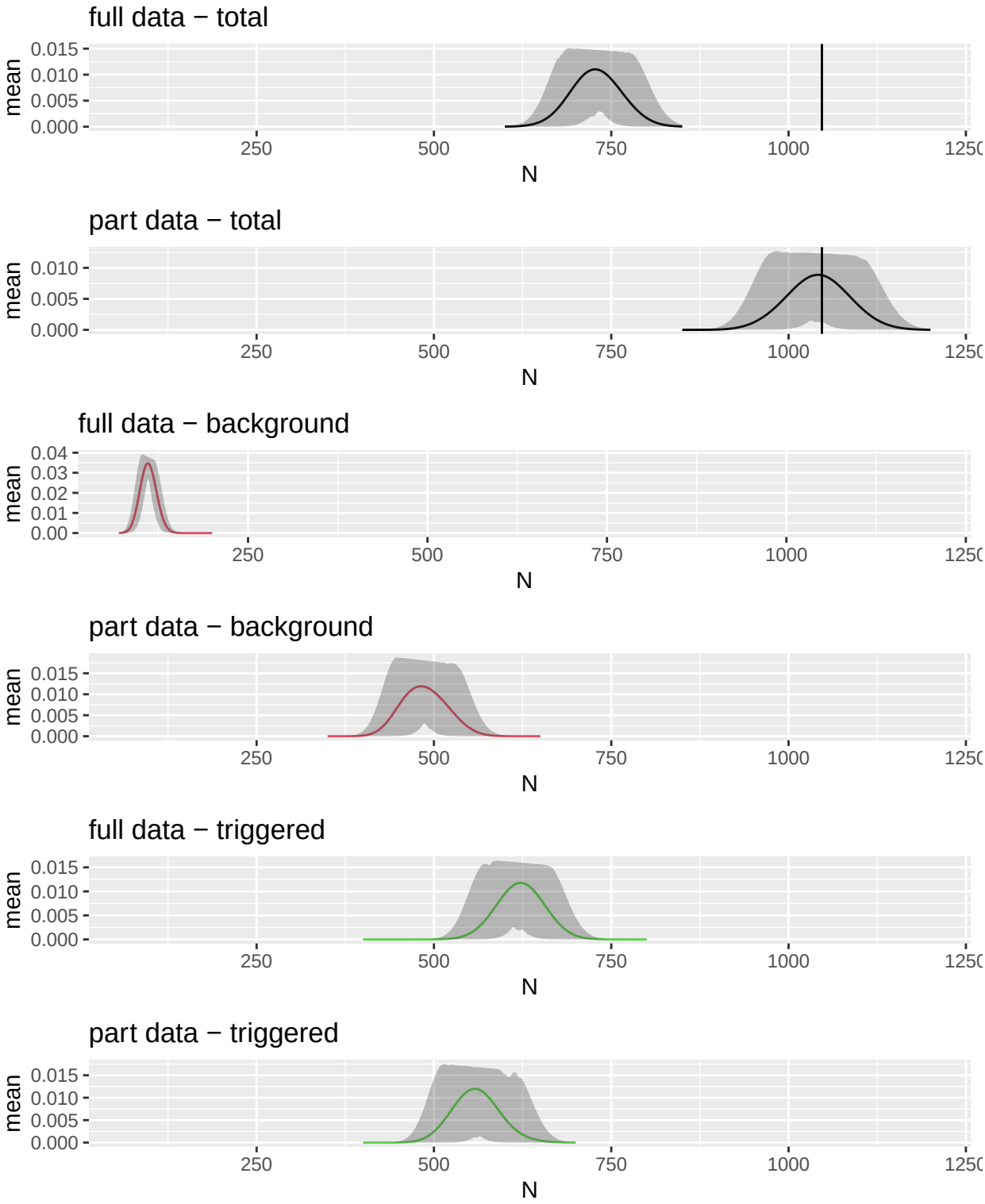
```
                                                    th.c = th.c,
                                                    th.p = th.pm1,
                                                    Sigma = Sigma.p,
                                                    M0 = 2.5, th = ts, xh = x, yh = y,
                                                    mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                                ))


# create plots
pl.full <- plot(N.distr.full) +
  geom_vline(xintercept = N.obs.weeks) + labs(title = 'full data - total') + xlim(70,1200)

pl.part <- plot(N.distr.part) +
  geom_vline(xintercept = N.obs.weeks) + labs(title = 'part data - total') + xlim(70,1200)
```

```
# retrieve distribution of the total number of point - background rate
N.distr.full.bkg <- predict(col.fit_sigma, df.bru[df.bru$ts < T2.w,],
                            ~ data.frame(N = 70:200,
                                  pdf = dpois(70:200,
                                              lambda = Lambda_(th.mu = th.mu,
                                                               th.K = th.K,
                                                               th.alpha = th.alpha,
                                                               th.c = th.c,
                                                               th.p = th.pm1,
                                                               Sigma = Sigma.p,
                                                               M0 = 2.5, th = ts, xh = x, yh = y,
                                                               mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                                                               part = 'background'))
                                  ))

N.distr.part.bkg <- predict(col.fit_part, df.bru[df.bru$ts < T2.w,],
                            ~ data.frame(N = 350:650,
                                  pdf = dpois(350:650,
                                              lambda = Lambda_(th.mu = th.mu,
                                                               th.K = th.K,
                                                               th.alpha = th.alpha,
                                                               th.c = th.c,
                                                               th.p = th.pm1,
                                                               Sigma = Sigma.p,
                                                               M0 = 2.5, th = ts, xh = x, yh = y,
                                                               mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                                                               part = 'background'))
                                  ))



pl.full.bkg <- plot(N.distr.full.bkg, color = 2) + labs(title = 'full data - background') + xlim(70,120

pl.part.bkg <- plot(N.distr.part.bkg, color = 2) + labs(title = 'part data - background') + xlim(70,120
```

```
# retrieve distribution of the total number of point - triggered part
N.distr.full.trig <- predict(col.fit_sigma, df.bru[df.bru$ts < T2.w,],
                            ~ data.frame(N = 400:800,
                                  pdf = dpois(400:800,
```

```
                                        lambda = Lambda_(th.mu = th.mu,
                                                         th.K = th.K,
                                                         th.alpha = th.alpha,
                                                         th.c = th.c,
                                                         th.p = th.pm1,
                                                         Sigma = Sigma.p,
                                                         M0 = 2.5, th = ts, xh = x, yh = y,
                                                         mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                                                         part = 'triggered'))
                                ))

N.distr.part.trig <- predict(col.fit_part, df.bru[df.bru$ts < T2.w,],
                             ~ data.frame(N = 400:700,
                                pdf = dpois(400:700,
                                        lambda = Lambda_(th.mu = th.mu,
                                                         th.K = th.K,
                                                         th.alpha = th.alpha,
                                                         th.c = th.c,
                                                         th.p = th.pm1,
                                                         Sigma = Sigma.p,
                                                         M0 = 2.5, th = ts, xh = x, yh = y,
                                                         mh = mags, T1 = T1.w, T2 = T2.w, bdy = bdy
                                                         part = 'triggered'))
                                ))



pl.full.trig <- plot(N.distr.full.trig, color = 3) + labs(title = 'full data - triggered') + xlim(70,12

pl.part.trig <- plot(N.distr.part.trig, color = 3) + labs(title = 'part data - triggered') + xlim(70,12
```

```
multiplot(pl.full, pl.part, pl.full.bkg, pl.part.bkg, pl.full.trig, pl.part.trig)
```



Extensions implementable with the present code

The code provided here is related to the Colfiorito sequence however it can be generalized to any other sequence. Besides the fact that this sequence is particularly difficult to predict, the present example can be improved in many ways. Possible examples may be:

1. Considering a better model for the background field, here I have considered a simple model Intercept + random effect, covariates may be introduced there. Also, a better mesh may be considered, the one considered here is okay but a finer one may be used.
2. Use declustering algorithms to fit the background model only on the observations assigned as background instead that on the whole data.
3. Consider the effect of different covariance matrices (which is considered fixed for now). Here, I have set the standard deviations equal to the median of the observed minimum distances between points and the correlation equal to the correlation between the coordinates. Many possibilities here can be explored and see how much our choices impact the results. Also, we may use declustering algorithms to use only part of the data.