White fishes

Ilaria Pia

18/03/2020

White fishes

The Gulf of Bothnia is a brackish water basin between Sweden and Finland covering an area of approximately 600×120 km. Its coastal areas play a central role in the ecosystem and many Baltic fish stocks are dependent on the coastal regions for their reproduction. White fish is a fresh water origin fish species that is found also from the northern parts of the brackish water Gulf of Bothnia (see Veneranta et al. (2013)). In addition White fishes are caught and sold for human consumption and, hence, they are economically important for local fishermen. It spawns in shallow coastal areas in spring. The aim of the following analysis is to study how White fish spawns distribution is affected by environmental changes.

Data exploration

A number of sites along the Finnish and Swedish coastal region in the Gulf of Bothnia were sampled during 2009-2011 and several environmental variables were measured. Let's have a look at the data:

```
setwd("/home/piailari/Documents/thesis")

library("rstan")
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(matrixStats)
options(mc.cores = parallel::detectCores())

library("raster")
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:rstan':
##
##     extract
```

```
library(Rcpp)
library(readxl)

sourceCpp("Matern32Covariance.cpp")
sourceCpp("expCovariance.cpp")

linearCovariance <- function(x1,x2,sigma2) {
  # K = exponentialCovariance(x1,x2,l,sigma2)
  #
  # A function that takes matrices x1 and x2 of input locations, lengthscale
  # parameter l and magnitude parameter sigma2 and constructs a covariance
  # matrix between f(x1) and f(x2) corresponding to an exponential covariance
  # function.

  K = as.matrix(x1)%*%diag(sigma2)%*%t(as.matrix(x2))
  return(K)
}

# Load the data
whitefish.dat.cov = read.table("white_fishes_final.txt", header=TRUE)
whitefish.raster = read.table("white_fishes_final_raster.txt", header=TRUE)
# full data, to get target variable
setwd("/home/piailari/Documents/BayesianDataAnalysis/exercises/week3/exercise2b")
whitefish.dat = read_xlsx("bsg653_3.xlsx")
colnames(whitefish.dat)[15:16]= c("N_etrs89", "E_etrs89")

head(whitefish.dat.cov)
```

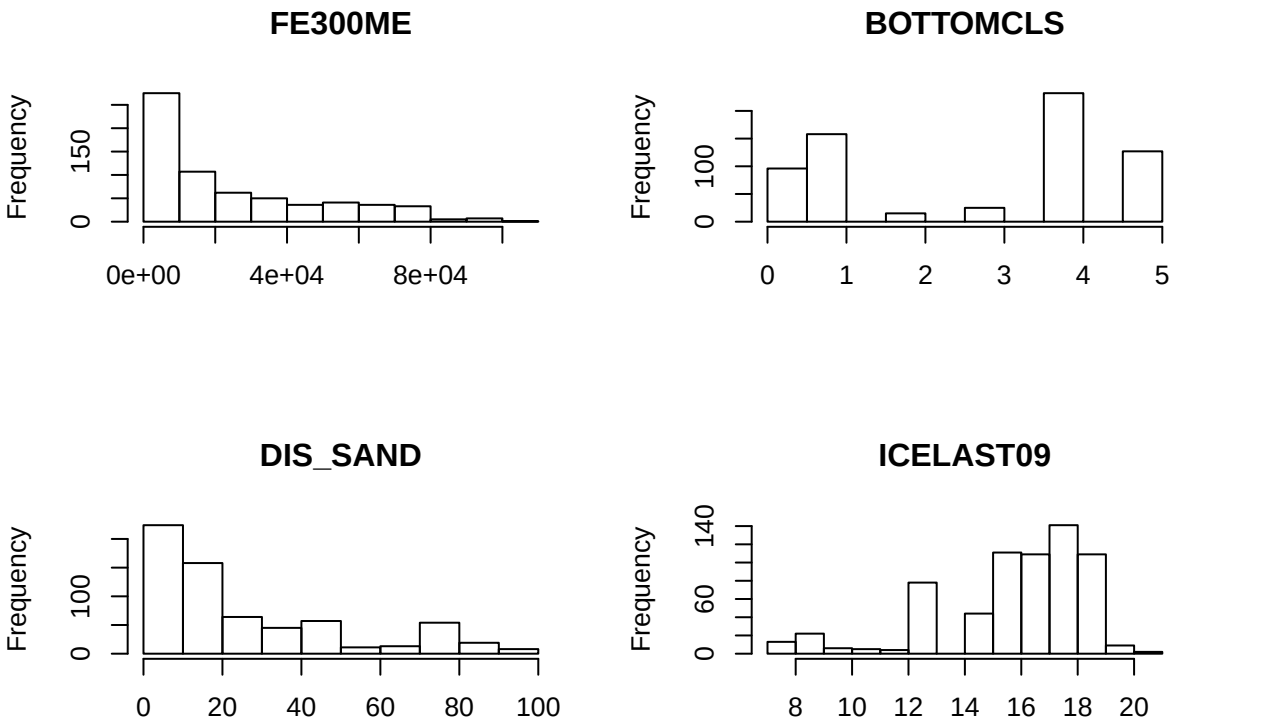
```
##   N_etrs89 E_etrs89 FE300ME BOTTOMCLS DIS_SAND ICELAST09 RIVERS SAL910WIN
## 1  4632492  4875736    7443         1       17         16  12053    3.5246
## 2  4632150  4876171    7378         5       17         16  12002    3.5246
## 3  4633801  4875794   30121         1       17         16  14062    3.5348
## 4  4637652  4878980    7631         5       15         16  18521    3.5348
## 5  4637717  4878414    7537         1       15         16  18458    3.5348
## 6  4638320  4879176    3290         1       14         16  18515    3.5348
##   DIST20M CHL_A  TEMP09M  SALT09M
## 1  152308 36.31 4.063705 2.793786
## 2  148793 36.40 4.016508 2.780895
## 3  160970 36.31 4.012287 2.779378
## 4   71396 36.31 3.960890 2.719728
## 5   91882 36.31 4.022866 2.724489
## 6   79882 36.31 3.962358 2.717036
```

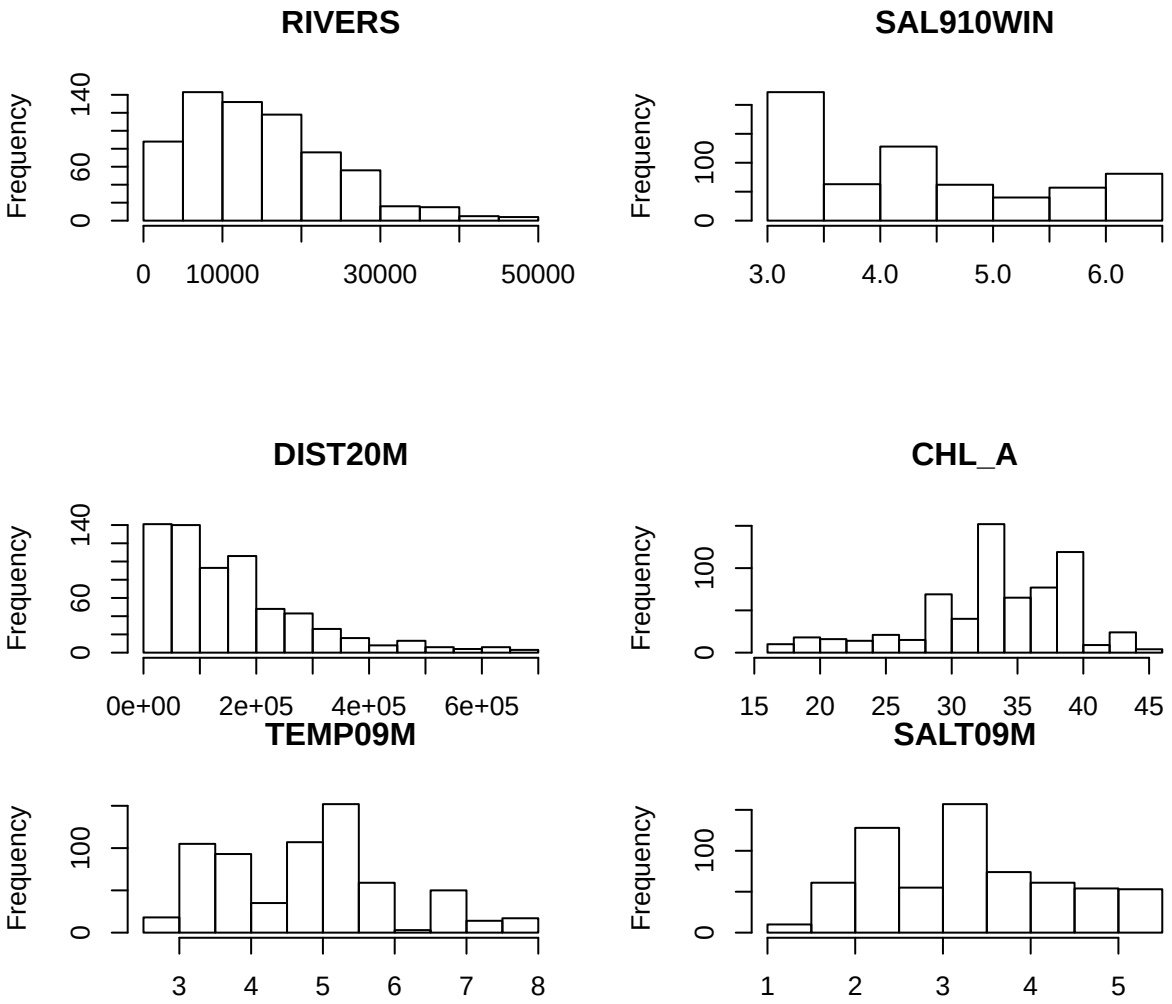
```
summary(whitefish.dat.cov)
```

```
##     N_etrs89          E_etrs89          FE300ME         BOTTOMCLS
##  Min.   :4301013   Min.   :4708348   Min.   :     9   Min.   :0.000
##  1st Qu.:4497606   1st Qu.:4868414   1st Qu.:  4584   1st Qu.:1.000
##  Median :4542586   Median :4906140   Median : 14700   Median :4.000
##  Mean   :4577380   Mean   :4895672   Mean   : 23964   Mean   :2.796
##  3rd Qu.:4725141   3rd Qu.:4932318   3rd Qu.: 38756   3rd Qu.:4.000
##  Max.   :4804683   Max.   :5036542   Max.   :100218   Max.   :5.000
##     DIS_SAND       ICELAST09         RIVERS        SAL910WIN
##  Min.   : 2.00   Min.   : 7.00   Min.   :  199   Min.   :3.329
```

```
##  1st Qu.: 9.00   1st Qu.:15.00   1st Qu.: 8301   1st Qu.:3.368
##  Median :15.00   Median :17.00   Median :13437   Median :4.489
##  Mean   :27.18   Mean   :16.21   Mean   :15041   Mean   :4.414
##  3rd Qu.:40.00   3rd Qu.:18.00   3rd Qu.:20814   3rd Qu.:5.114
##  Max.   :94.00   Max.   :21.00   Max.   :48435   Max.   :6.468
##     DIST20M           CHL_A          TEMP09M         SALT09M
##  Min.   :     0   Min.   :16.28   Min.   :2.624   Min.   :1.201
##  1st Qu.: 60000   1st Qu.:30.01   1st Qu.:3.785   1st Qu.:2.332
##  Median :126852   Median :33.50   Median :4.909   Median :3.335
##  Mean   :155008   Mean   :33.22   Mean   :4.834   Mean   :3.271
##  3rd Qu.:209220   3rd Qu.:37.71   3rd Qu.:5.446   3rd Qu.:4.070
##  Max.   :697705   Max.   :44.33   Max.   :7.604   Max.   :5.412
```

```r
par(mfrow =c(2,2))
for (i in 3:ncol(whitefish.dat.cov)) {
  hist(whitefish.dat.cov[,i], main = colnames(whitefish.dat.cov)[i], xlab="")
}
```

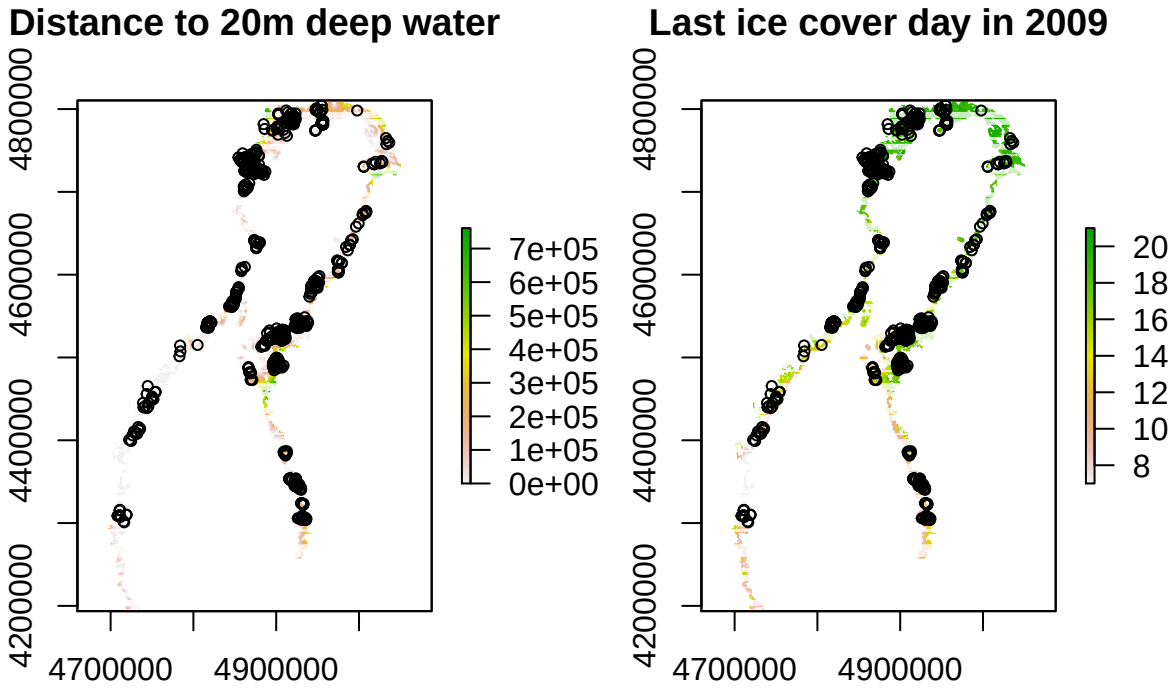




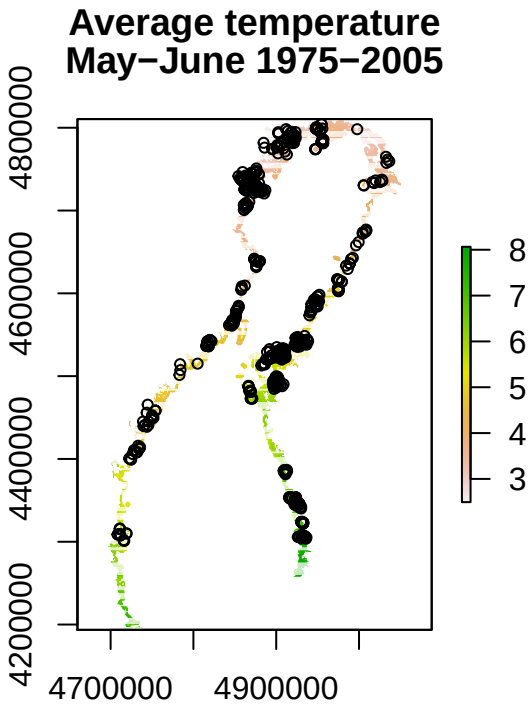
The coordinate system in the data is ETRS89 (http://www.euref.eu/euref_egrs.html) which is a 3D Cartesian coordinate system constructed specifically to European areas. Hence, we can use the coordinates as such. the coordinates are N_etrs89 and E_etrs89.

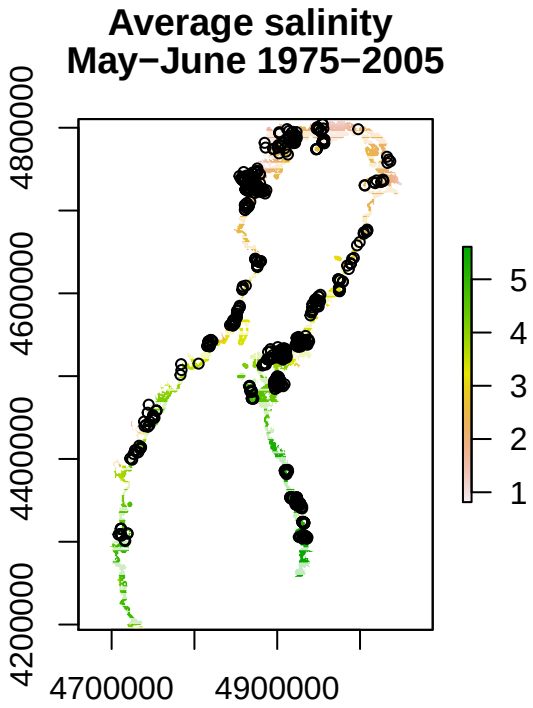
We now examine the environmental covariates by plotting few of the raster layers

```
# Visualize few environmental covariates
e <- extent(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89))
r <- raster(e, ncol=length(unique(whitefish.raster$E_etrs89)), nrow=length(unique(whitefish.raster$N_et
# Visualize the study area

par(mfrow=c(1,2))
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$DIST20M,
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main="Distance to 20m d
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)

z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$ICELAST0
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main="Last ice cover da
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
```




```
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$TEMP09M,
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average tempera
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)

z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$SALT09M,
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average salinit
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
```





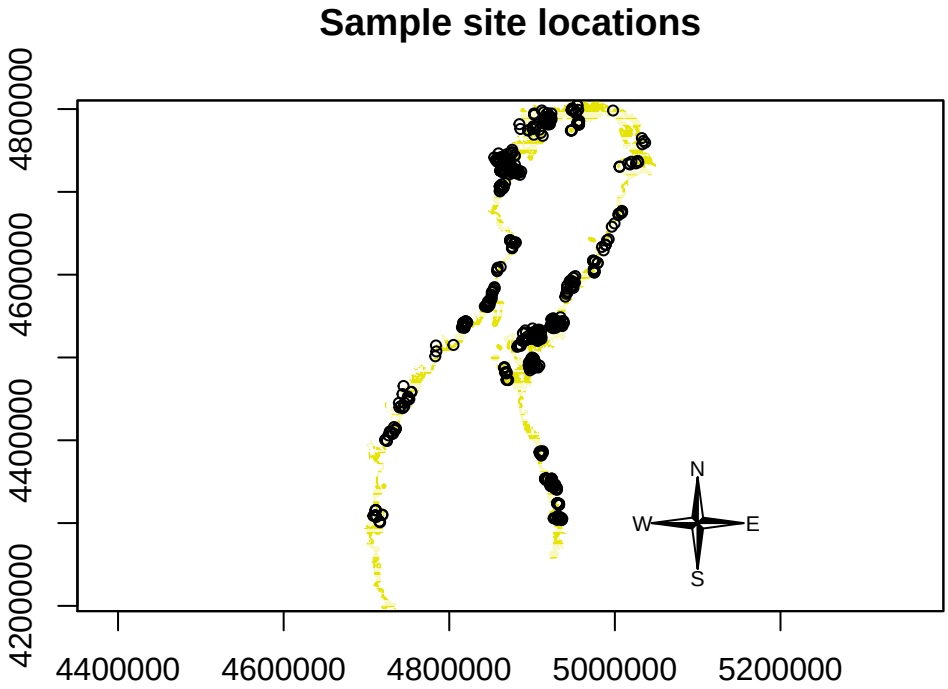
Beside the coordinates the original dataset contains the variable 'YEAR' and other 33 environmental variables. The variables 'BOTTOM' And 'BOTTOMCOV' are categorical.

Sites locations

```
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, rep(1,nrow(whitefish.raste
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
      ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main="Sample site loca
 # Plot the locations of sampling sites
 points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
 library(GISTools)
```

```
## Loading required package: maptools
## Checking rgeos availability: TRUE
## Loading required package: RColorBrewer
## Loading required package: MASS
##
## Attaching package: 'MASS'
## The following objects are masked from 'package:raster':
##
##     area, select
## Loading required package: rgeos
## rgeos version: 0.5-2, (SVN revision 621)
##  GEOS runtime version: 3.6.2-CAPI-1.10.2
##  Linking to sp version: 1.4-1
##  Polygon checking: TRUE
```

```
 compassRose(5100000,4300000, cex = 0.7)
```



Covariates

Species distribution modeling is directly related to inferring species' responses to environmental factors. We now select the environmental covariates. Gulf of Bothnia hosts rich variety of environmental conditions. Coastal areas are affected by inflows from land as well as shallow and complex topography. In the scale of Gulf of Bothnia, there is a gradient in river influence, salinity, temperature and length of ice cover period from north to south. We used seven real-valued and one categorical abiotic environmental covariates. These are the quantities we will consider :

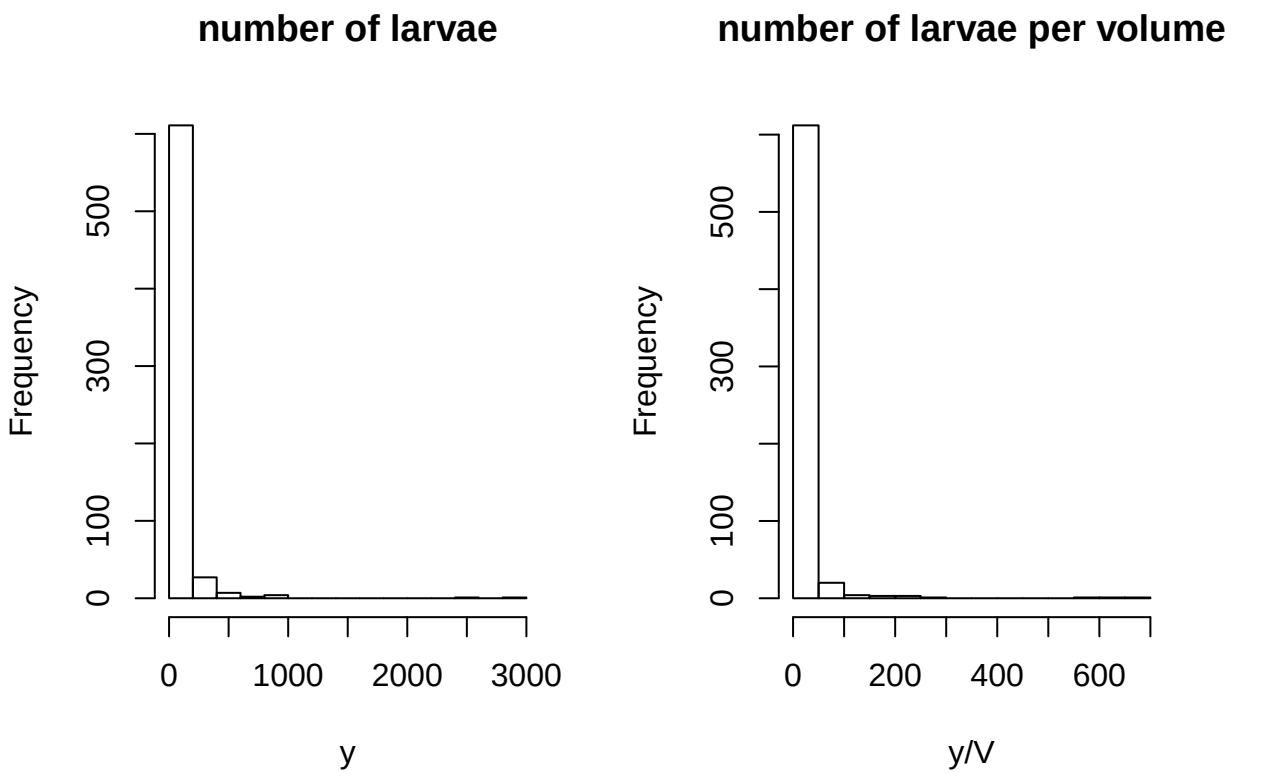
- BOTTOMCLS - Bottom type classification, a categorical variable with classes 0 = not shallow 1 = open water 2 = other 3 = sand 4 = sand/mud 5 = sand/stone
- DIS_SAND - distance to sandy shore, continuous variable
- FE300ME - The average fetch (openness/exposure) over all directions, continuous variable
- ICELAST09 - The last ice cover date in winter 2009-10, continuous variable
- RIVERS - Influence of rivers (~weighted average distance to river mouths), continuous variable
- DIST20M - distance to deep
- CHL_A - chlorophyll a
- TEMP09M - mean temperature in Aprile-June 1975-2005 at 0-9 meters depth
- SALT09M - mean salinity in Aprile-June 1975-2005 at 0-9 meters depth

The dependent variable is WHISUM - the number of whitefishes caught in sampling occasion.

An "off-set" variable / covariate for likelihood VOLUME - the volume of water sampled We can notice that there are 7 observations where the variable volume has value 0, this is probably due to some error in collecing the data, and it causes problems in the model, as we would have troubles in computing the density, hence we discard such datapoints from the analysis.

```
# Set up data
s = as.matrix(cbind(whitefish.dat.cov$E_etrs89, whitefish.dat.cov$N_etrs89)) / 1000   # spatial coordin
x = matrix(0,nrow=nrow(s),ncol=14)      # intercept + 5 BOTTOMCLS classes + 8 continues covariates
x[,1] = 1                              # Set the column corresponding to intercept to 1
x[whitefish.dat.cov$BOTTOMCLS==1,2] = 1   # Set the elements corresponding to BOTTOMCLS = 1 to 1
x[whitefish.dat.cov$BOTTOMCLS==2,3] = 1   # Set the elements corresponding to BOTTOMCLS = 2 to 1
```

```
x[whitefish.dat.cov$BOTTOMCLS==3,4] = 1   # Set the elements corresponding to BOTTOMCLS = 3 to 1
x[whitefish.dat.cov$BOTTOMCLS==4,5] = 1   # Set the elements corresponding to BOTTOMCLS = 4 to 1
x[whitefish.dat.cov$BOTTOMCLS==5,6] = 1   # Set the elements corresponding to BOTTOMCLS = 5 to 1
xcont = as.matrix(cbind(whitefish.dat.cov$DIS_SAND,
                        whitefish.dat.cov$FE300ME,
                        whitefish.dat.cov$ICELAST09,
                        whitefish.dat.cov$RIVERS,
                        whitefish.dat.cov$DIST20M,
                        whitefish.dat.cov$CHL_A,
                        whitefish.dat.cov$TEMP09M,
                        whitefish.dat.cov$SALT09M))
stdxcont = apply(xcont, 2, sd)
mxcont = apply(xcont, 2, mean)
x[,7:14] = t( apply( t(apply(xcont,1,'-',mxcont)),1,'/',stdxcont) )    # "standardize" the continuous c

# End variable
y = whitefish.dat$WHISUM               # number of counted fish larvae
par(mfrow=c(1,2))
hist(y, main ="number of larvae")
# Sampling effort; that is the volume of water sampled
V = whitefish.dat$VOLUME
hist(y/V, main ="number of larvae per volume")
```




```
# 0 volume observations
vol0 = which(V==0)
cat(paste(c("Zero volume observations:", vol0)))
```

```
## Zero volume observations: 618 619 620 621 622 623 625
```

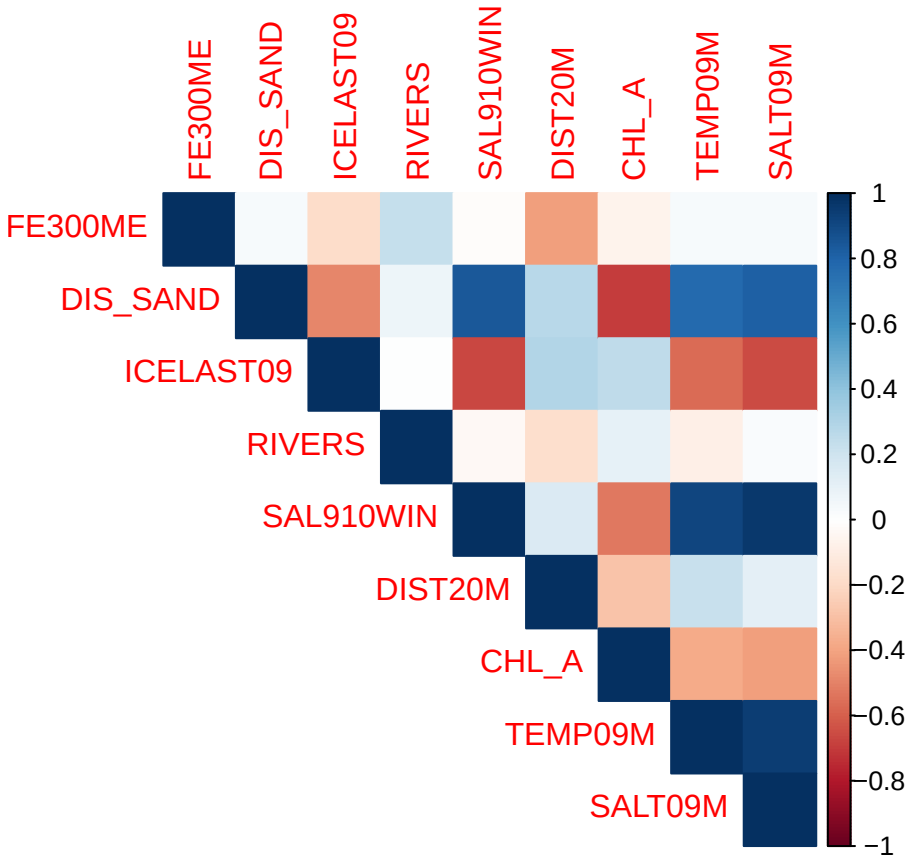
```
# remove the observations with 0 volume
y=y[-vol0]
x=x[-vol0,]
s=s[-vol0,]
V=V[-vol0]

# Prediction variables
spred = as.matrix(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89)) / 1000  # spatial coordin
xpred = matrix(0,nrow=nrow(spred),ncol=14)      # intercept + 6 BOTTOMCLS classes + 5 continues covaria
xpred[,1] = 1                                 # Set the column corresponding to intercept to 1
xpred[whitefish.raster$BOTTOMCLS==1,2] = 1   # Set the elements corresponding to BOTTOMCLS = 1 to 1
xpred[whitefish.raster$BOTTOMCLS==2,3] = 1   # Set the elements corresponding to BOTTOMCLS = 2 to 1
xpred[whitefish.raster$BOTTOMCLS==3,4] = 1   # Set the elements corresponding to BOTTOMCLS = 3 to 1
xpred[whitefish.raster$BOTTOMCLS==4,5] = 1   # Set the elements corresponding to BOTTOMCLS = 4 to 1
xpred[whitefish.raster$BOTTOMCLS==5,6] = 1   # Set the elements corresponding to BOTTOMCLS = 5 to 1
xpredcont = as.matrix(cbind(whitefish.raster$DIS_SAND,
                            whitefish.raster$FE300ME,
                            whitefish.raster$ICELAST09,
                            whitefish.raster$RIVERS,
                            whitefish.raster$DIST20M,
                            whitefish.raster$CHL_A,
                            whitefish.raster$TEMP09M,
                            whitefish.raster$SALT09M))
xpred[,7:14] = t( apply( t(apply(xpredcont,1,'-',mxcont)),1,'/',stdxcont) )    # "standardize" the cont
```

Look at the correlation among the continuous covariates. We are excluding the variable SAL910WIN as we have salinity data from SeaSmart dataset, which contains more accurate information about salinity and is highly correlated with SAL91WIN. Salinity and temperature are also really strong correlated, nevertheless we are keeping both of them since it has been shown they are really relevant envionmental variables for studying climate change effect on species distribution.

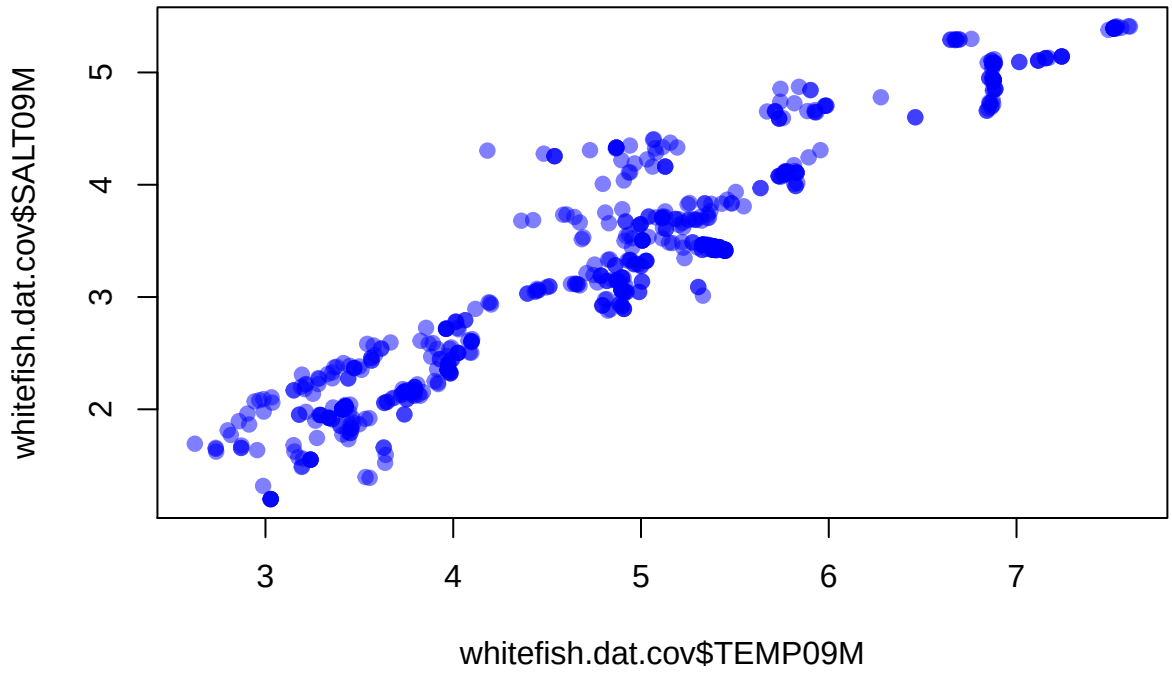
```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

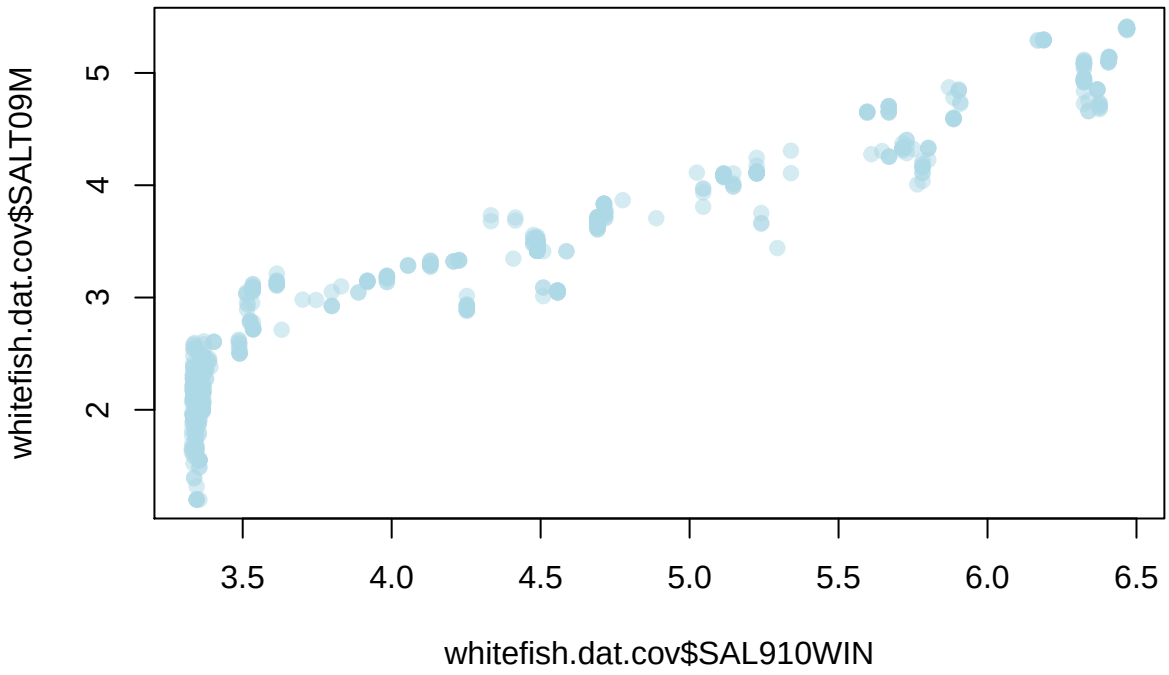
```
corr = cor(whitefish.dat.cov[, -c(1,2,4)])
corrplot(corr, type =  "upper", method="color" )
```




```
plot(whitefish.dat.cov$TEMP09M, whitefish.dat.cov$SALT09M, pch=19, col=alpha("blue", 0.5))
```




```
plot(whitefish.dat.cov$SAL910WIN, whitefish.dat.cov$SALT09M, pch=19, col=alpha("lightblue", 0.5))
```



Hierarchical specie distribution model

Hierarchical specie distribution model follows the generic hierarchical structure as presented by Wikle (2003), Cressie and Wikle (2011) and Banerjee et al. (2015).

$$[data|process, parameters] : \pi_Y(y(x,s)|f(x,s),\eta),$$

$$[process|parameters] : \pi_f(f(x,s)|\theta),$$

$$[parameters] : \pi(\eta,\theta),$$

where we have three hierarchical layers. The first layer is the observation process which describes the conditional distribution of observations $y = [y_1, \ldots, y_n]^T$ given the latent process $f(s)$ and observation model parameters, η. The data $y(x,s)$, are evaluated at spatial location $s \in D \in \mathbb{R}^2$ with associated covariates $x \in \mathbb{R}^d$. The second layer specifies the model for the latent process conditionally to the process model parameters θ, and the third layer specifies the prior for all the unknown parameters , also called hyperparameters.

Such models can be extended to Hierarchical multivariate species distribution model, by considering $j \in 1 \ldots J$ different species.

The variable of interest y represents the number of whitefishes caught in sampling locations. Let $i \in 1 \ldots N$ represent the sampling sites and j the number of different species considered (in this initial case we have $J = 1$), we assume the response variable has distribution

$$y_{ij}|\alpha_j, \beta_j, \phi_j \sim Poisson(V_i e^{f_j(x_i,s_i)})$$

where $e^{f_j(x_i,s_i)})$ models the larval density in the water, V_i is the sampled volume of water, and serves as an offset. We first assume the latent variable follows the linear model

$$f_j(x_i, s_i) = x_i^T \beta_j + \phi_j(s_i)$$

where $x_i \in \mathbb{R}^{13}$ contains the environmental covariates at location $s_i \in \mathbb{R}^2$, and has all 1 in the first column. Such models can be seen as an extention of Generalized liear models to (spatial) random effect. Given

$\mathbb{E}(y_i) = \mu_i \ \forall i \in 1, \ldots, N$ observations, the model can be written as:

$$g(\frac{\mu_i}{V_i}) = x_i^T \beta_j + \phi_j(s_i)$$

where $g(\cdot) = \log(\cdot)$ is the link function.

Note that, in our data, the first covariate is categorical with six different classes, the linear predictor function in the model can be expressed as

$$m(x) = \alpha_0 + \alpha_1 \delta_1(x_1) + \cdots + \alpha_5 \delta_5(x_1) + x_2\beta_1 + \cdots + x_6\beta_5$$

where $\delta_d(x_1) = 1$ if $x_1 = d \in 1, \ldots, 5$ and zero otherwise (the intercept corresponds to class 0), we will denote the 13 dimensional vector β_j. For the linear coefficients we assume uninformative priors $\beta_{jd} \sim N(0, 10) \ \forall d \in 1, \ldots, 13$. The variable $\phi(s)$ models the spatial random effects that describes temporally constant associations, unexplained by the available covariates.

We construct and analyse different Gaussian latent variable models (GLVM). These models are such that the process $f_j(x, s)$ conditional on hyperparameters, is assumed to be Gaussian random variable or Gaussian stochastic process.

```
## Test with smaller data first
n = nrow(whitefish.dat)-length(vol0)
m = 217 #smaller dataset
x = x[seq(1,n,length=m),]
s = s[seq(1,n,length=m),]
y = y[seq(1,n,length=m)]
V = V[seq(1,n,length=m)]
```

We first implement a model where $\phi \sim N(0, \sigma_\phi^2)$, i.e. a model where there is no spatial correlation induced by the random effect:

```
whitefishmodel_no_sre = "
  data {
    int<lower=1> N;
    int<lower=1> Dx;
    matrix[N,Dx] x;
    int<lower=0> y[N];
    vector[N] V;
  }

  parameters {
    vector[Dx] b;
    vector[N] phi;
    real<lower =0> s2_phi;
  }
  transformed parameters {
   vector[N] f;

    for (n in 1:N)
      f[n] =   dot_product(  x[n,], b) + phi[n]; //uncorrelated locations -> spatial random effect
  }
  model {
    s2_phi ~ student_t(4, 0, 1);
    for (i in 1:Dx)
      b[i] ~ normal(0, sqrt(10));

    for (n in 1:N) {
```

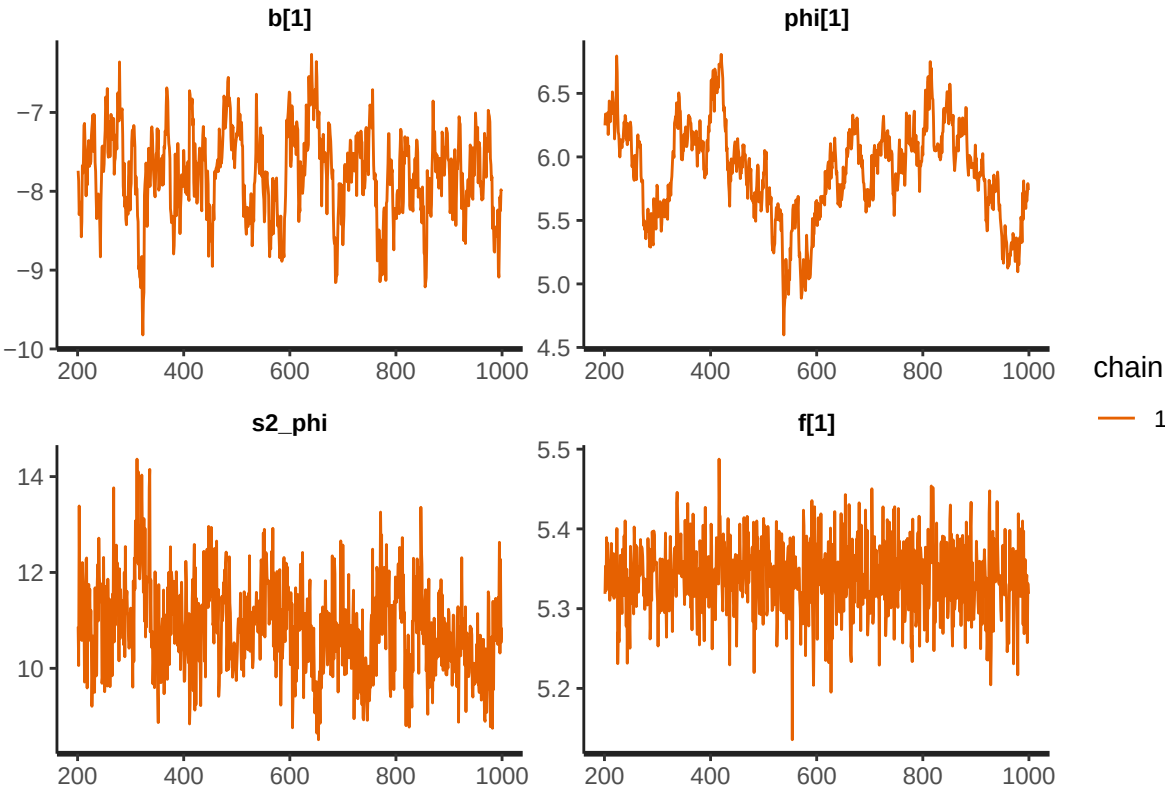
```
      phi[n] ~ normal(0,sqrt(s2_phi));
      y[n] ~ poisson(V[n]*exp(f[n]));
    }


  }
"
```

Check the model

```
whitefish_data <- list(Dx = ncol(x),
                       N = nrow(x),
                       x = x,
                       y = y,
                       V = V)

# Compile the STAN model
# fit00 = stan(model_code =  whitefishmodel_no_sre, data = whitefish_data, warmup=200,
#              iter = 1000, chains = 1,  control = list(adapt_delta = 0.99) )
# saveRDS(fit00, file = "fit_wf_nornd.rds")
fit00 <- readRDS("fit_wf_nornd.rds")


m00 = as.matrix(fit00)
#print(fit00)
summary(fit00, pars = c( "b[1]","phi[1]","s2_phi", "f[1]") )            # summary
```

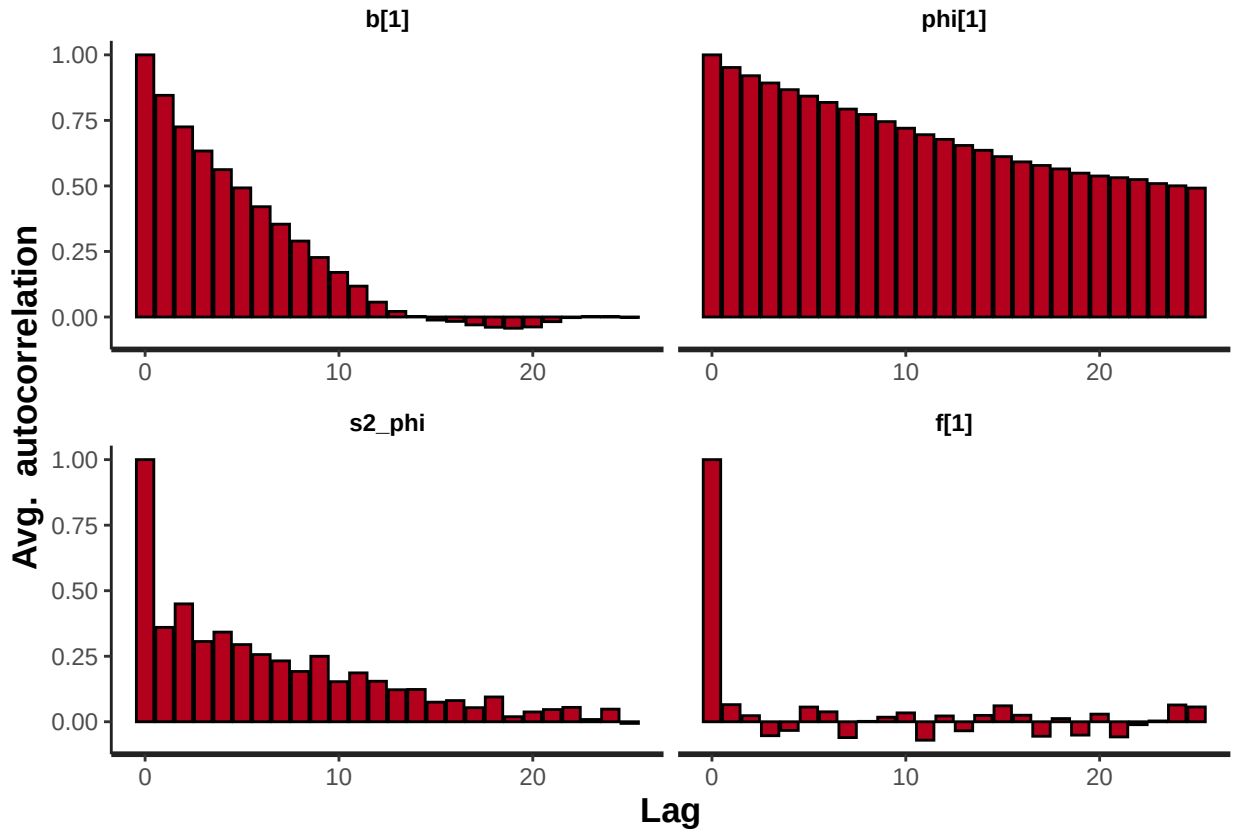
```
## $summary
##             mean     se_mean         sd      2.5%       25%       50%       75%
## b[1]   -7.773728 0.064707839 0.55624019 -8.991201 -8.151147 -7.744109 -7.384155
## phi[1]  5.881192 0.104913491 0.37895414  5.082065  5.642012  5.927722  6.136764
## s2_phi 10.843056 0.103733059 0.96877840  9.079733 10.125186 10.794038 11.504357
## f[1]    5.339292 0.001716608 0.04572697  5.247346  5.310588  5.341533  5.369192
##            97.5%     n_eff      Rhat
## b[1]   -6.774486  73.89430 0.9990517
## phi[1]  6.602588  13.04700 1.0047114
## s2_phi 12.822014  87.21969 1.0716789
## f[1]    5.424651 709.58173 1.0005730
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter      mean         sd      2.5%       25%       50%       75%
##    b[1]   -7.773728 0.55624019 -8.991201 -8.151147 -7.744109 -7.384155
##    phi[1]  5.881192 0.37895414  5.082065  5.642012  5.927722  6.136764
##    s2_phi 10.843056 0.96877840  9.079733 10.125186 10.794038 11.504357
##    f[1]    5.339292 0.04572697  5.247346  5.310588  5.341533  5.369192
##          stats
## parameter     97.5%
##    b[1]   -6.774486
##    phi[1]  6.602588
##    s2_phi 12.822014
##    f[1]    5.424651
```

```
stan_trace(fit00, pars = c( "b[1]","phi[1]","s2_phi", "f[1]") )          # traceplot
```

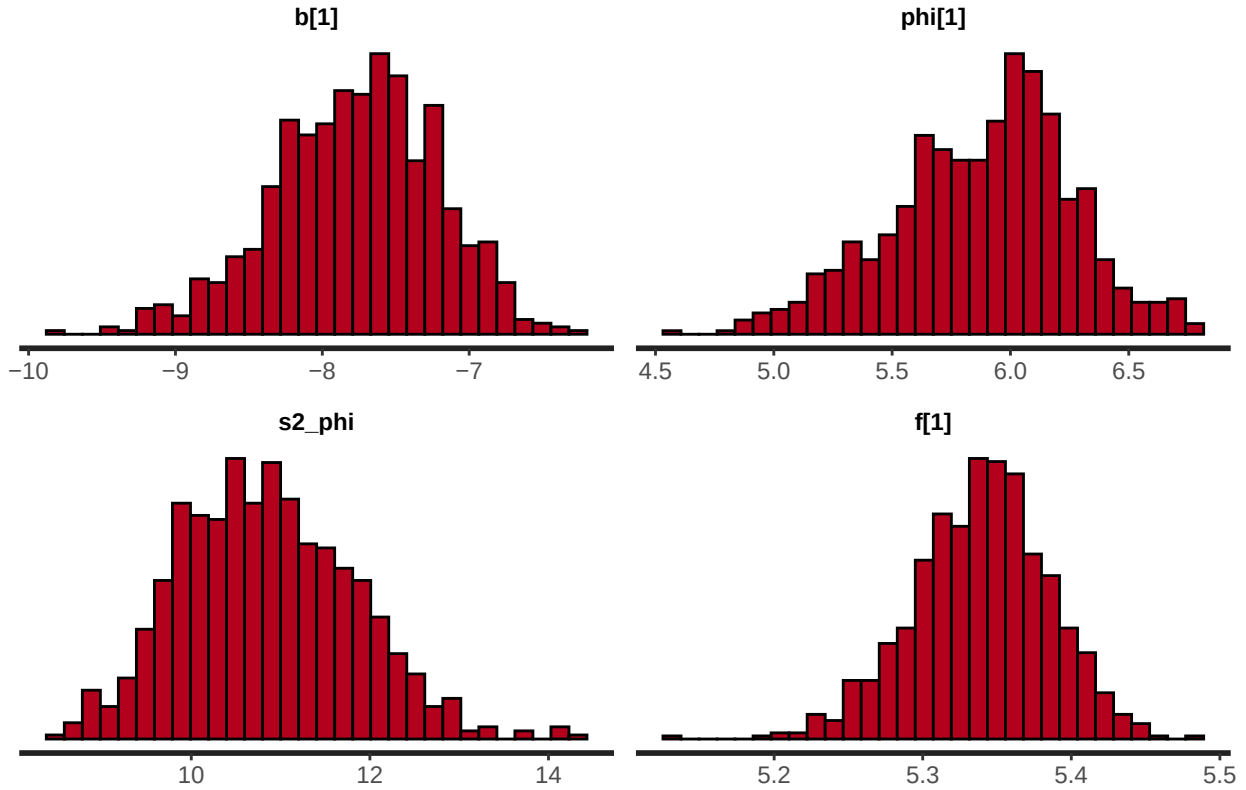



```
stan_ac(fit00,inc_warmup = FALSE, lags = 25, pars = c( "b[1]","phi[1]","s2_phi", "f[1]") )    # autocorr
```

```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit00, pars = c( "b[1]","phi[1]","s2_phi", "f[1]"))  )
```



Now we implement a linear mixed model (or linear random effects model). We model ϕ by using a zero mean

Gaussian Process

$$\phi_j(s)|\sigma^2_\phi, l \sim N(0, \Sigma_\phi)$$

This is a GLVM since each additive component is Gaussian, which implies that the marginal distribution for any $f = [f(s_1), \ldots, f(s_n)]$ is again Gaussian:

$$f|S, X(S) \sim N(0, X(S)\Sigma_\beta X(S)^T + \Sigma_\phi)$$

where $X(S)^T = [x(s_1), \ldots, x(s_n)]$, $\Sigma_\phi = Cov(\phi, \phi)$ and $\phi = [\phi(s_1), \ldots, \phi(s_n)]^T$.
In our case $\Sigma_\beta = \sigma^2_\beta \mathbf{I}$. Denoted by $\Sigma_\beta(s, s') = x(s)\Sigma_\beta x(s')^T$, we have that

$$f|S, X \sim GP(0, \Sigma_\beta(s, s') + \Sigma_\phi(s, s'))$$

or equivalently

$$f \sim GP(x(s)^T\beta, \Sigma_\phi(s, s'))$$

The covariance matrix of the random effect ϕ is such that each element follows a squared exponential covariance function

$$\Sigma_\phi(s_i, s_h) = \sigma^2_\phi \exp\Big(- \sum_{k=1}^{2} \frac{|s_{i,k} - s_{h,k}|^2}{l^2_k} \Big)$$

where we assume the covariance parameters priors:

$$\sigma^2_\phi \sim Student - t_{\nu=4}(\mu = 0, \sigma = 1)$$

$$l \sim Student - t_{\nu=4}(\mu = 0, \sigma = 32)$$

We set an uninformative prior for the variance parameter σ, while, for the length-scale, l, which governs how fast the correlation decreases as a function of distance, we select a half-Student-t prior with scale parameter $\sigma_l = 32$ so that it gives 90% prior probability correlation ranges below 50km, where the correlation range, measured by l, indicates the distance at which the covariance function has dropped to 5 % of its maximum. We set such value as we want to prefer solutions where the spatial correlation in the spatial random effect shrinks to about zero between different sampling sites, that we assume being at least 50km far apart.

Note that, for computational reasons, we define the model so that we sample from the posterior of $z = L^{-1}f$, where L is a matrix that approximates a square root of the posterior covariance of f.

```
GP_whitefishmodel = "
  data {
    int<lower=1> N;
    int<lower=1> Dx;
    matrix[N,Dx] x;
    int<lower=1> Ds;
    matrix[N,Ds] s;
    int<lower=0> y[N];
    vector[N] V;
  }
  transformed data {
    vector[N] mu;
    matrix[N, N] Dist_spatial;
    matrix[N, N] Sigma_lin;
    real s2_lin;
    s2_lin = 10;
    for (i in 1:N)
      mu[i] = 0;
    // off-diagonal elements
```

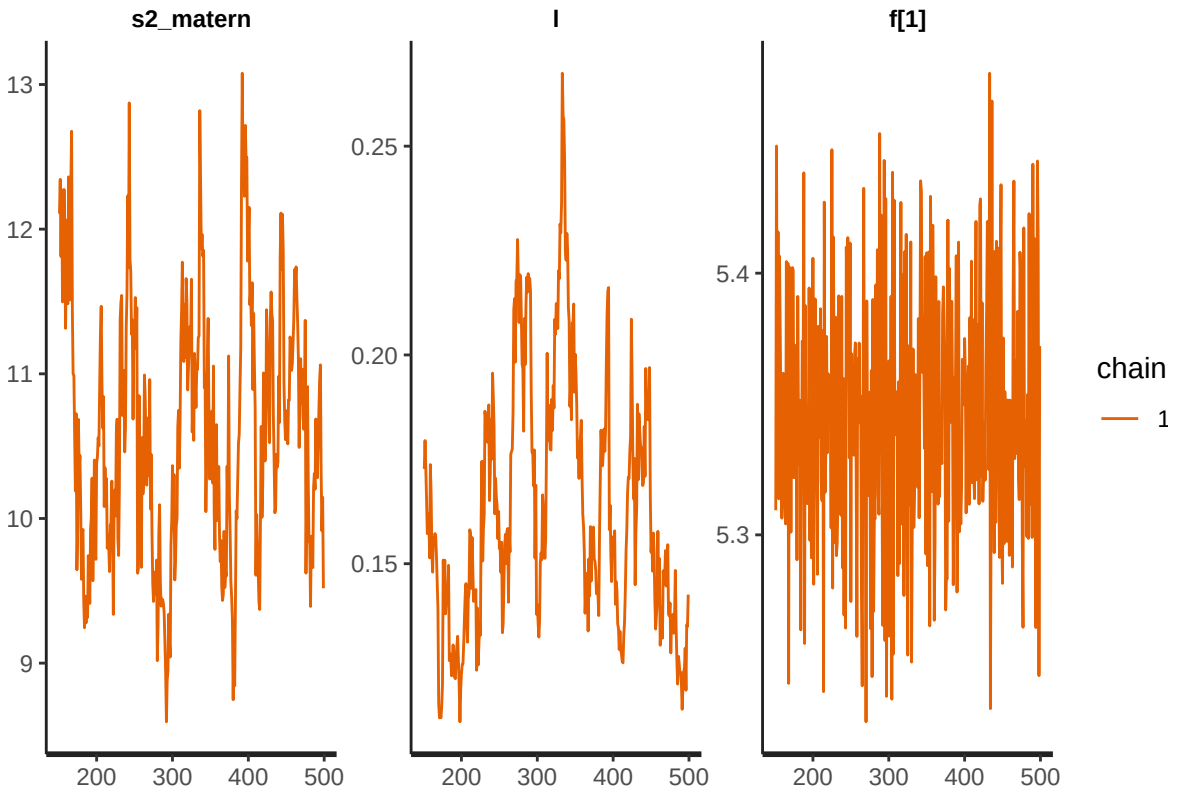
```
    for (i in 1:(N-1)) {
      for (j in (i+1):N) {
        Dist_spatial[i, j] = pow(dot_self(s[i] - s[j]),0.5)  ;
        Sigma_lin[i, j] = s2_lin * dot_product(x[i],x[j]);     // linear covariance function

        // Fill in the other half
        Dist_spatial[j, i] = Dist_spatial[i, j];
        Sigma_lin[j, i] = Sigma_lin[i, j];
      }
    }
    // diagonal elements
    for (k in 1:N){
      Dist_spatial[k, k] = 0;
      Sigma_lin[k, k] = s2_lin * dot_product(x[k],x[k]) + 1e-6;   // add also some jitter
    }
  }
  parameters {
    real<lower=0> l;
    real<lower=0> s2_matern;
    vector[N] z;
  }
  transformed parameters {
    matrix[N, N] Sigma;
    matrix[N, N] L;
    real<lower=0> inv_l;

    inv_l = inv(l);
   //      Sigma = s2_matern*exp(-inv_l*Dist_spatial ) + Sigma_lin; // Exponential
 Sigma = s2_matern*(1 + pow(3,0.5)*inv_l*Dist_spatial).*exp(-pow(3,0.5)*inv_l*Dist_spatial)  + Sigma_lin
    L = cholesky_decompose(Sigma);
  }
model {
    vector[N] ff;

    // A weakly informative prior for magnitude
    s2_matern ~ student_t(4, 0, 1);

    // A weakly informative prior for l, that shrinks to 0 cor. among locations with more than 50km dist
    l ~ gamma(7,0.1);

    z ~ normal(0, 1);
    ff = L*z;

    for (n in 1:N)
      y[n] ~ poisson(V[n]*exp(ff[n]));

  }
  generated quantities {
    vector[N] f;
    // derived quantity (transform)
    f = L*z;
  }"
```

Check the model

```r
whitefish_dat <- list(Dx = ncol(x),
                      N = nrow(x),
                      Ds = ncol(s),
                      x = x,
                      s = s,
                      y = y,
                      V = V)

# Compile the STAN model
# fit0 = stan(model_code =  GP_whitefishmodel, data = whitefish_dat, warmup=150,
#             init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2=1)),
#             iter = 500, chains = 1 , control = list(adapt_delta = 0.99), pars=c("f", "s2", "l")  )
# saveRDS(fit0, file = "fit_wf_exp_gamma.rds")
fit0 <- readRDS("fit_wf_exp_gamma.rds")

m0 = as.matrix(fit0)
#print(fit0)              # Rhat
summary(fit0, pars = c("s2_matern", "l", "f[1]"))            # summary
```

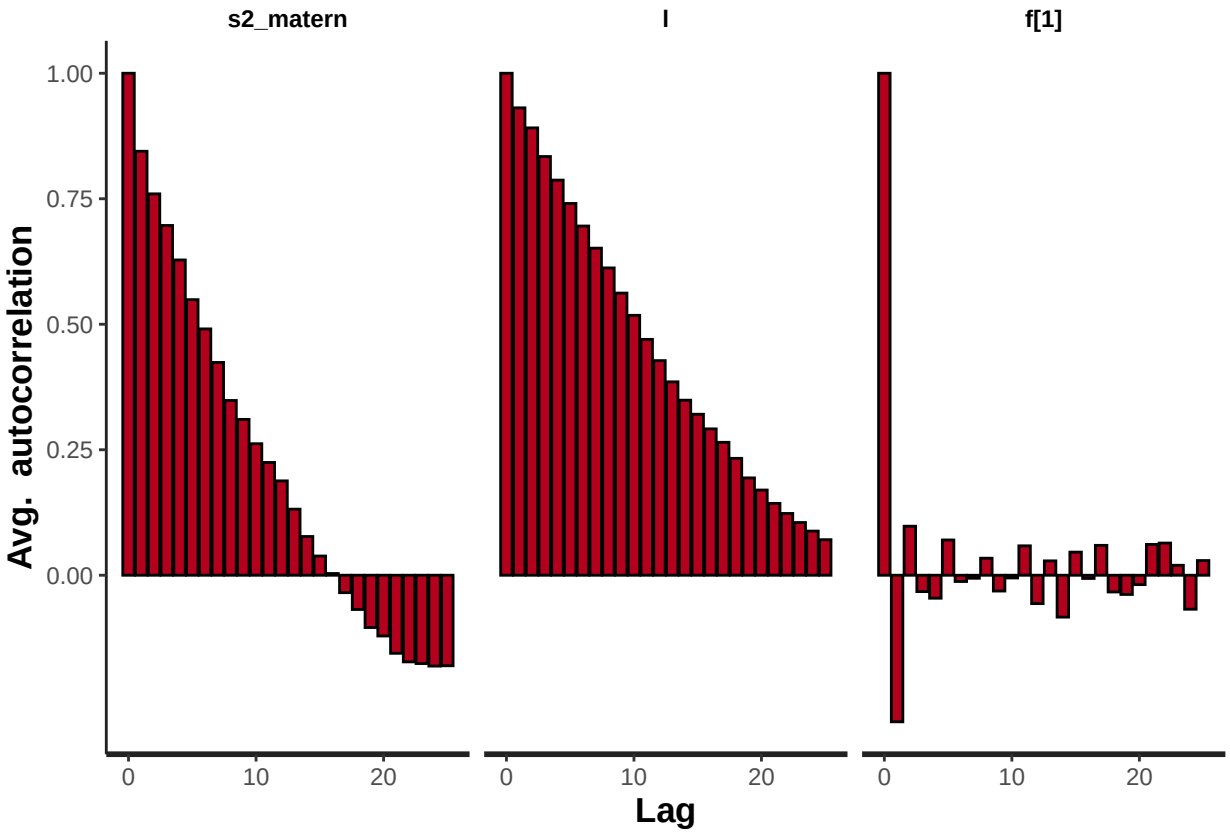
```
## $summary
##                 mean     se_mean         sd      2.5%      25%        50%
## s2_matern 10.5989222 0.169190289 0.87631210 9.1505766 9.923650 10.5570815
## l          0.1628884 0.008166456 0.03018032 0.1203038 0.140229  0.1565145
## f[1]       5.3462508 0.001998981 0.04999862 5.2499704 5.311497  5.3461249
##                 75%      97.5%     n_eff      Rhat
## s2_matern 11.2205480 12.3730900  26.82667 1.0140815
## l          0.1819574  0.2258724  13.65779 0.9993014
## f[1]       5.3839370  5.4384023 625.60273 1.0033989
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter         mean         sd      2.5%      25%        50%        75%
##   s2_matern 10.5989222 0.87631210 9.1505766 9.923650 10.5570815 11.2205480
##   l          0.1628884 0.03018032 0.1203038 0.140229  0.1565145  0.1819574
##   f[1]       5.3462508 0.04999862 5.2499704 5.311497  5.3461249  5.3839370
##            stats
## parameter        97.5%
##   s2_matern 12.3730900
##   l          0.2258724
##   f[1]       5.4384023
```

```r
stan_trace(fit0 , pars = c("s2_matern", "l", "f[1]"))         # traceplot
```

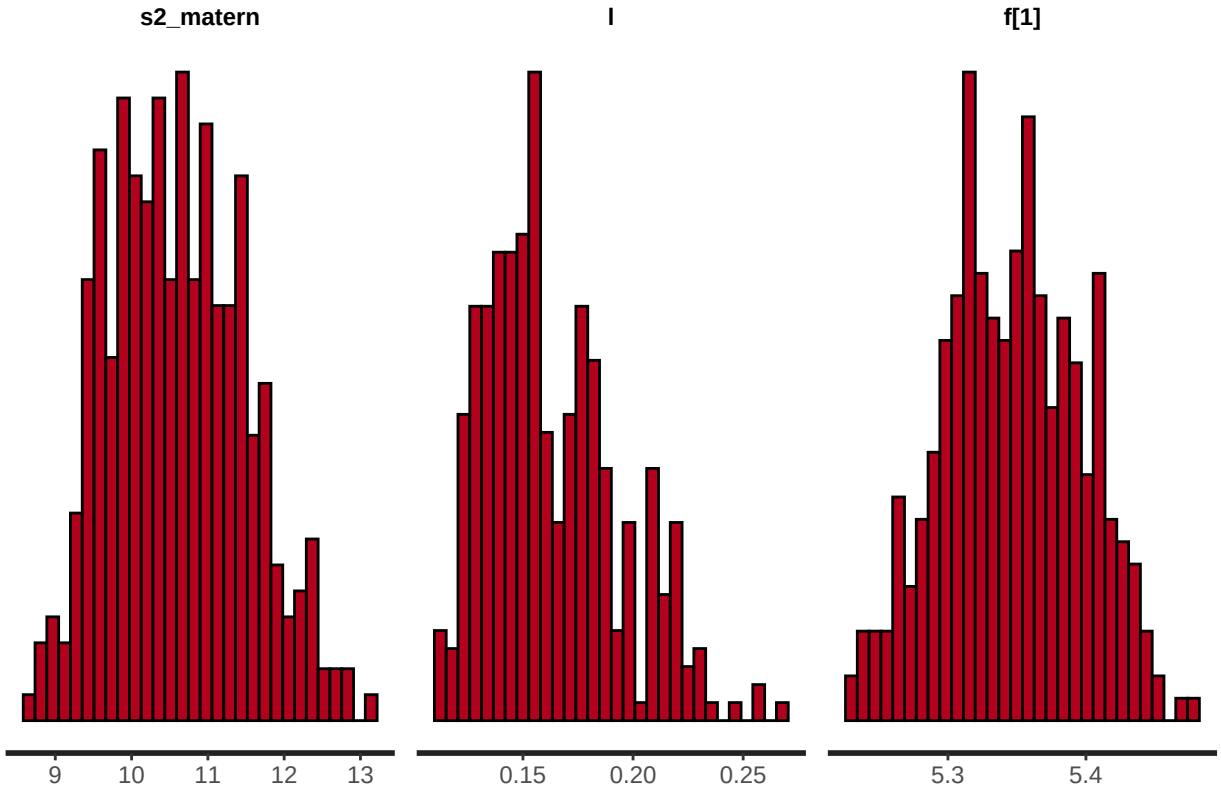



```
stan_ac(fit0,inc_warmup = FALSE, lags = 25 , pars = c("s2_matern", "l", "f[1]"))    # autocorrelation be
```

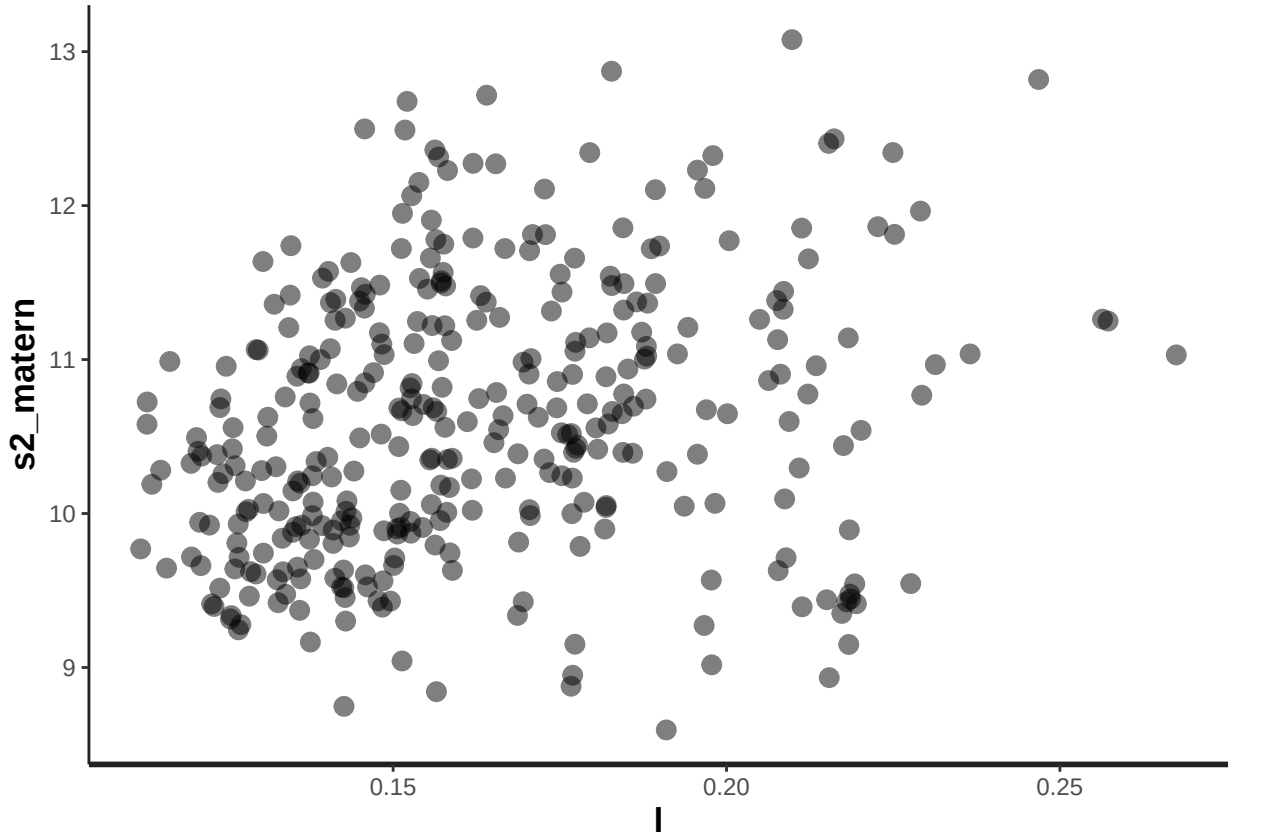
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit0, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```r
# scatter plot of parameters of spatial random effect
stan_scat(fit0, pars = c("l", "s2_matern"), color = "black", size = 3)
```



Now, instead of an exponential covariance function we chose the Matern covariance function with parameter $\nu = \frac{3}{2}$ for ϕ_j. This gives

$$k_{mater}(s_i, s_h) = \sigma^2(1 + \sqrt{3}r(s_i, s_h))e^{-\sqrt{3}r(s_i,s_h)}$$

where $\sigma^2 = 10$ and

$$r(s_i, s_h) = \sqrt{\sum_{k=1}^{2} \frac{(s_{k,i} - s_{k,h})^2}{l_k^2}}$$

This is equivalent to model directly the latent variable with a Gaussian Processes, so that

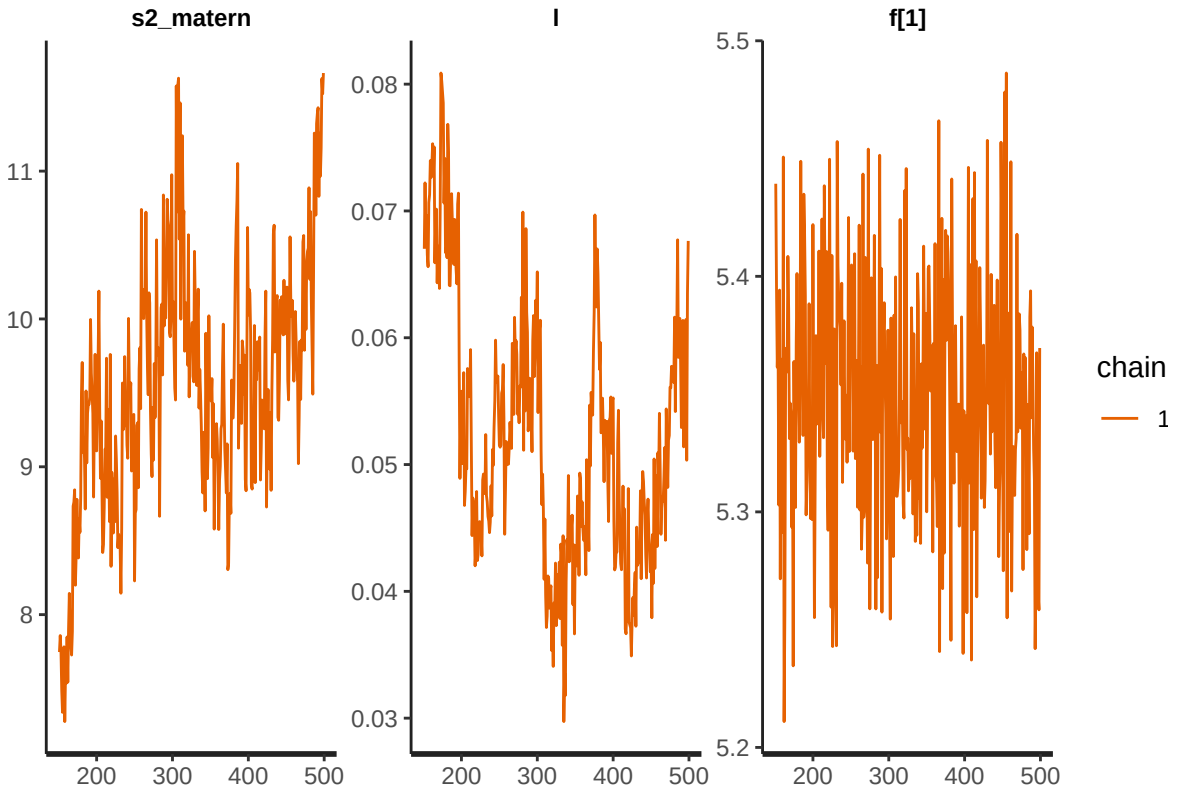
$$f_j(s, x) \sim GP(m(x), k_{matern}(s, s'|l, \sigma^2))$$

The covariance parameters priors are as above.

```r
# fit = stan(model_code =  GP_whitefishmodel3, data = whitefish_dat, warmup=150, iter = 500, chains = 1
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
#saveRDS(fit, file = "fit_wf_mater3_gamma.rds")
fit <- readRDS("fit_wf_mater3_gamma.rds")


m = as.matrix(fit)
#print(fit)              # Rhat
summary(fit, pars = c("s2_matern", "l", "f[1]"))            # summary
```

```
## $summary
##                  mean     se_mean          sd       2.5%        25%        50%
## s2_matern 9.54817331 0.234168627 0.82647081 7.75197887 9.05555982 9.53081844
## l         0.05224353 0.003125708 0.01018748 0.03692284 0.04448884 0.05112107
## f[1]      5.34867866 0.002574157 0.05092654 5.25200388 5.31462260 5.34577102
##                  75%       97.5%     n_eff      Rhat
## s2_matern 10.01371756 11.34004210  12.45655 1.0635749
## l          0.05883251  0.07401797  10.62275 1.1602087
## f[1]       5.38176847  5.45084612 391.39760 0.9976878
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter         mean         sd       2.5%        25%        50%         75%
##   s2_matern 9.54817331 0.82647081 7.75197887 9.05555982 9.53081844 10.01371756
##   l         0.05224353 0.01018748 0.03692284 0.04448884 0.05112107  0.05883251
##   f[1]      5.34867866 0.05092654 5.25200388 5.31462260 5.34577102  5.38176847
##            stats
## parameter         97.5%
##   s2_matern 11.34004210
##   l          0.07401797
##   f[1]       5.45084612
```

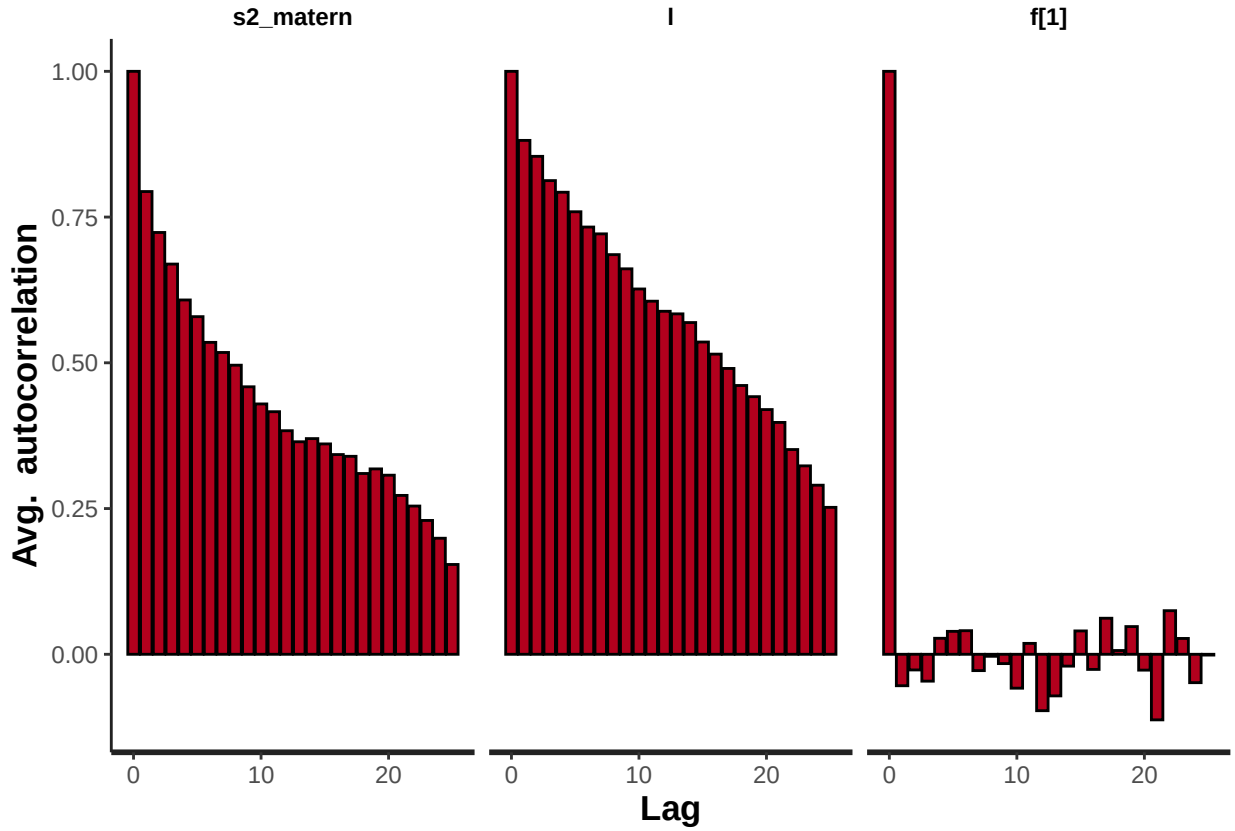
```
stan_trace(fit, pars = c("s2_matern", "l", "f[1]"))         # traceplot
```



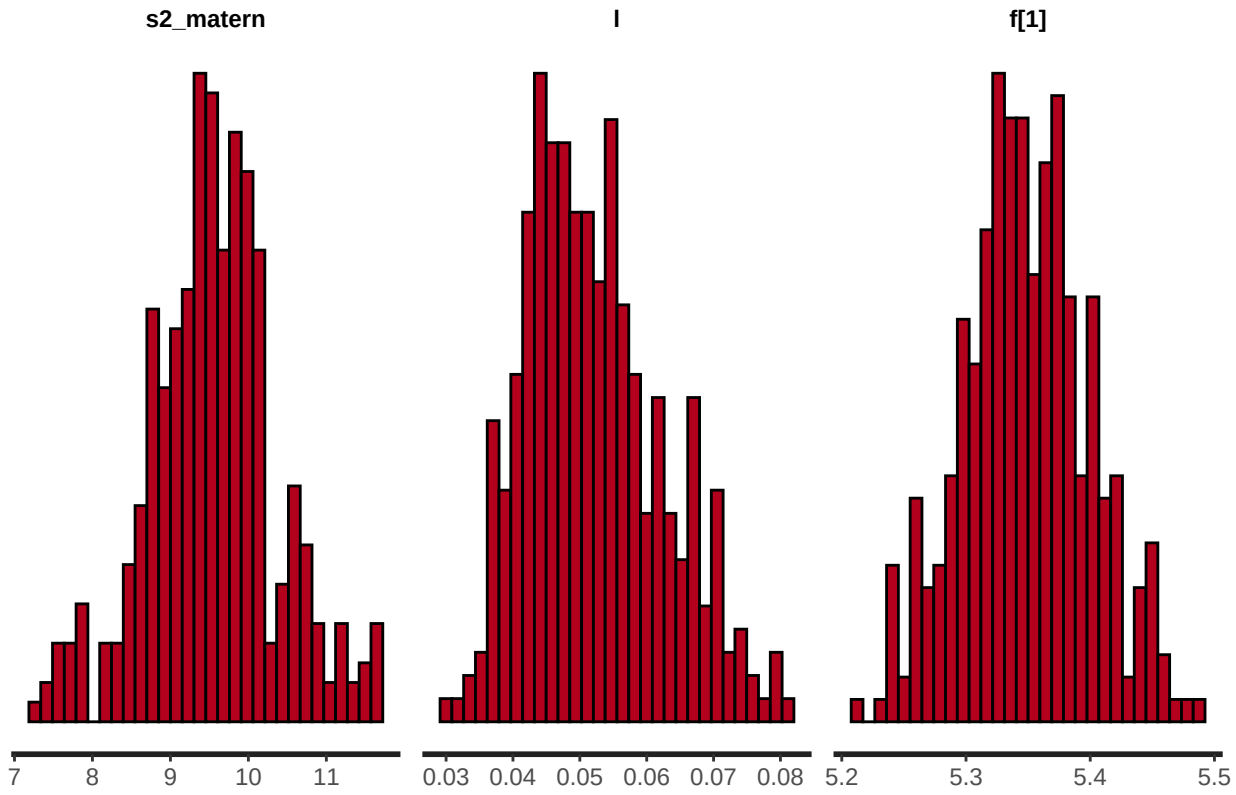

```
stan_ac(fit,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation betw
```

```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```

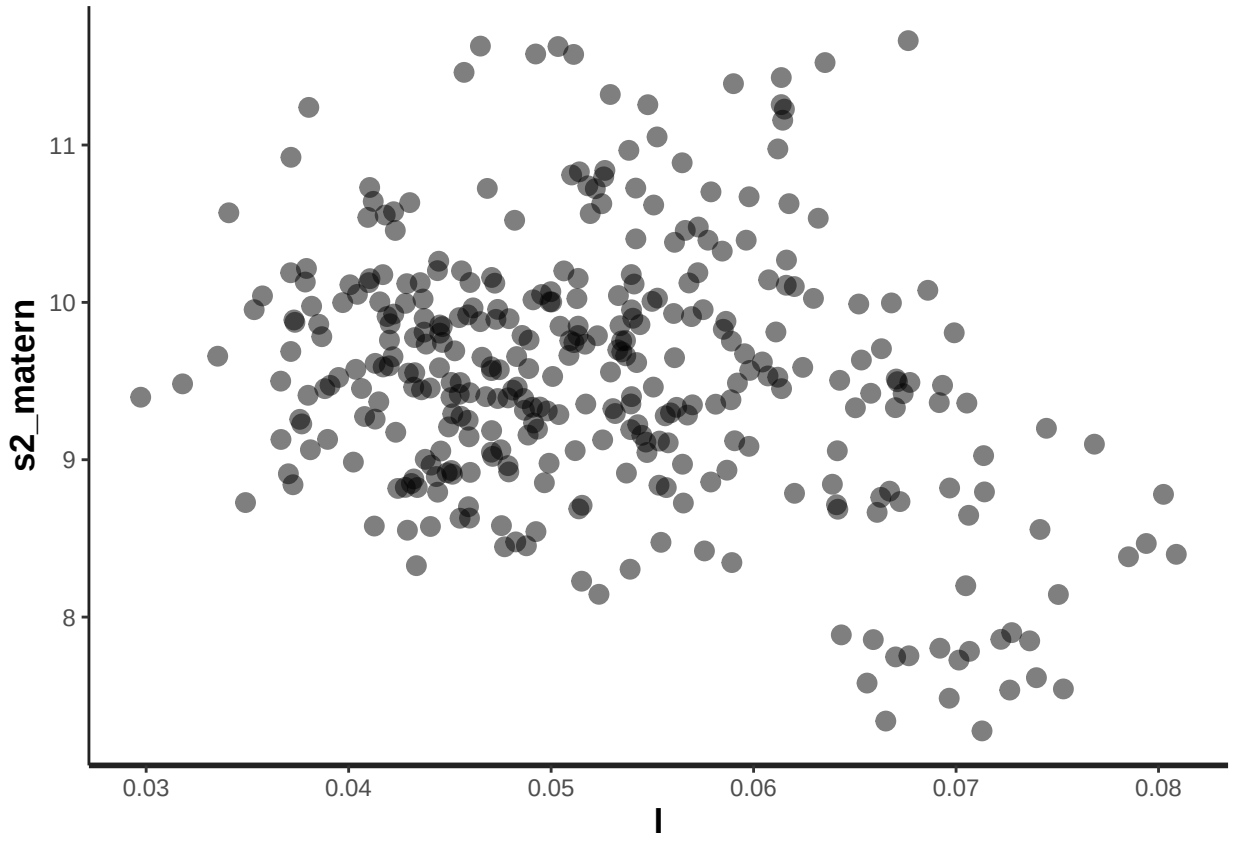
```
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```




```
# scatter plot of parameters of spatial random effect
stan_scat(fit, pars = c("l", "s2_matern"), color = "black", size = 3)
```



In both models we have a really high autocorrelation. The exponential covariance model has clear problems with convergence, in the Matern covariance model we still have some convergence problems for a few parameters estimating the latent function f_i. We select the latter to do our predictions on the whole Gulf of Bothnia.

Posteror distributions and MCMC

Given the hierarchical model described previously, our inferential objective is the posterior distributions of the hyperparameters and the latent function, as well as the predictive distribution for new observations. These posterior probability distributions cannot be solved analytically in general.

Markov chain Monte Carlo (MCMC) is a technique (or morecorrectly, a family of techniques) for sampling probability distributions. Typical applications are in Bayesian modelling, the target distributions being posterior distributions of unknownparameters, or predictive distributions for unobserved phenomena. In order to estimate the posterior density of the parameters Stan uses a tailored Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 1996, 2011) where the tuning of the sampling parameters is done in automated manner (Hoffman and Gelman, 2014). Hamiltonian Monte Carlo utilizes the gradient information of the log posterior distribution to direct the sampling to interesting regions and, hence, to speed up the convergence and improve mixing. When using this method understanding how to tune the sampling parameters can be challenging. Even though MCMC methods are theoretically appealing and the Monte Carlo estimate is proved to converge to the correct distribution, they are often hard to implement in practice. Moreover, after the convergence the sample chain might mix poorly which results in high autocorrelation and low number of efficient samples. In order to reduce autocorrelation in our model sample, we thin the chains, that is, we discard all but every k-th observation.

Predictions

The posterior predictive density of latent variables, conditional on hyperparameters, is

$$\tilde{f}|S, X(S), y, \tilde{S}, \tilde{X}(\tilde{S}), \theta \sim N(K_{\tilde{f},f}(K_{f,\tilde{f}} + \sigma^2 I)^{-1} y, K_{f,\tilde{f}} - K_{f,\tilde{f}}(K_{f,\tilde{f}} f + \sigma^2 I)^{-1} K_{f,\tilde{f}})$$

where

$$K_{f,\tilde{f}} = \tilde{X}(\tilde{S})\Sigma_\beta X(S) + K_{\phi,\tilde{\phi}}$$

$$K_{\tilde{f},\tilde{f}} = \tilde{X}(\tilde{S})\Sigma_\beta = \tilde{X}(\tilde{S} + K_{\phi,\tilde{\phi}}$$

and

$$K_{f,f} = X(S)\Sigma_\beta X(S) + K_{\phi,\phi}.$$

The posterior distribution for linear weights, conditional on the known error covariance is then

$$\beta|y, X, \sigma^2, \Sigma_\phi \sim N(\mu_p, \Sigma_p)$$

where in the case of dependent errors,

$$\mu_p = \Sigma_\beta X^T (X\Sigma_\beta X^T + \sigma^2 I)^{-1} y$$

$$\Sigma_p = \Sigma_\beta - \Sigma_\beta X^T (X\Sigma_\beta X^T + \sigma^2 I)^{-1} X\Sigma_\beta$$

We calculate the posterior distribution for the linear predictor parameters: β_j and report the posterior mean and 95% credible interval for each of them, in order to evaluate the effect of each environmental variable on white fish larvae density. We calculate the posterior predictive mean and variance of the log density: $f(x, s)$ as well as the posterior median of the density of larvae throughout the study region, using the prediction raster dataset predrasterwhitefish.txt .

Note: We can simulate from a Gaussian process with mean function $\mu(s)$ and covariance function $k(s, s_0)$ at locations $S = [s_1, \ldots, s_n]^T$ as follows. Construt a vector $\mu = [\mu(s_1), \ldots, \mu(s_n)]^T$ and a covariance matrix $[K_{f,f}]_{ii,j} = k(s_i, s_j)$. Form a Cholesky decomposition of the covariance matrix LL^T . Form an $n \times 1$ vector of i.i.d. zero mean and unit variance Gaussian random variables, $z \sim N(0, I)$. After this form a vector $f = \mu + Lz$. The vector f is then a sample from the Gaussian process at locations S.

```
prediction= function(m, Covariance, x, xpred , s, spred, thin=8) {
  m_thinned = as.matrix(m[seq(1,nrow(m),thin),])

  EfMCMC = matrix(nrow=nrow(xpred),ncol=nrow(m_thinned)) # E(f) in each location, for each MCMC iter(th
  sampf_linMCMC = matrix(nrow=ncol(x),ncol=nrow(m_thinned)) # sample f: to get beta estimation for each
  VarfMCMC = matrix(nrow=nrow(xpred),ncol=nrow(m_thinned)) # Var(f) in each location, for each MCMC ite

  for (i1 in 1:nrow(m_thinned)){
    l = as.double(m_thinned[i1,"l"])
    sigma2 = as.double(m_thinned[i1,"s2_matern"])
    s2_lin = as.vector(rep(10,ncol(x)))   # Linear covariance function

    y = m_thinned[i1,3:(nrow(x)+2)]

    Kt = Covariance(s,s,l,sigma2, 1) + linearCovariance(x,x,s2_lin) + diag(rep(1e-6,nrow(x))) #add jiit
    Kpt = Covariance(spred, s, l,sigma2,0)
    Kpt_lin = linearCovariance(xpred,x,s2_lin) # xpred * diag(sigma2) * x^T
    Kpt_all = Kpt + Kpt_lin

    # Find the Cholesky Decomposition of the covariance matrix
    L = t( chol(Kt) )
    a = solve(t(L),solve(L,y)) #a =(L.t*L)-1 y   (z)

    # Plot the different components

    #  --------------- The full f(t) -------------
    # The posterior predictive mean
    EfMCMC[,i1] = Kpt_all%*%a
    LKtp = solve(L,t(Kpt_all))
    VarfMCMC[,i1] = matrix(sigma2 + xpred^2%*%s2_lin,nrow(Kpt_all),1) - as.matrix( colSums( LKtp*LKtp )

    # The posterior of the linear weights
    xpred22 = diag(ncol(x))
    Kpt_lin = linearCovariance(xpred22,x,s2_lin) # diag(s2)*X^T

    Ef_linMCMC = Kpt_lin%*%a # mean f tilde
    LKtp = solve(L,t(Kpt_lin))
    Covf =  xpred22%*%diag(s2_lin) - as.matrix(  t(LKtp)%*%LKtp ) # + diag(rep(1e-6,nrow(xpred22)))# co
    sampf_linMCMC[,i1] = Ef_linMCMC + chol(Covf)%*%rnorm(nrow(Covf)) # sample beta coeff E(ftilde) +Lz
    #Varf_linMCMC[,i1] = matrix(xpred22^2%*%s2_lin,nrow(Kpt_lin),1) - as.matrix( colSums( LKtp*LKtp ) )
  }

  # linear coefficients(weights): beta estimates
  # Posterior mean and standard deviation of fixed effects
    mean=rowMeans(sampf_linMCMC)
   sd=rowSds(sampf_linMCMC)
  # 95% quantiles for fixed effects
  q=apply(sampf_linMCMC, 1, quantile, probs = c(0.025, 0.975))
```

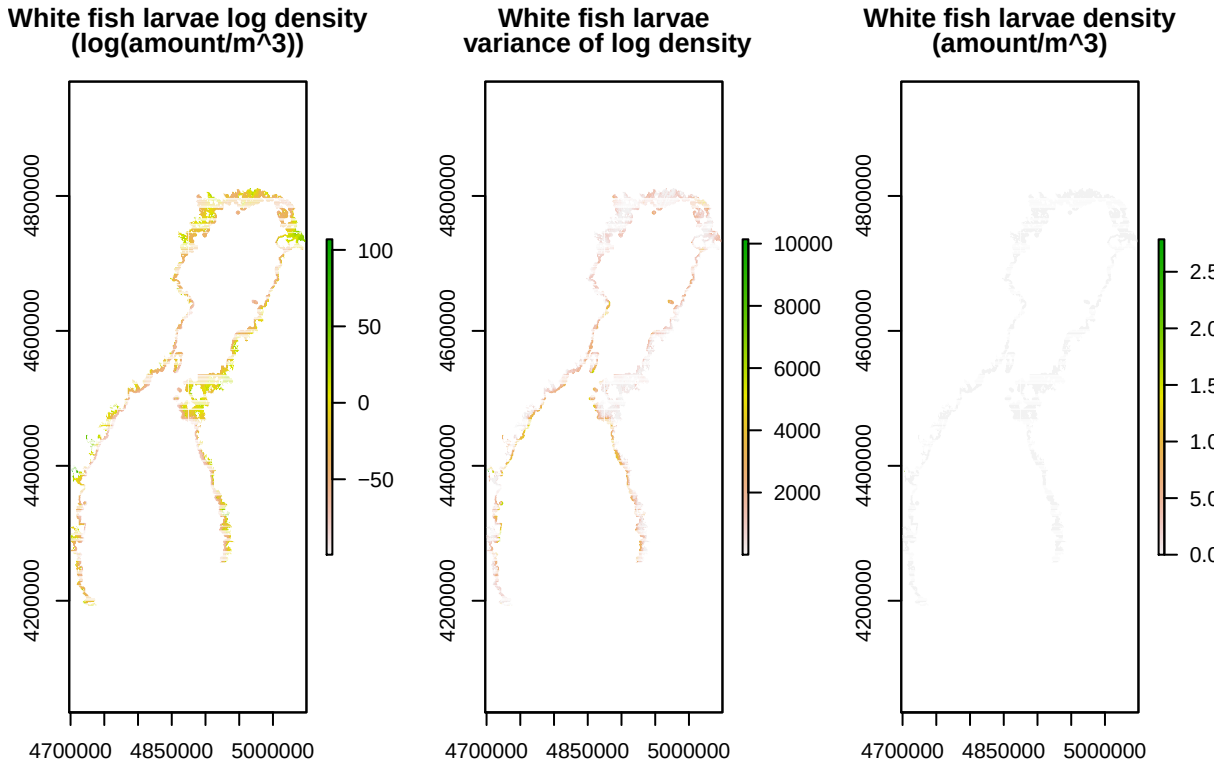
```
  summary_beta=data.frame(mean,sd,t(q))
  colnames(summary_beta)=c("Mean", "Stand_Dev", "quant2.5%", "quant97.5%")
  if(ncol(x)==14) rownames(summary_beta)=c( paste( "BOTTOMCLS", 0:5, sep="_"), colnames(whitefish.dat.co
  else rownames(summary_beta)=c( paste( "BOTTOMCLS", 0:5, sep="_"), colnames(whitefish.dat.cov)[-c(1,2,4

  # larvae density f: mean and variance per each location
  Ef = rowMeans(EfMCMC)
  Varf = rowMeans(VarfMCMC) + rowSds(EfMCMC)^2
return( list("summary_beta"=summary_beta, "Ef"=Ef, "Varf"=Varf, "betas"=sampf_linMCMC, "EfMC"=EfMCMC))
}
```

With exp.covariance:

```
#p= prediction(m0, expCovariance, x, xpred , s, spred)
#saveRDS(p, file="pred_exp.RData")
p=readRDS("pred_exp.RData")

# linear coefficients(weights): beta estimates
p$summary_beta
```

```
##                     Mean Stand_Dev  quant2.5% quant97.5%
## BOTTOMCLS_0   0.02151013  4.892018  -6.722456  8.1395289
## BOTTOMCLS_1  -7.94973270 10.896729 -26.103619  6.7830632
## BOTTOMCLS_2   4.54003272  1.575264   1.704998  7.2297165
## BOTTOMCLS_3 -12.21092461 14.332544 -35.419093  7.2209839
## BOTTOMCLS_4  -2.12725243  5.471383 -11.159825  5.5936381
## BOTTOMCLS_5   0.74098414  3.021077  -4.218462  5.2062346
## FE300ME      -3.06240656  1.826911  -5.947174 -0.6176779
## DIS_SAND    -25.62855455 22.865180 -62.867762  5.5000497
## ICELAST09    -2.76064082  2.946301  -7.610932  1.3730066
## RIVERS        5.97318342  5.293252  -1.303198 14.6691966
## DIST20M     -12.26454273 10.752096 -29.747358  2.2758414
## CHL_A        -6.33378721  5.787758 -15.670345  1.5360187
## TEMP09M      25.95796542 22.925992  -5.343016 63.5124590
## SALT09M     -24.86298082 22.491285 -61.596553  5.8934474
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

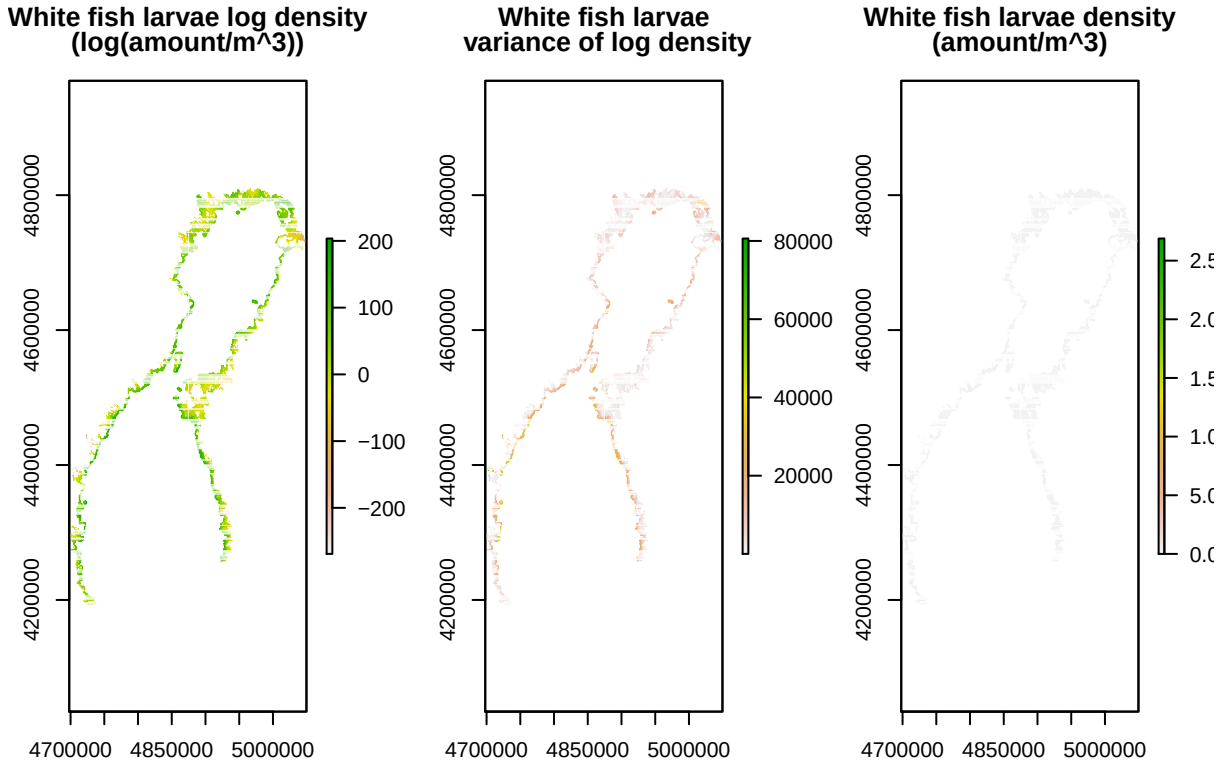
with Matern covariance:

```
# p1= prediction(m, Matern32Covariance, x, xpred , s, spred)
# saveRDS(p1, file="pred_mater.RData")
p1=readRDS("pred_mater.RData")

# linear coefficients(weights): beta estimates
p1$summary_beta
```

```
##                   Mean Stand_Dev   quant2.5% quant97.5%
## BOTTOMCLS_0 -19.226358 13.883513  -41.876142   6.203583
## BOTTOMCLS_1  31.902339 29.861391  -24.278459  78.918492
## BOTTOMCLS_2  -6.796395 11.540641  -25.127292  14.493003
## BOTTOMCLS_3  41.203649 39.875853  -34.009561 104.447275
## BOTTOMCLS_4  17.638888 13.991967   -8.364906  39.755921
## BOTTOMCLS_5  11.119864  7.037135   -2.312788  22.456766
## FE300ME       3.961735  5.783772   -6.651985  13.391057
## DIS_SAND     57.162051 62.251490  -59.789608 155.593552
## ICELAST09     5.719045  5.771077   -5.133910  14.910301
## RIVERS      -14.375634 15.582678  -39.031402  14.766619
## DIST20M      28.555855 31.330525  -30.464471  78.070844
## CHL_A        16.260767 17.640261  -16.800032  44.111526
## TEMP09M     -60.971021 66.173421 -165.316067  63.822088
## SALT09M      58.743898 63.444271  -60.965349 158.689458
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p1$Ef)
```

```
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p1$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p1$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

iid Random effects

To account for over dispersion we model Y with a negative binomial distribution.

$$y_i|\beta, \phi_i, \epsilon_i \sim Negative - Binomial(V_i e^{f(x_i,s_i)}, r)$$

That is equivalent to assume

$$y_i|\beta, \phi_i, \epsilon_i \sim Poisson(\epsilon_i V_i e^{f(x_i,s_i)})$$

where $\epsilon_i \sim Gamma(r, r)$ are independent random effects. For the dispersion parameter r, we assume as prior $r \sim Gamma(2, 1)$.

```
GP_whitefishmodel_ire = "
  data {
    int<lower=1> N;
```

```
  int<lower=1> Dx;
  matrix[N,Dx] x;
  int<lower=1> Ds;
  matrix[N,Ds] s;
  int<lower=0> y[N];
  vector[N] V;
}
transformed data {
  vector[N] mu;
  matrix[N, N] Dist_spatial;
  matrix[N, N] Sigma_lin;
  real s2_lin;
  s2_lin = 10;
  for (i in 1:N)
    mu[i] = 0;
  // off-diagonal elements
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      Dist_spatial[i, j] = pow(dot_self(s[i] - s[j]),0.5)  ;
      Sigma_lin[i, j] = s2_lin * dot_product(x[i],x[j]);     // linear covariance function

      // Fill in the other half
      Dist_spatial[j, i] = Dist_spatial[i, j];
      Sigma_lin[j, i] = Sigma_lin[i, j];
    }
  }
  // diagonal elements
  for (k in 1:N){
    Dist_spatial[k, k] = 0;
    Sigma_lin[k, k] = s2_lin * dot_product(x[k],x[k]) + 1e-6;   // add also some jitter
  }
}
parameters {
  real<lower=0> l;
  real<lower=0> s2_matern;
  vector[N] z;
  real<lower=0> r;
}
transformed parameters {
  matrix[N, N] Sigma;
  matrix[N, N] L;
  real<lower=0> inv_l;

  inv_l = inv(l);
Sigma = s2_matern*exp(-inv_l*Dist_spatial ) + Sigma_lin; // Exponential
 //     Sigma = s2_matern*(1 + pow(3,0.5)*inv_l*Dist_spatial).*exp(-pow(3,0.5)*inv_l*Dist_spatial)  +
  L = cholesky_decompose(Sigma);
}
model {
  vector[N] ff;

  // A weakly informative prior for magnitude
  s2_matern ~ student_t(4, 0, 1);
```

```
    // A weakly informative prior for l, that shrinks to 0 cor. among locations with more than 50km dis
    l ~ gamma(7, 0.1);

// A weakly informative prior for
    r ~ gamma(2, 0.1 );

    z ~ normal(0, 1);
    ff = L*z;

    for (n in 1:N) {
      y[n] ~ neg_binomial_2(V[n]*exp(ff[n]), r);
    }


  }
  generated quantities {
    vector[N] f;
    // derived quantity (transform)
    f = L*z;
  }"
```

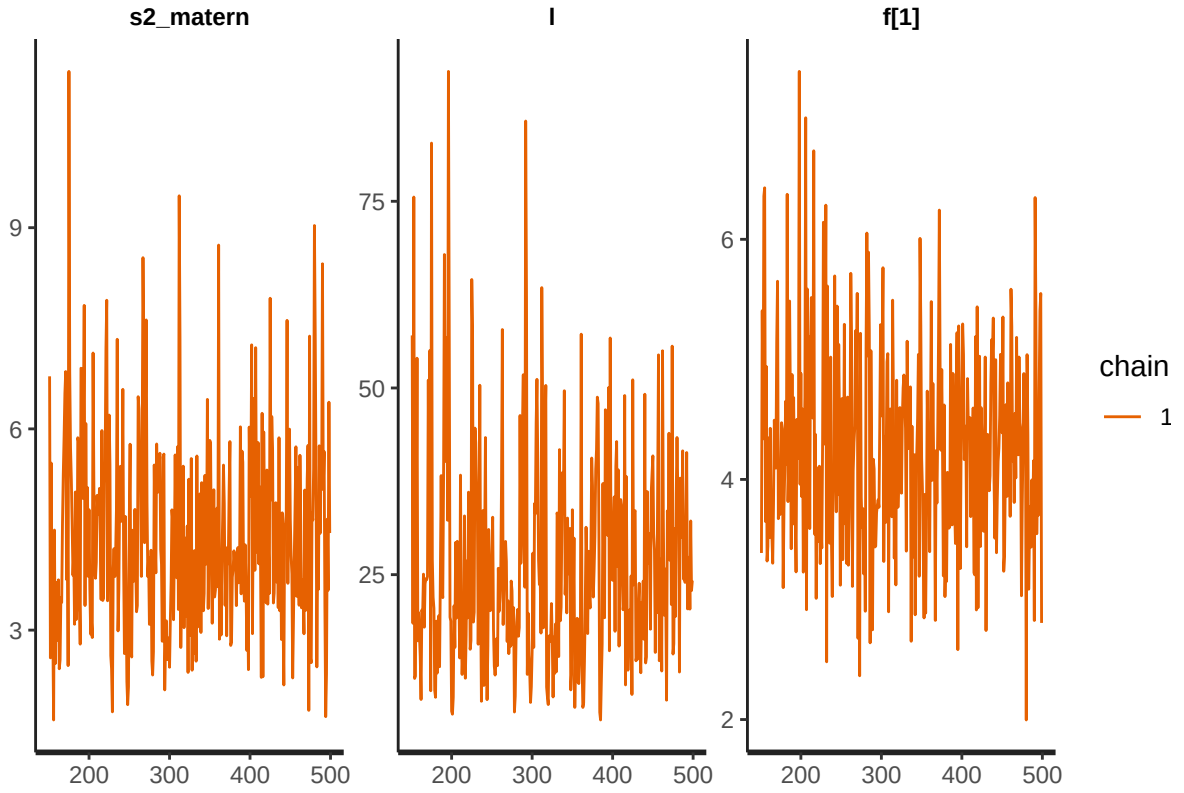
with exp covariance and iid random effects:

```
# fit1 = stan(model_code =  GP_whitefishmodel_ire, data = whitefish_dat, warmup=150, iter = 500, chains
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
# saveRDS(fit1, file = "fit_wf_ire_exp.rds")
fit1 <- readRDS("fit_wf_ire_exp_gamma.rds")


m1 = as.matrix(fit1)
#print(fit1)              # Rhat
summary(fit1, pars = c("s2_matern", "l", "f[1]"))            # summary
```

```
## $summary
##                mean    se_mean          sd     2.5%       25%       50%
## s2_matern  4.397279 0.10340165  1.4575859 2.272378  3.367786  4.159860
## l         26.550174 1.15166351 14.2660944 7.870027 16.184554 23.597483
## f[1]       4.244003 0.04820654  0.8423423 2.788936  3.655459  4.186406
##                75%     97.5%    n_eff      Rhat
## s2_matern  5.275089  7.684248 198.7071 0.9974423
## l         33.505898 57.875305 153.4472 0.9971464
## f[1]       4.770665  6.169996 305.3271 0.9990502
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter        mean         sd     2.5%       25%       50%       75%
##   s2_matern  4.397279  1.4575859 2.272378  3.367786  4.159860  5.275089
##   l         26.550174 14.2660944 7.870027 16.184554 23.597483 33.505898
##   f[1]       4.244003  0.8423423 2.788936  3.655459  4.186406  4.770665
##            stats
## parameter      97.5%
```

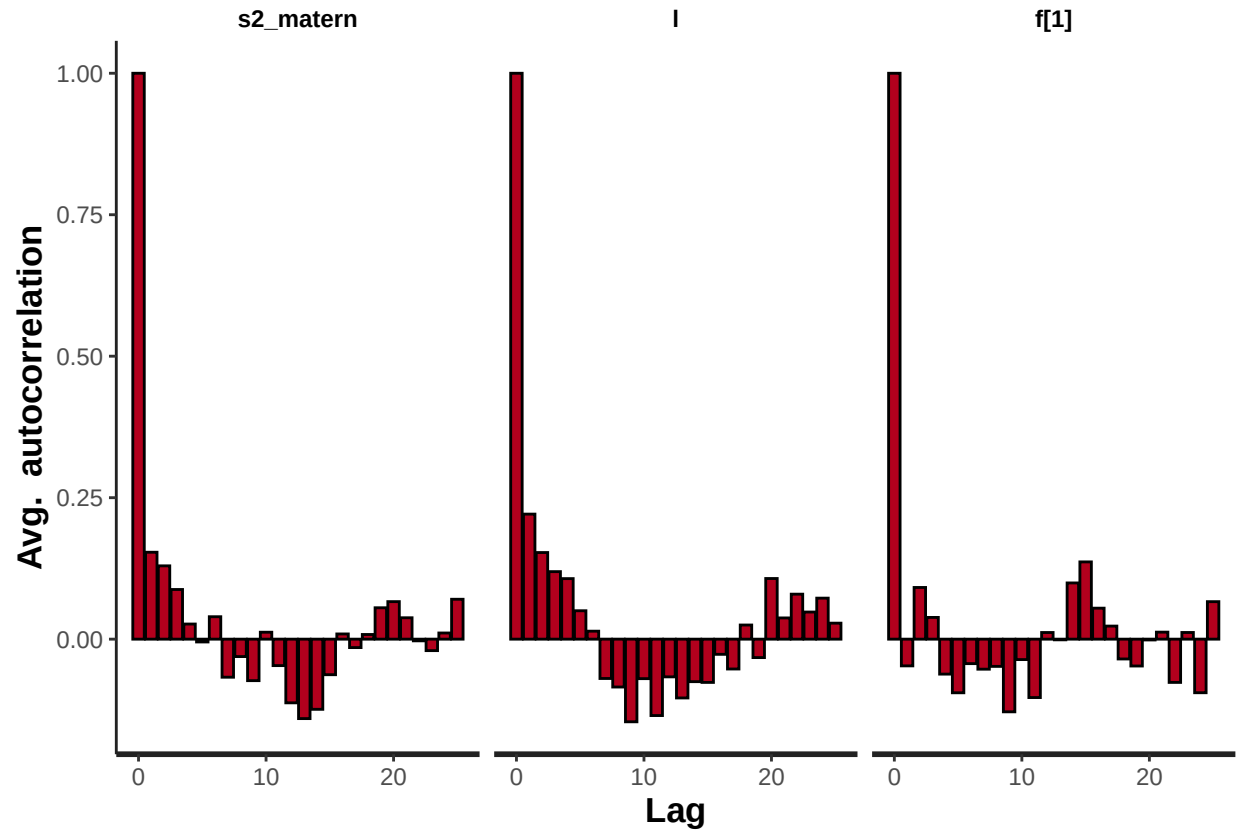
```
##   s2_matern  7.684248
##   l         57.875305
##   f[1]       6.169996
```

```
stan_trace(fit1, pars = c("s2_matern", "l", "f[1]"))          # traceplot
```

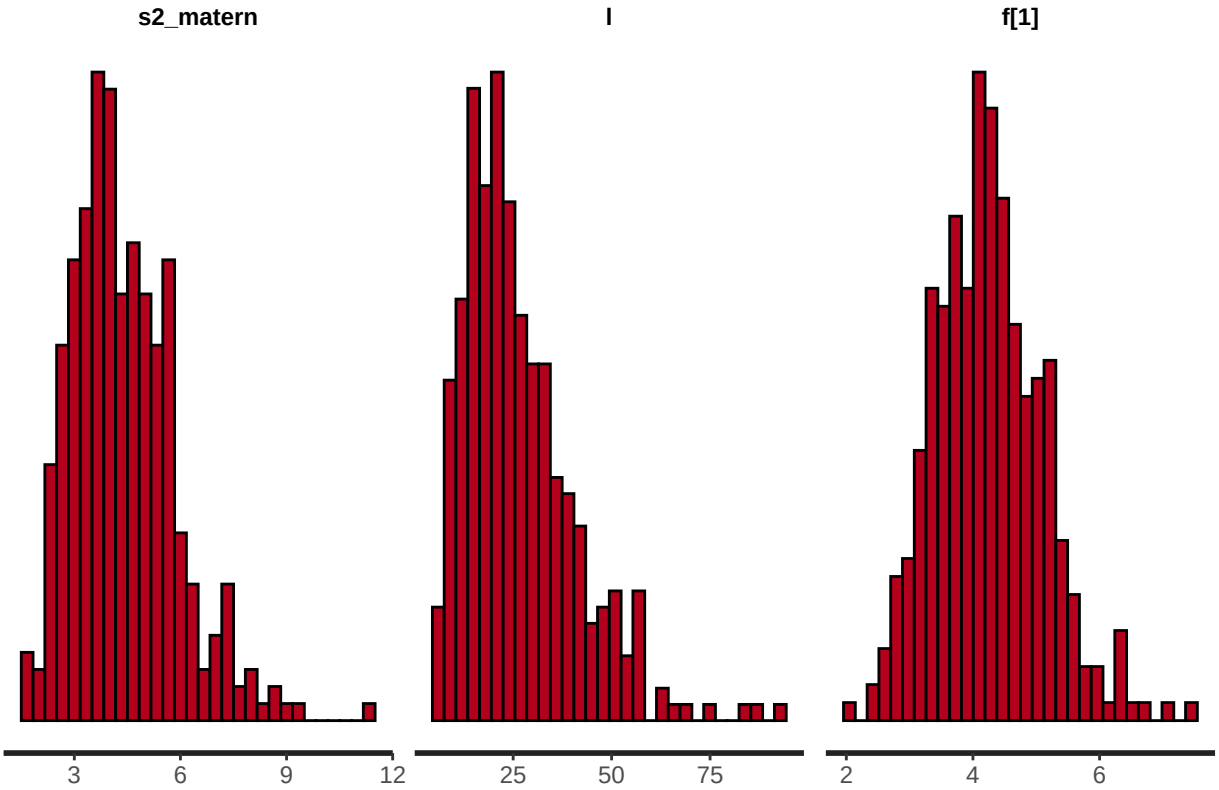



```
stan_ac(fit1,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation bet
```

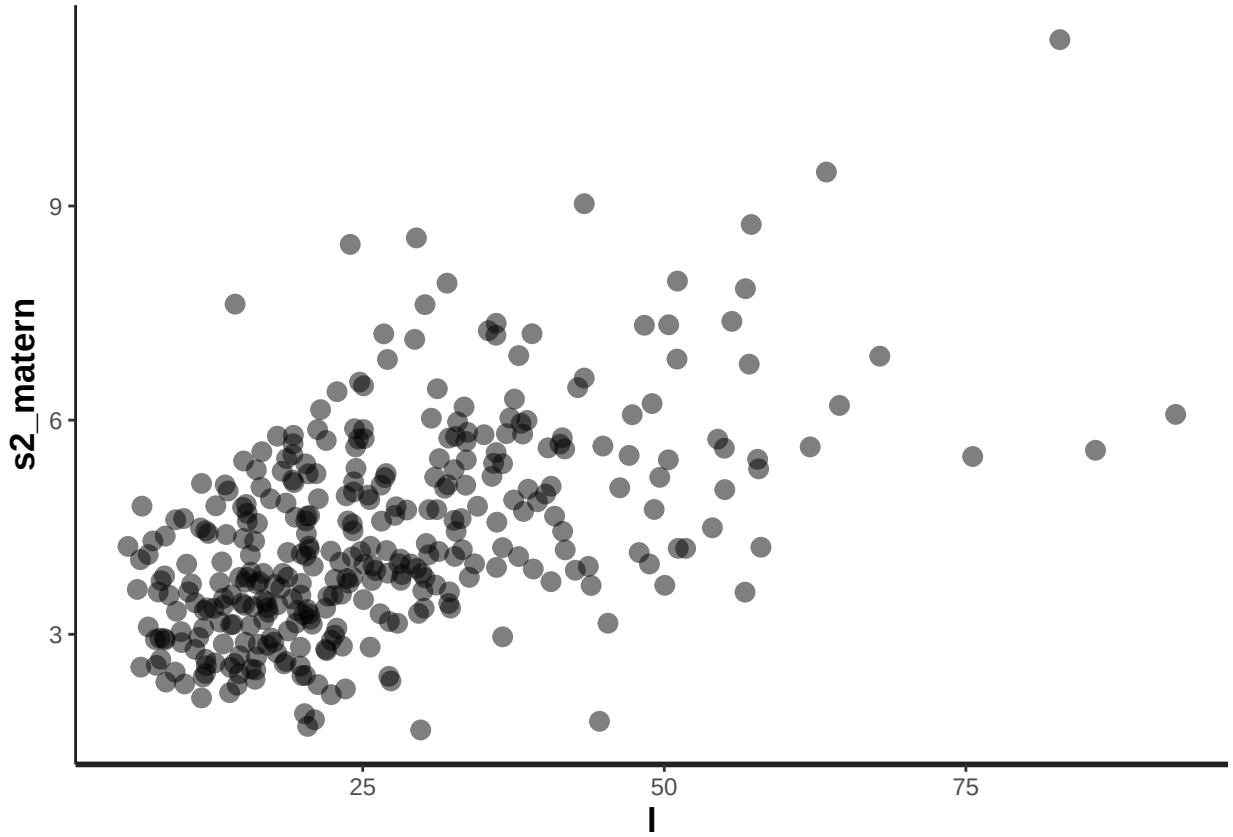
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit1, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```r
# scatter plot of parameters of spatial random effect
stan_scat(fit1, pars = c("l", "s2_matern"), color = "black", size = 3)
```



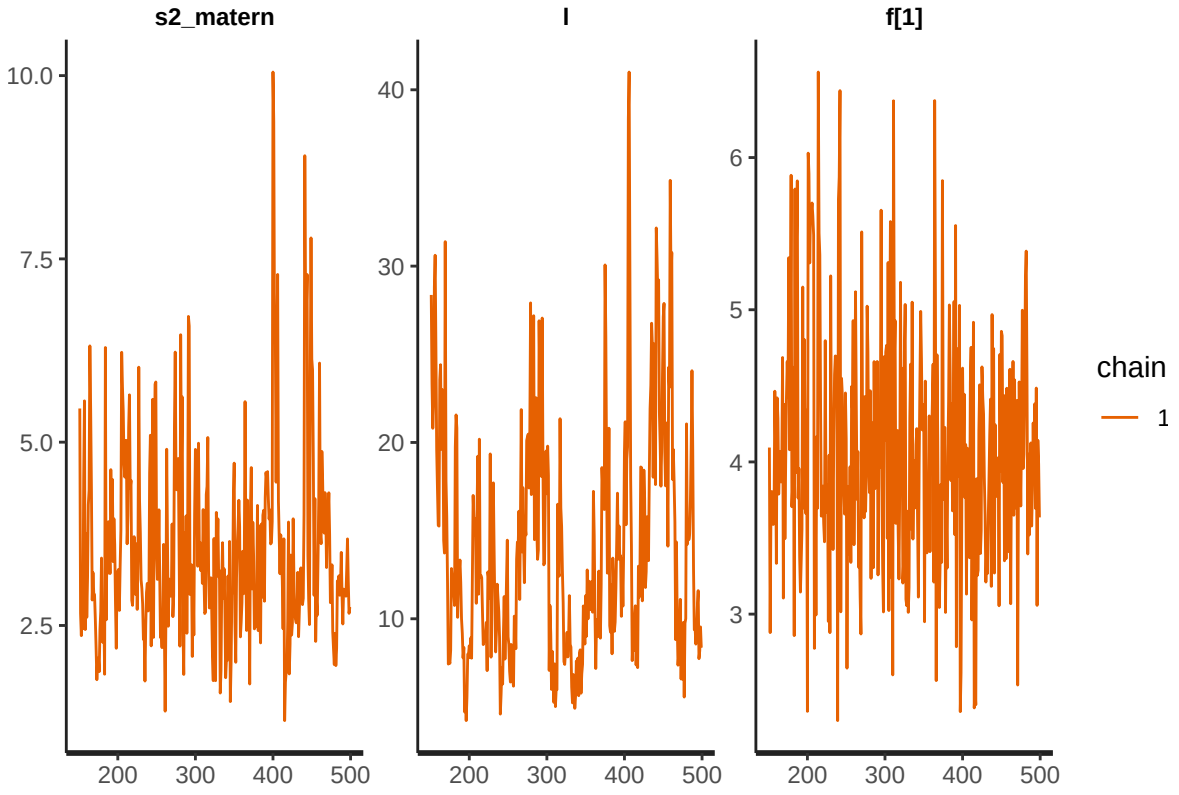
with Matern covariance and iid random effects:

```r
# fit2 = stan(model_code =  GP_whitefishmodel_ire, data = whitefish_dat, warmup=150, iter = 500, chains
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
# saveRDS(fit2, file = "fit_wf_ire_matern.rds")
fit2 <- readRDS("fit_wf_ire_matern_gamma.rds")


m2 = as.matrix(fit2)
#print(fit2)              # Rhat
summary(fit2, pars = c("s2_matern", "l", "f[1]"))            # summary
```

```
## $summary
##                mean   se_mean        sd     2.5%      25%       50%       75%
## s2_matern  3.530900 0.1539386 1.2987344 1.756799 2.668510  3.242945  4.077257
## l         14.038169 1.2761705 6.5608899 5.576570 8.970824 12.527173 17.922824
## f[1]       4.071438 0.0490382 0.7737365 2.611289 3.590556  4.020011  4.520454
##                97.5%     n_eff      Rhat
## s2_matern  6.620269  71.17797 0.9973665
## l         29.893351  26.43067 1.0004615
## f[1]       5.846997 248.95286 1.0149069
##
## $c_summary
## , , chains = chain:1
```

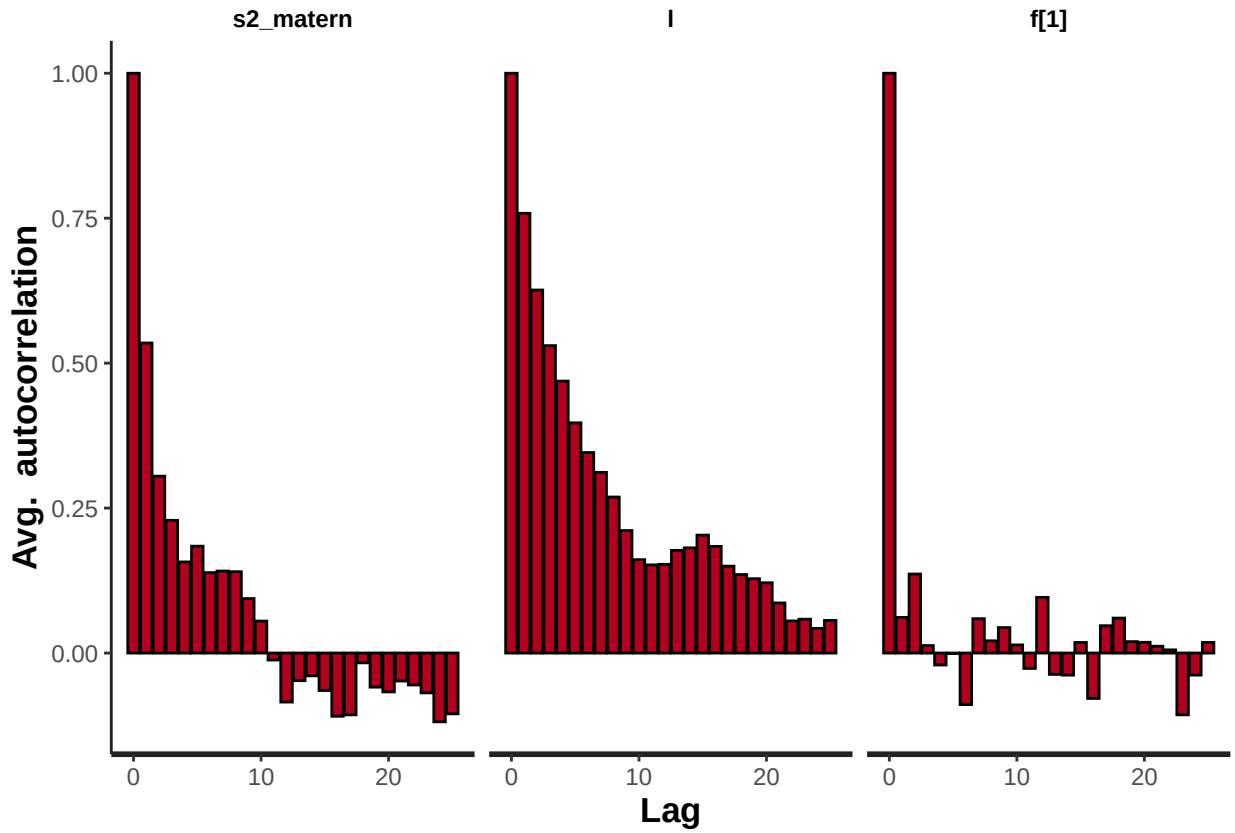
```
##
##            stats
## parameter        mean        sd     2.5%      25%       50%       75%     97.5%
##   s2_matern  3.530900 1.2987344 1.756799 2.668510  3.242945  4.077257  6.620269
##   l         14.038169 6.5608899 5.576570 8.970824 12.527173 17.922824 29.893351
##   f[1]       4.071438 0.7737365 2.611289 3.590556  4.020011  4.520454  5.846997
```

```
stan_trace(fit2, pars = c("s2_matern", "l", "f[1]"))         # traceplot
```

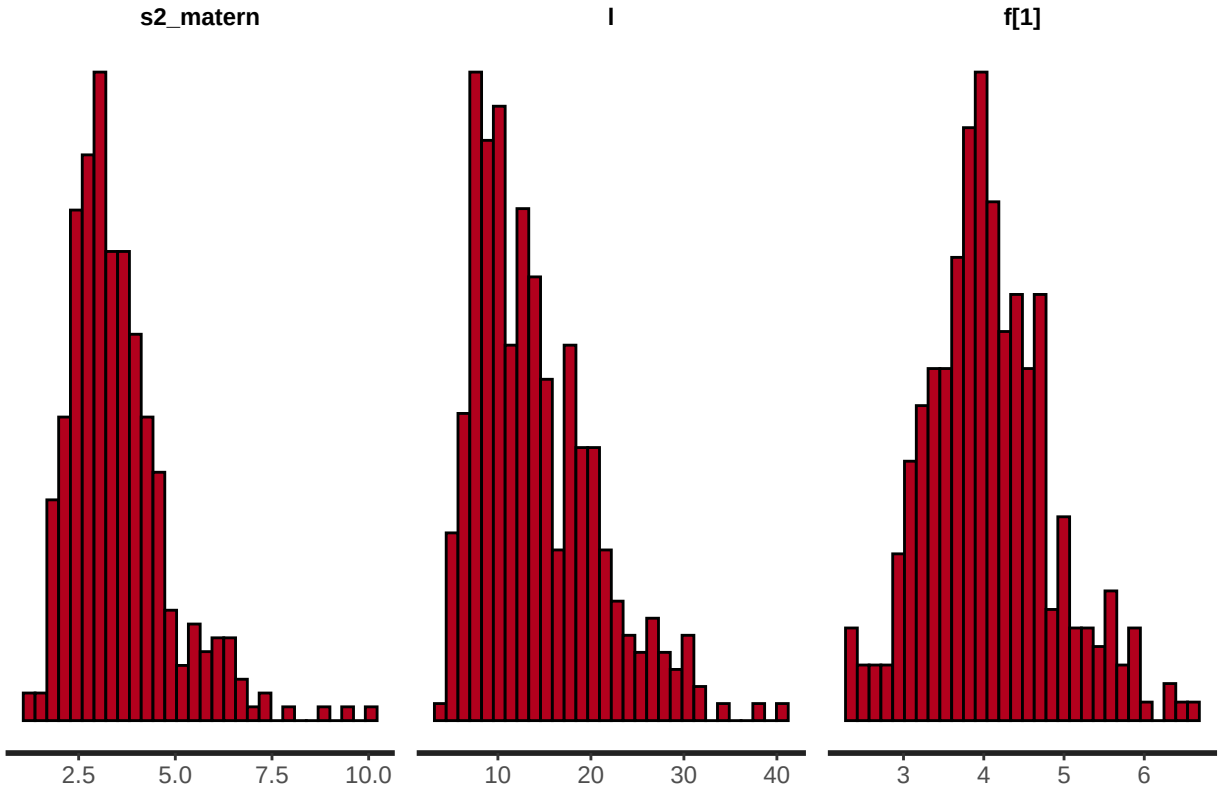



```
stan_ac(fit2,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation bet
```

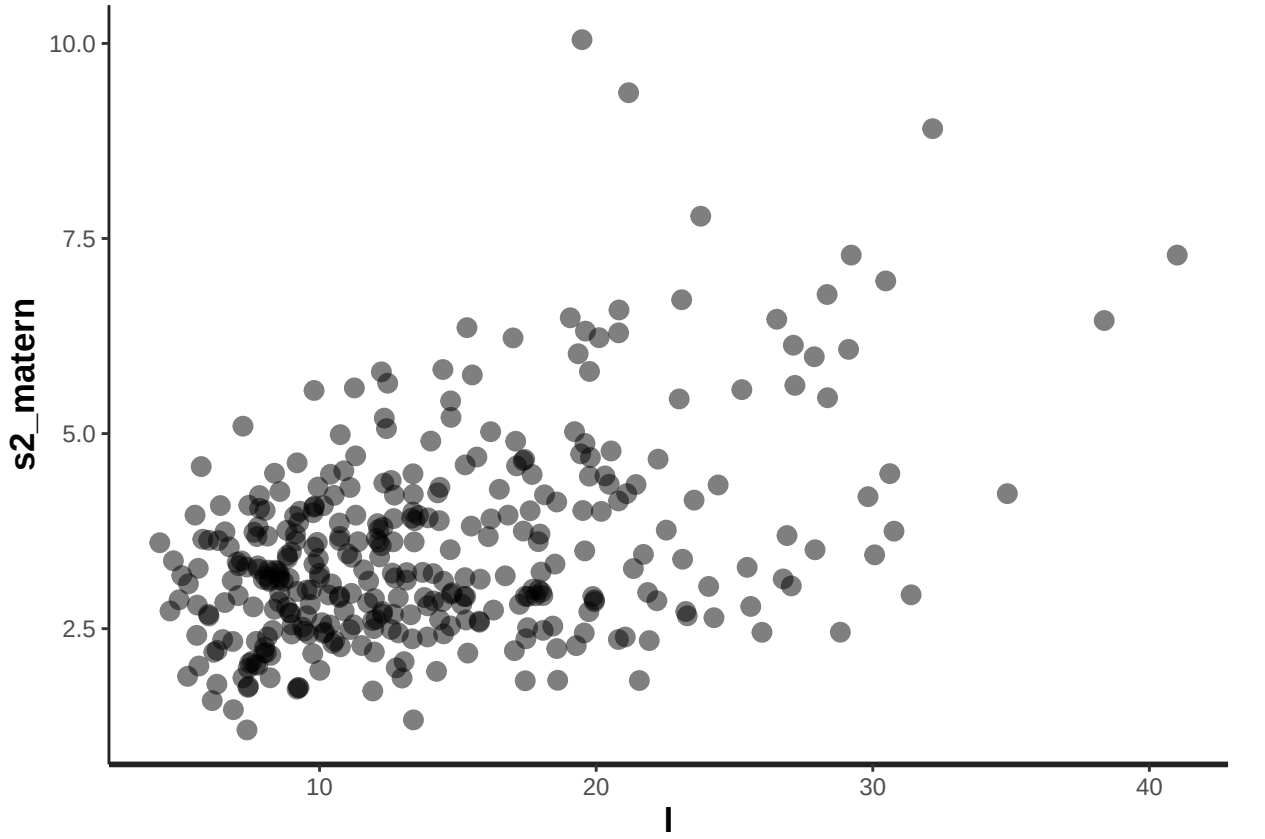
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit2, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```
# scatter plot of parameters of spatial random effect
stan_scat(fit2, pars = c("l", "s2_matern"), color = "black", size = 3)
```



Adding iid random effects to the model reduces autocorrelation and increases convergence of our posterior parametres. Nevertheless the overdispersion seems not solved yet, as l estimation is still ariund 100-200 m, hece it is lower than the data resolution.

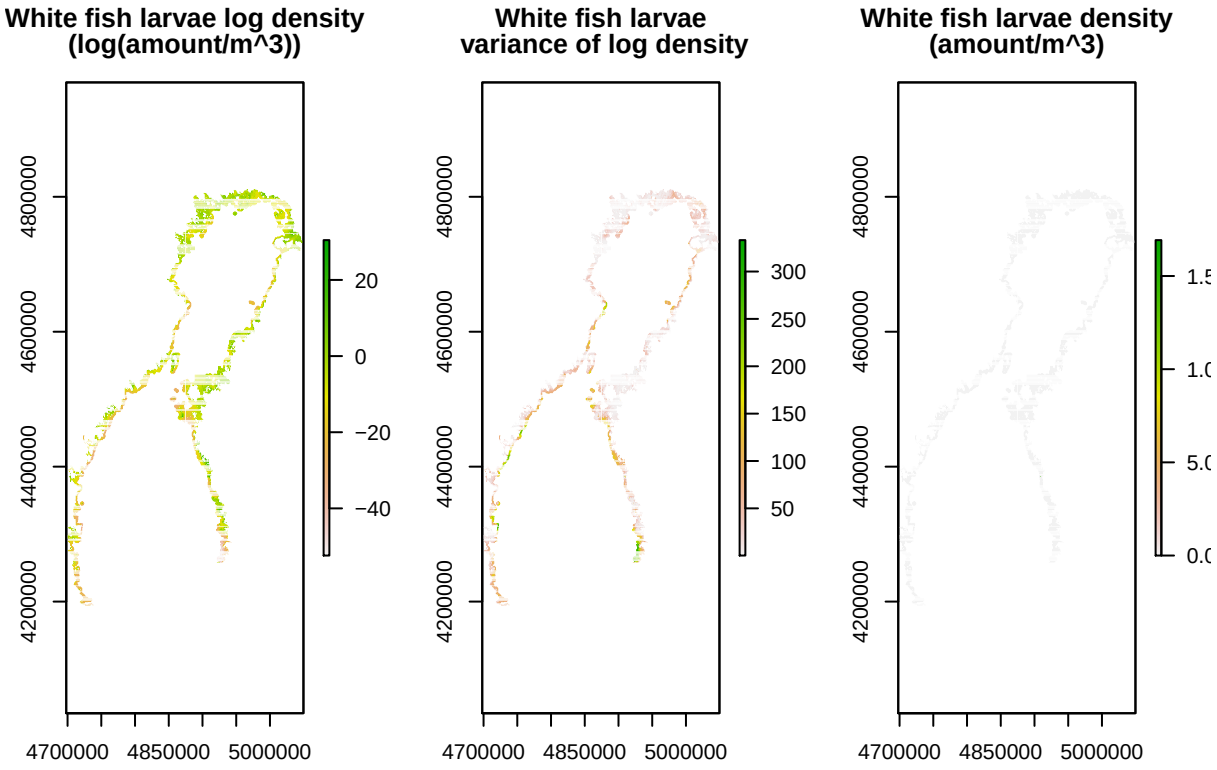
Let's see how predictions change. With exp.covariance:

```
#p3= prediction(m1, expCovariance, x, xpred , s, spred)
#saveRDS(p3, file="pred_exp_ire.RData")
p3=readRDS("pred_exp_ire.RData")

# linear coefficients(weights): beta estimates
p3$summary_beta
```

```
##                   Mean Stand_Dev    quant2.5% quant97.5%
## BOTTOMCLS_0 -1.7920297 0.8029136  -3.16768178 -0.7334971
## BOTTOMCLS_1  1.5589651 0.9287934  -0.03774671  2.9208456
## BOTTOMCLS_2 -0.7988603 1.2587529  -3.47893361  1.0005467
## BOTTOMCLS_3  0.2208273 1.5467416  -2.39144639  2.2529493
## BOTTOMCLS_4  1.5832842 0.7151851   0.32062167  2.5659390
## BOTTOMCLS_5  1.2362796 0.5512854   0.50660709  2.2433395
## FE300ME     -9.6331772 3.0388945 -16.00536846 -4.5570158
## DIS_SAND    -8.7616760 4.4575462 -17.38233662 -2.9673723
## ICELAST09   -1.1193613 0.5736391  -2.13848164 -0.2029349
## RIVERS       1.9327147 1.0207803   0.36768144  3.9186839
## DIST20M     -2.0439980 1.6351844  -5.14665239  0.1546927
```

```
## CHL_A       -4.3903751 2.2210629  -9.66164497 -0.8322096
## TEMP09M     -3.1162781 1.9241262  -6.76300263  0.2106239
## SALT09M      6.0477792 2.3108623   1.80338602 10.1934341
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p3$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p3$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p3$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```



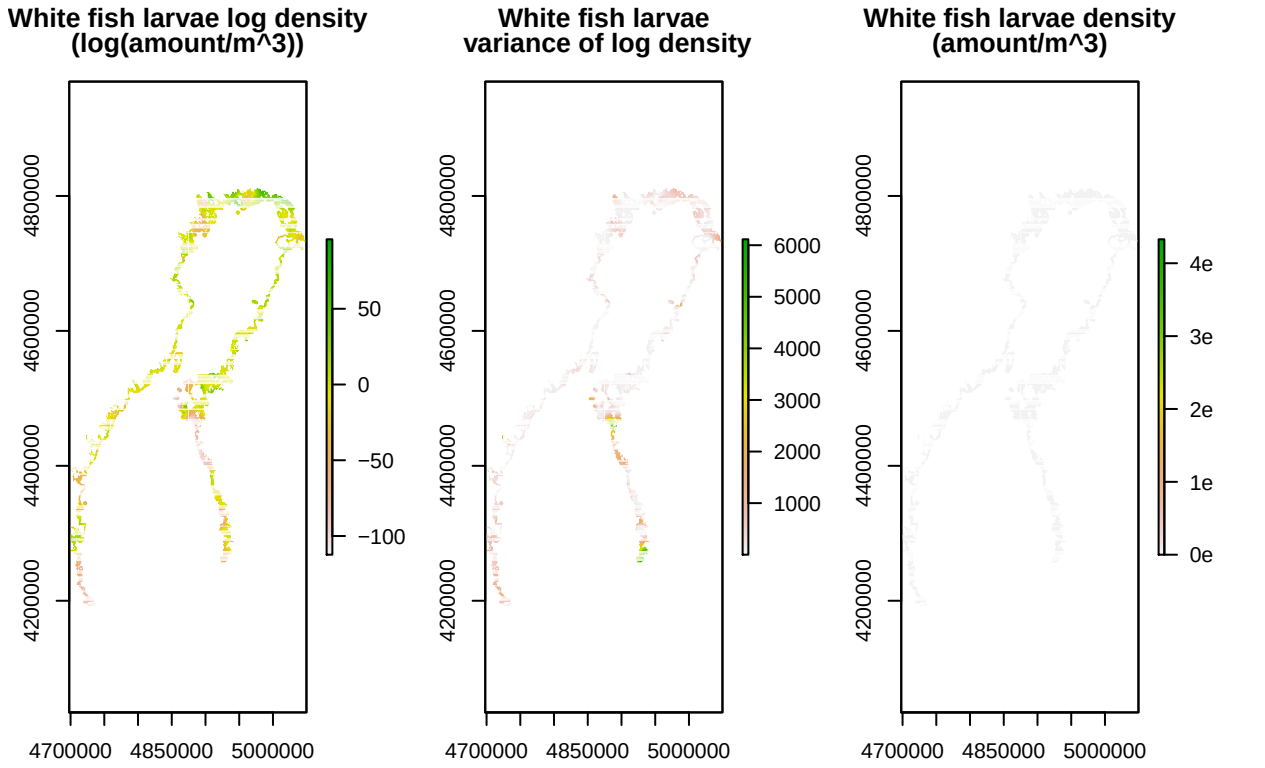

```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

with Matern covariance:

```
#p4= prediction(m2, Matern32Covariance, x, xpred , s, spred)
#saveRDS(p4, file="pred_mater_ire.RData")
p4=readRDS("pred_mater_ire.RData")

# linear coefficients(weights): beta estimates
p4$summary_beta
```

```
##                   Mean  Stand_Dev    quant2.5% quant97.5%
## BOTTOMCLS_0  -1.838932  3.1831577  -9.01121532   2.267093
## BOTTOMCLS_1   4.792570  1.3747743   2.97287647   7.561994
## BOTTOMCLS_2   4.143804  1.2285756   2.08343907   6.242645
## BOTTOMCLS_3   3.723420  0.8501036   2.50043050   5.568907
## BOTTOMCLS_4   3.888608  1.2542986   2.04757988   6.753083
## BOTTOMCLS_5   2.902690  1.2030116   1.52330077   5.738371
## FE300ME     -12.279548 10.5011265 -26.25328830  10.737270
## DIS_SAND      1.555742  0.6174351   0.68901048   2.870660
## ICELAST09     4.918151  3.2776809  -0.03491385  12.840881
## RIVERS       -4.943907  4.4492723 -11.68226404   5.769936
## DIST20M      12.619600  4.7963545   6.83396949  22.352234
## CHL_A        21.608707  8.2230849   6.28099981  35.396677
## TEMP09M     -13.651398  9.1527902 -29.49139794   4.017755
## SALT09M      -0.944679 19.4470255 -30.40728306  36.726225
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p4$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p4$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p4$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

Vendace SDM

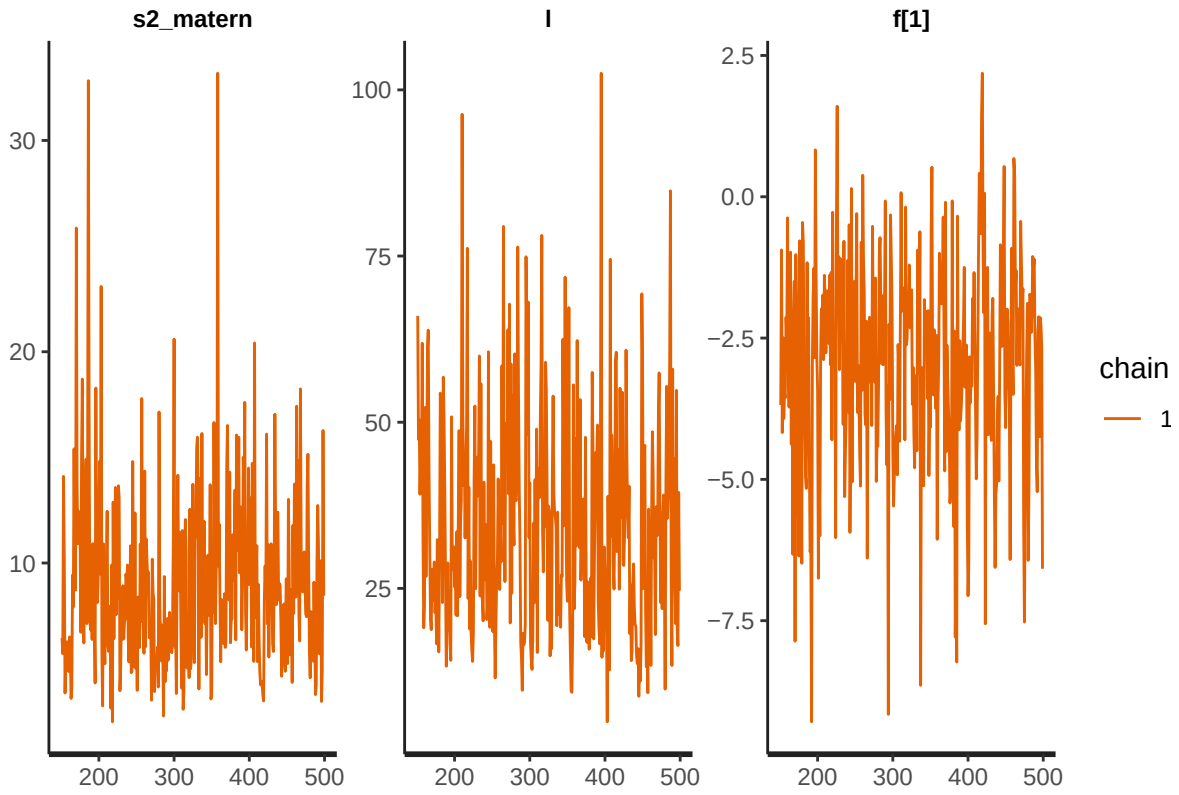
We now implement a HSDM for vendace abboundance, in the same way we implemented the last whitefish model in the previous section, with exponential covariance

```
y.ven = whitefish.dat$VENSUM[-vol0]

# n = nrow(whitefish.dat)
# m = 217 #smaller dataset
# y.ven = y.ven[seq(1,n,length=m)]
# x = x[seq(1,n,length=m),]
# s = s[seq(1,n,length=m),]
# V = V[seq(1,n,length=m)]


ven_data <-  list(Dx = ncol(x),
                  N = nrow(x),
                  Ds = ncol(s),
                  x = x,
                  s = s,
                  y = y.ven,
                  V = V)
# fit.v = stan(model_code =  GP_whitefishmodel_ire, data = ven_data, warmup=150, iter = 500, chains = 1
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
#saveRDS(fit.v, file = "fit_v_ire_exp_gamma.rds")
fit.v <- readRDS("fit_v_ire_exp_gamma.rds")
```

```r
m = as.matrix(fit.v)


#print(fit.v)              # Rhat
summary(fit.v, pars = c("s2_matern", "l", "f[1]"))            # summary
```

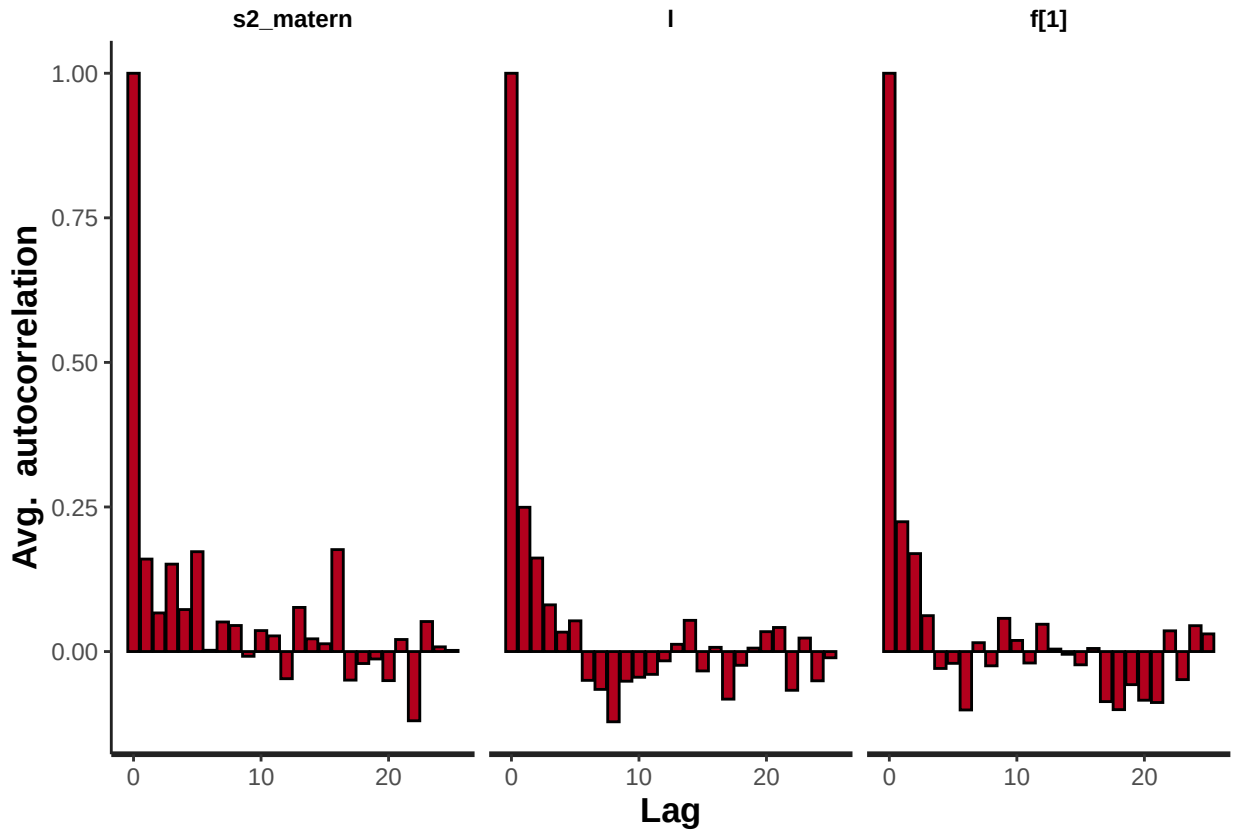
```
## $summary
##                mean   se_mean        sd      2.5%       25%       50%       75%
## s2_matern  9.066733 0.3566606  4.204266  3.580947  6.139980  8.331732 10.902924
## l         34.439697 1.2780331 16.352208 12.400558 21.781744 31.946794 44.637579
## f[1]      -2.975778 0.1375748  1.865130 -6.834051 -4.068735 -2.830109 -1.758638
##                97.5%    n_eff     Rhat
## s2_matern 17.9096629 138.9537 1.005397
## l         72.5620535 163.7075 1.007028
## f[1]       0.3815185 183.7978 0.998683
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter        mean        sd      2.5%       25%       50%       75%
##   s2_matern  9.066733  4.204266  3.580947  6.139980  8.331732 10.902924
##   l         34.439697 16.352208 12.400558 21.781744 31.946794 44.637579
##   f[1]      -2.975778  1.865130 -6.834051 -4.068735 -2.830109 -1.758638
##            stats
## parameter       97.5%
##   s2_matern 17.9096629
##   l         72.5620535
##   f[1]       0.3815185
```

```r
stan_trace(fit.v, pars = c("s2_matern", "l", "f[1]"))         # traceplot
```

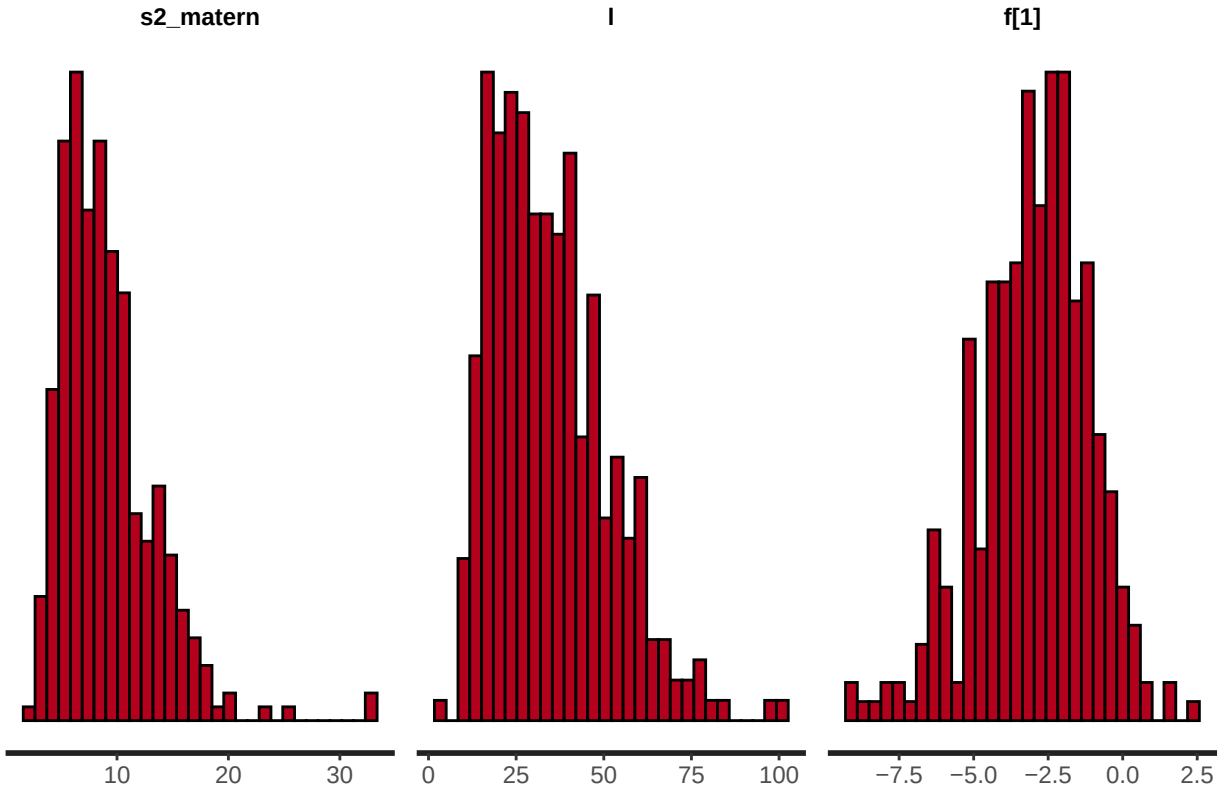



```
stan_ac(fit.v,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation be
```

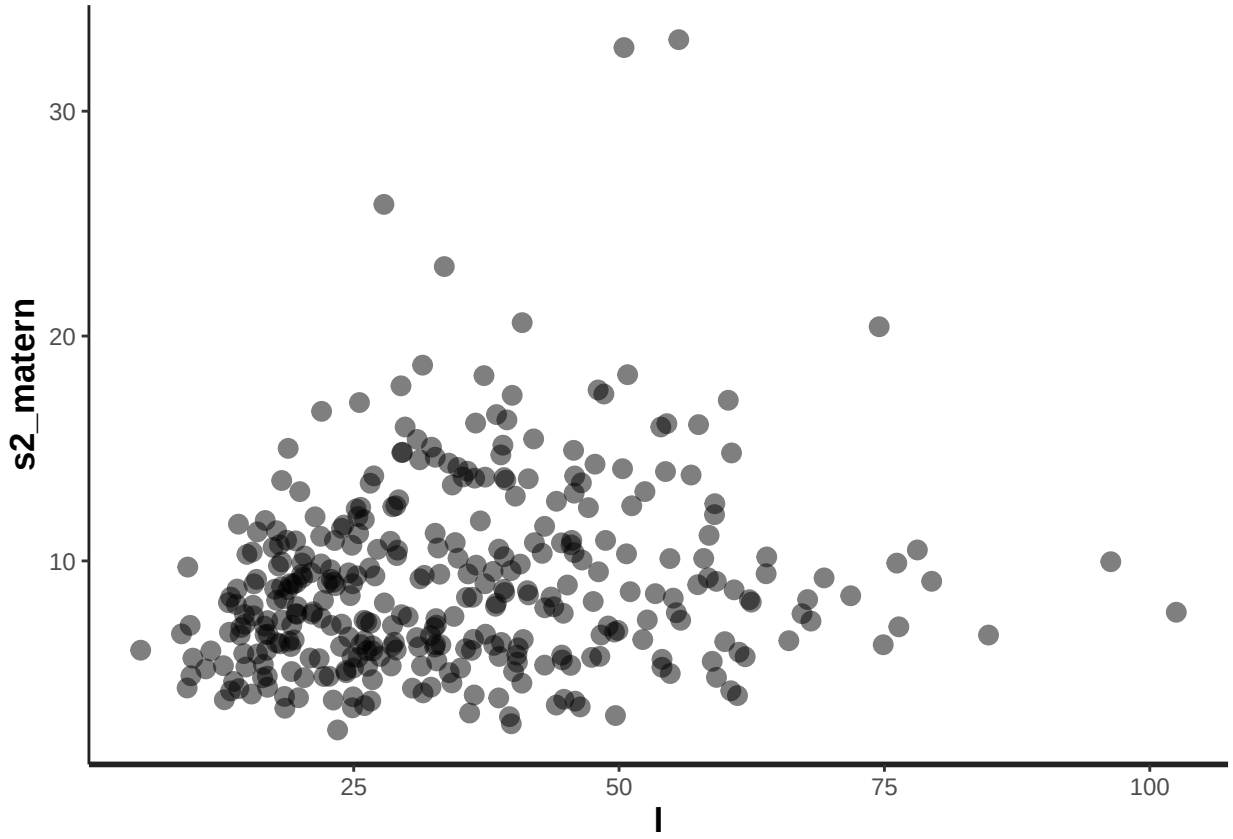
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit.v, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```
# scatter plot of parameters of spatial random effect
stan_scat(fit.v, pars = c("l", "s2_matern"), color = "black", size = 3)
```



The model works fine, the autocorrelation is less problematic.

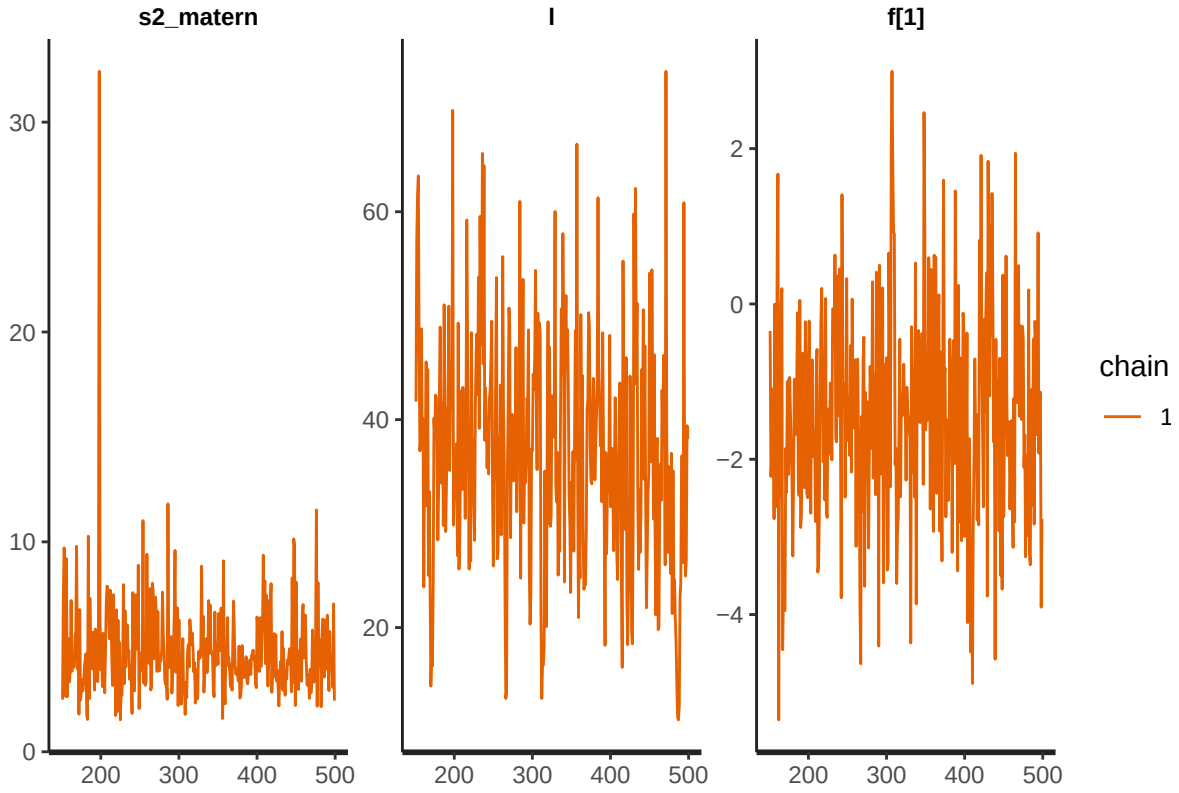
With matern covariance

```
# fit.v = stan(model_code =  GP_whitefishmodel_ire, data = ven_data, warmup=150, iter = 500, chains = 1
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
#saveRDS(fit.v, file = "fit_v_ire_matern_gamma.rds")
fit.v2 <- readRDS("fit_v_ire_matern_gamma.rds")
m2 = as.matrix(fit.v2)


#print(fit.v)              # Rhat
summary(fit.v2, pars = c("s2_matern", "l", "f[1]"))            # summary
```

```
## $summary
##                mean    se_mean        sd      2.5%       25%       50%
## s2_matern  4.825220 0.13991060  2.385841  2.016819  3.362254  4.401016
## l         36.926620 0.96119498 10.974087 16.270874 29.740242 36.891643
## f[1]      -1.429949 0.09285596  1.358964 -3.991628 -2.345910 -1.454705
##                 75%     97.5%    n_eff      Rhat
## s2_matern  5.7167955  9.626049 290.7915 1.0009457
## l         43.0965126 61.086154 130.3508 1.0111852
## f[1]      -0.6050704  1.493876 214.1885 0.9980173
##
## $c_summary
```

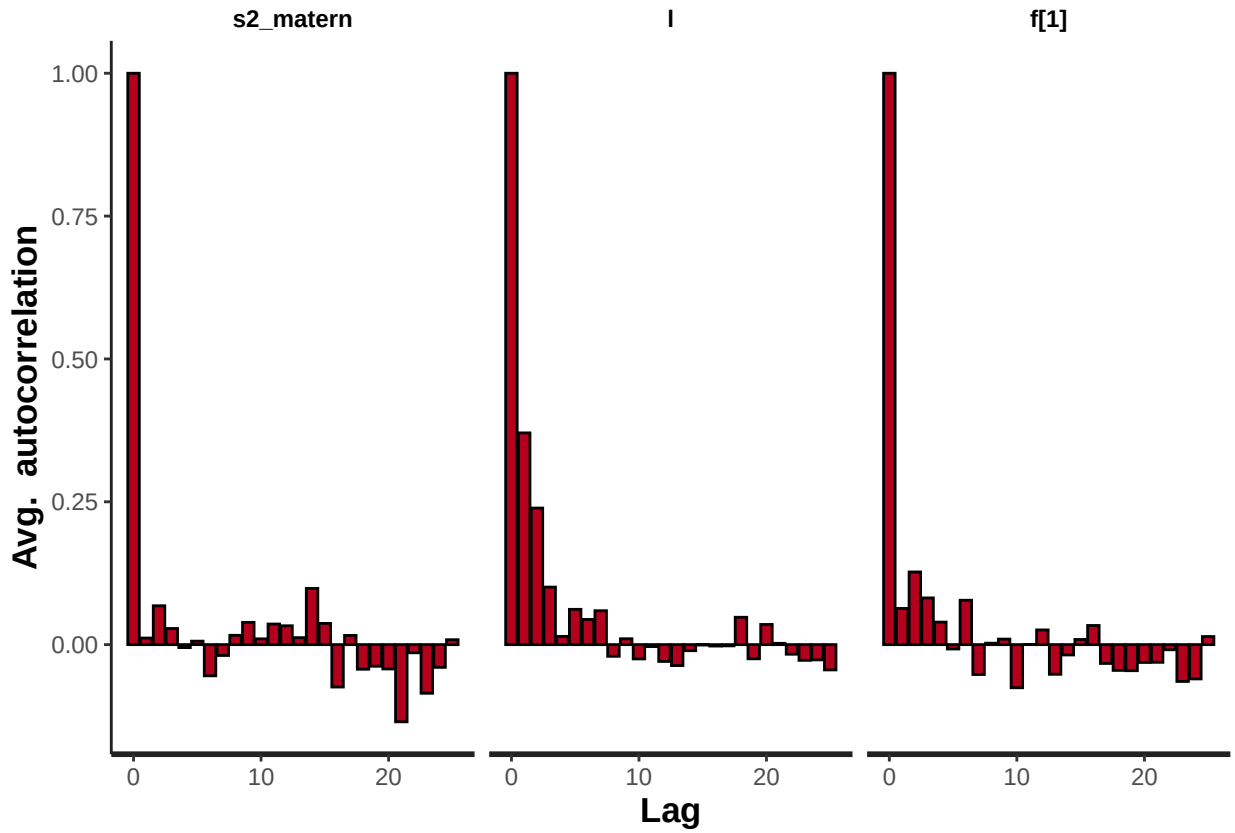
```
## , , chains = chain:1
##
##            stats
## parameter        mean        sd      2.5%       25%       50%        75%
##   s2_matern  4.825220  2.385841  2.016819  3.362254  4.401016  5.7167955
##   l         36.926620 10.974087 16.270874 29.740242 36.891643 43.0965126
##   f[1]      -1.429949  1.358964 -3.991628 -2.345910 -1.454705 -0.6050704
##            stats
## parameter       97.5%
##   s2_matern  9.626049
##   l         61.086154
##   f[1]       1.493876
```

```
stan_trace(fit.v2, pars = c("s2_matern", "l", "f[1]"))          # traceplot
```

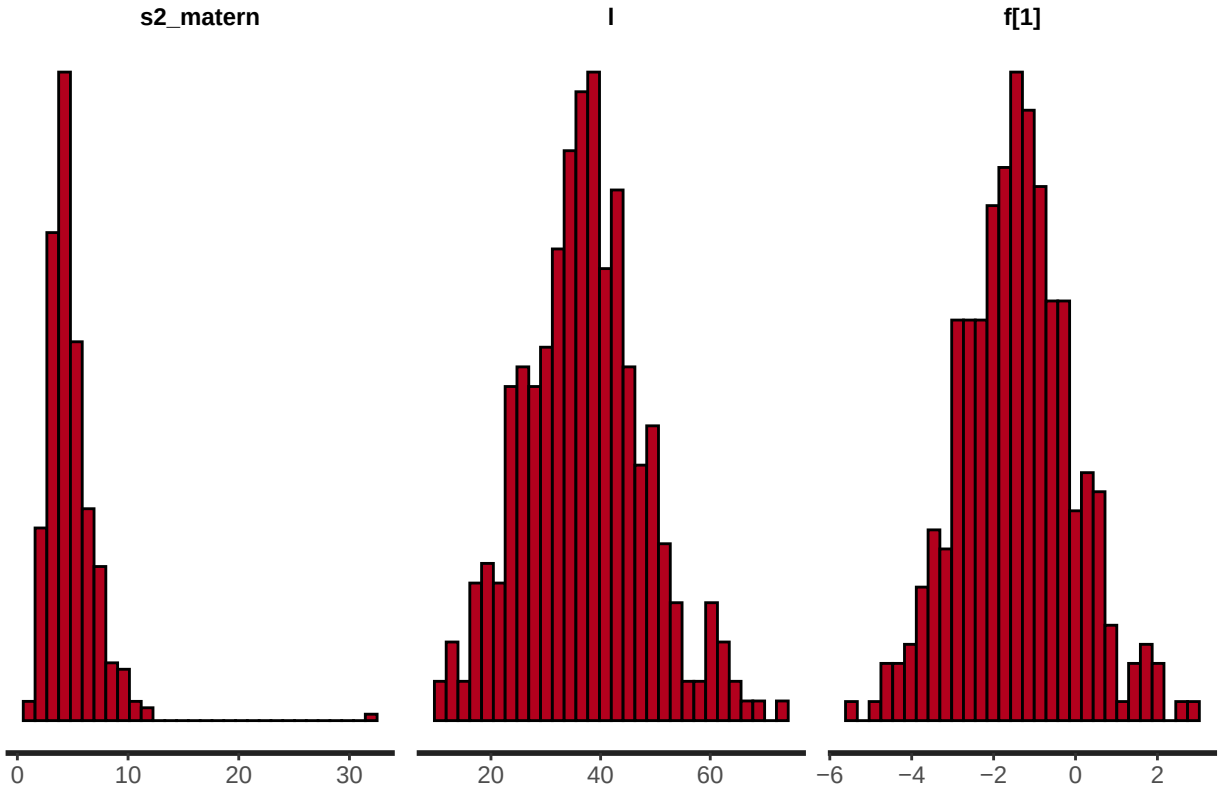



```
stan_ac(fit.v2,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation b
```

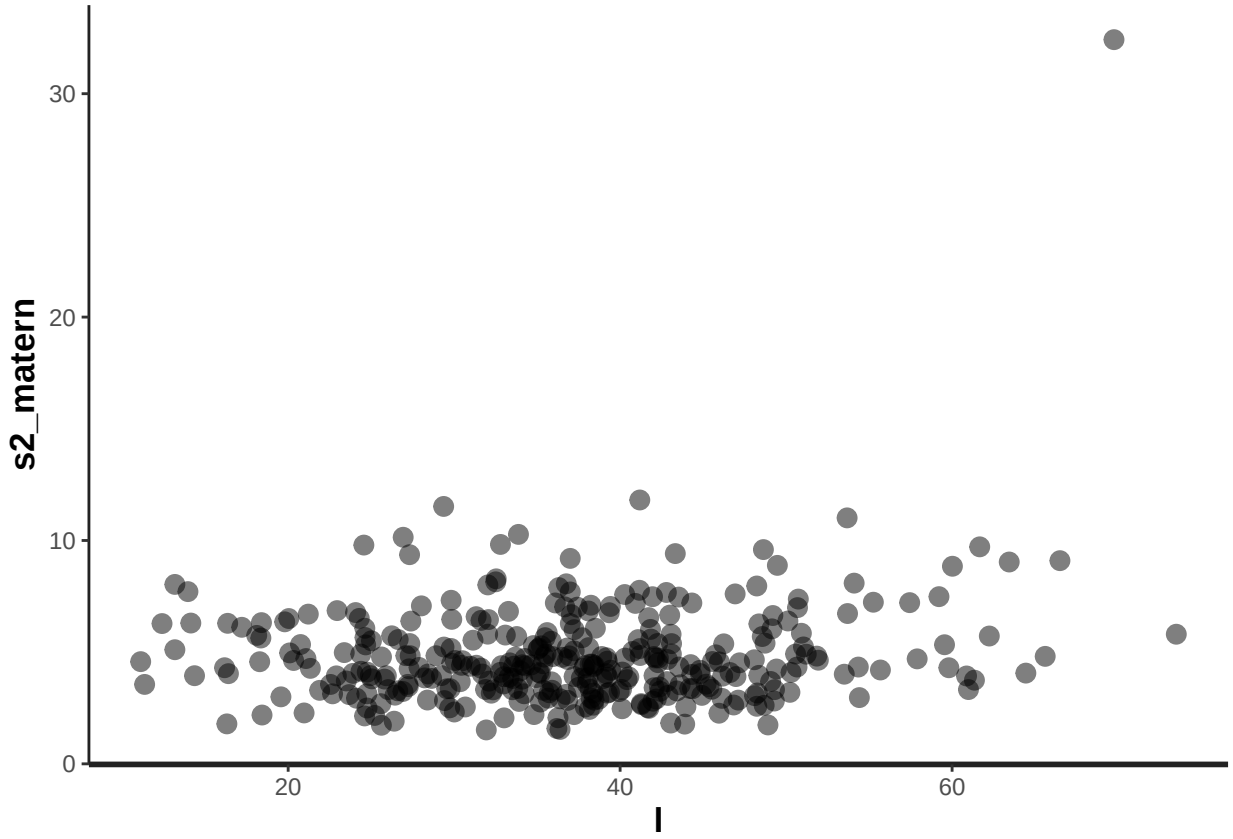
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit.v2, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```
# scatter plot of parameters of spatial random effect
stan_scat(fit.v2, pars = c("l", "s2_matern"), color = "black", size = 3)
```

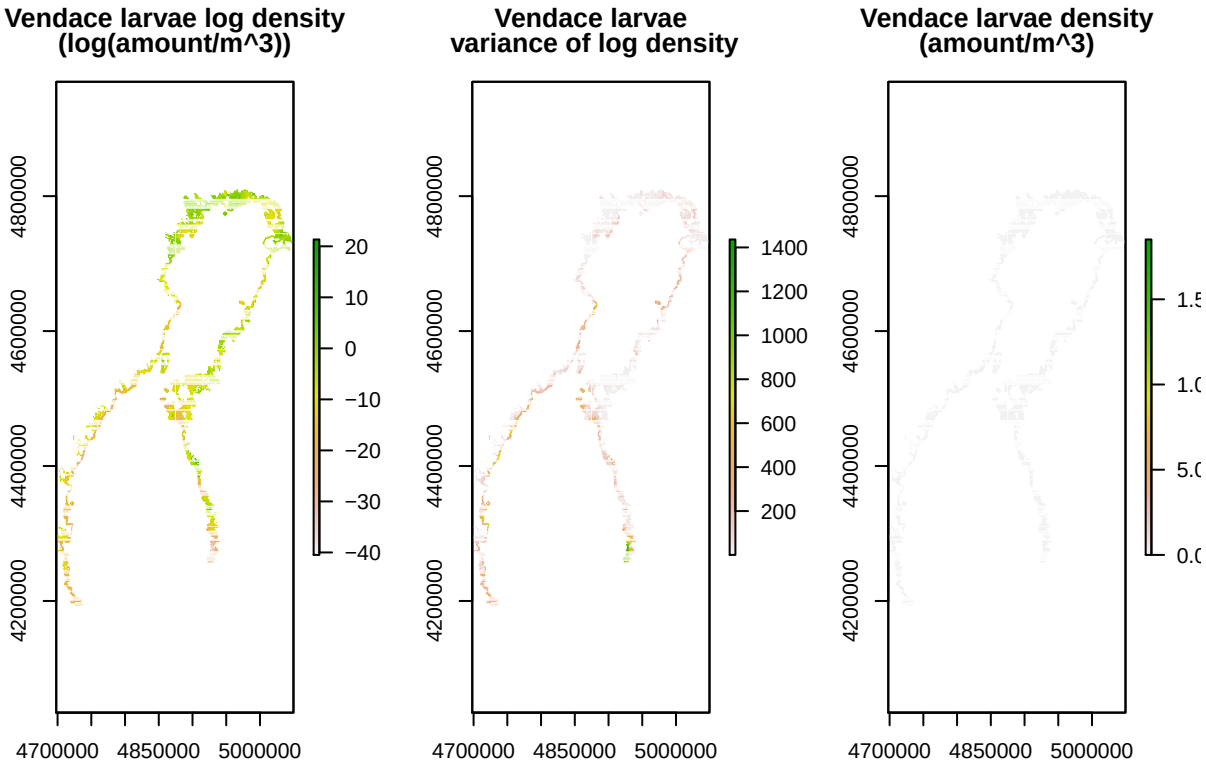


We do spatial prediction with exp.covariance:

```
#p= prediction(m, expCovariance, x, xpred , s, spred)
#saveRDS(p, file="pred_exp_v.RData")
p=readRDS("pred_exp_v.RData")

# linear coefficients(weights): beta estimates
p$summary_beta
```

```
##                   Mean Stand_Dev  quant2.5% quant97.5%
## BOTTOMCLS_0 -6.3474930 1.3352409  -9.007461 -3.9474189
## BOTTOMCLS_1  1.0065128 1.4124953  -2.103872  3.0715427
## BOTTOMCLS_2 -0.6921200 2.1501864  -5.120356  3.3721914
## BOTTOMCLS_3 -0.5199866 2.4629629  -5.186740  3.3910459
## BOTTOMCLS_4  0.5737600 1.1894409  -1.731858  2.4008895
## BOTTOMCLS_5  0.7094450 0.9147844  -1.373633  2.2967107
## FE300ME     -2.7734510 5.5829129 -14.393347  8.8932341
## DIS_SAND    -5.2235887 7.4799568 -20.966404  6.0329953
## ICELAST09   -1.8760081 0.9556802  -3.755450 -0.5519029
## RIVERS       1.0704028 1.8173650  -1.425183  5.2834092
## DIST20M     -0.4067052 2.5100221  -5.685520  3.3552815
## CHL_A       -4.3744302 4.7046750 -14.192077  2.4703457
## TEMP09M     -5.9152688 3.2531130 -12.470688 -1.0093605
## SALT09M      3.9249892 3.5856671  -2.456285 11.3827568
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae l
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae"
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae d
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

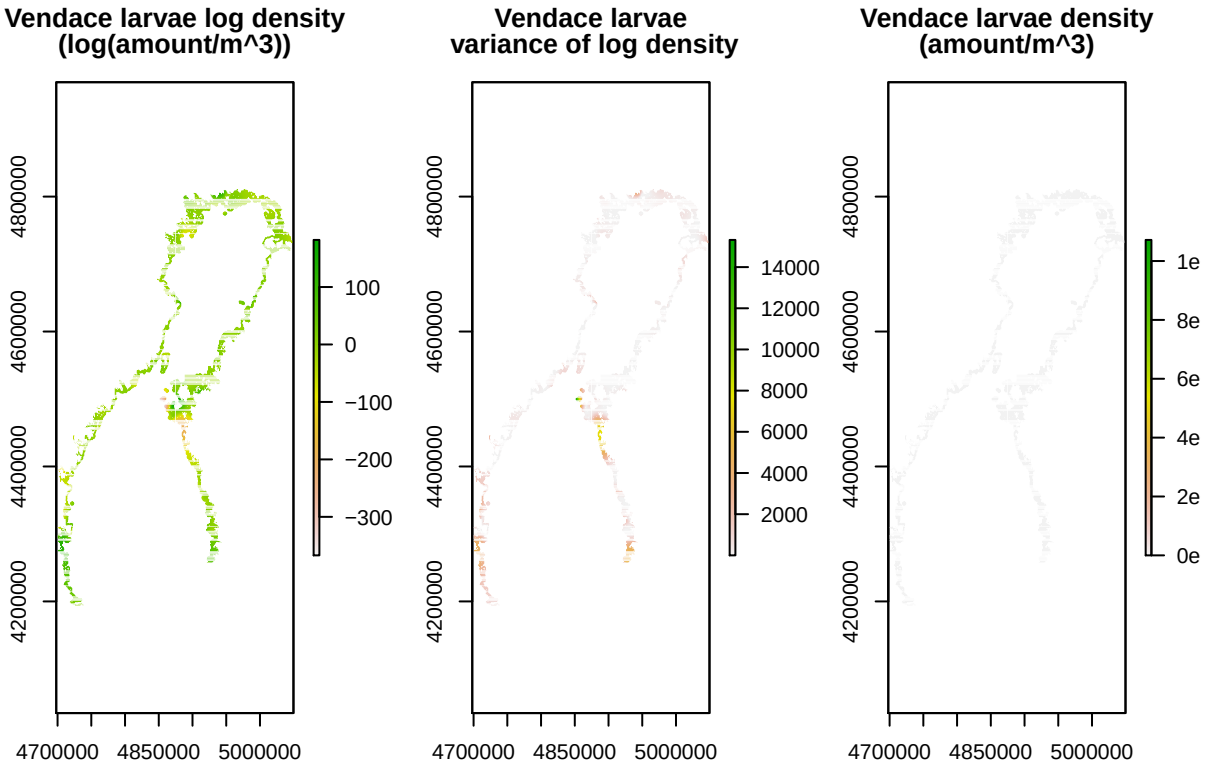
And with matedrn covariance

```
#p= prediction(m2, Matern32Covariance, x, xpred , s, spred)
#saveRDS(p, file="pred_mater_v.RData")
p=readRDS("pred_mater_v.RData")

# linear coefficients(weights): beta estimates
p$summary_beta
```

```
##                    Mean  Stand_Dev   quant2.5% quant97.5%
## BOTTOMCLS_0 -32.739285  8.0240804 -45.8787474 -20.016984
```

```
## BOTTOMCLS_1   2.265752  0.5005544   1.4732971   3.557764
## BOTTOMCLS_2   1.343979  0.8883513   0.2771863   3.538488
## BOTTOMCLS_3   1.577109  0.5875312   0.4350434   2.731284
## BOTTOMCLS_4   1.381209  0.8363622   0.5145056   2.826867
## BOTTOMCLS_5   2.141583  1.0966908   0.8923706   4.861834
## FE300ME      15.338925  8.4744455   5.9673912  34.403692
## DIS_SAND      2.534576  0.7138618   1.5829592   4.412096
## ICELAST09    -3.493229  1.2023646  -6.1279848  -2.040287
## RIVERS       -3.570418  4.2820296 -16.2493182   1.232343
## DIST20M       3.152478  2.2377176   0.2684626   9.893341
## CHL_A         1.995395 12.8335298 -15.3239532  29.881392
## TEMP09M     -13.036748  6.5404508 -25.9708088  -3.081861
## SALT09M      30.871444 22.4231024   1.2554655  79.104046
```

```r
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae l
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae"
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae d
```



```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

Quadratic effect

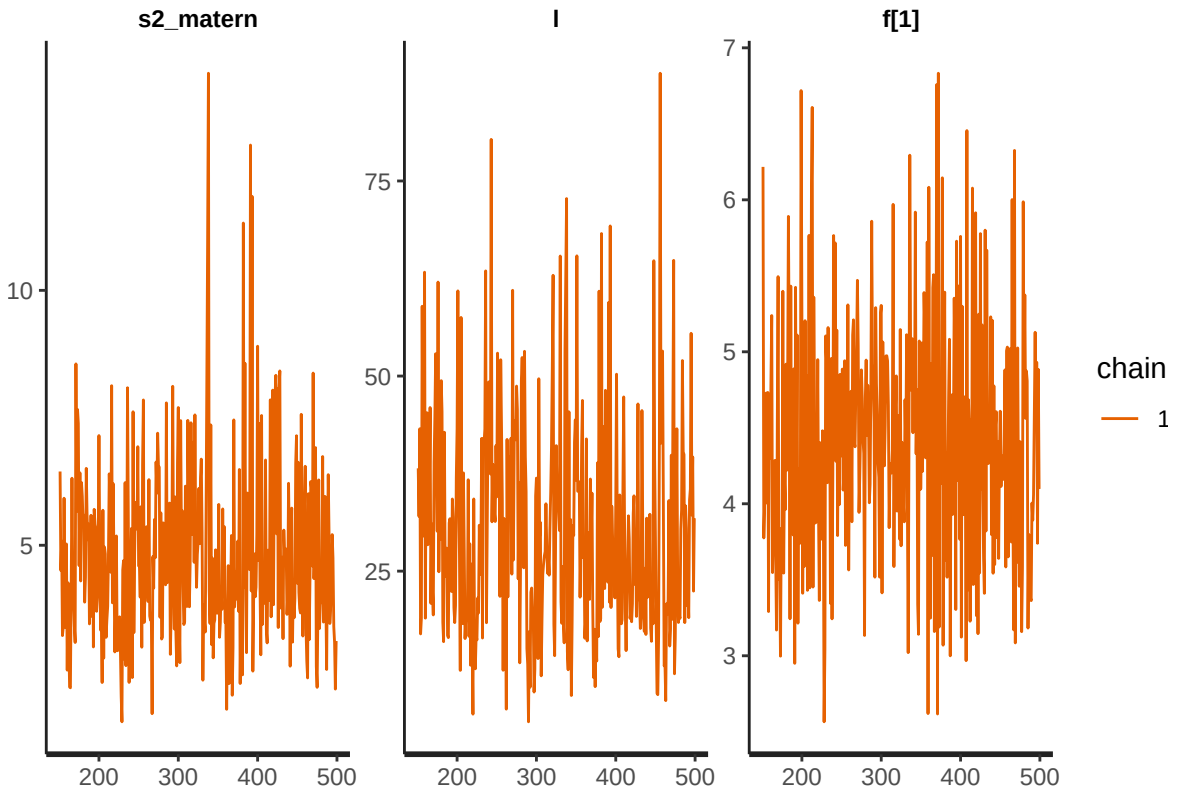
```
x2=apply(x[,7:14], 2, '^',2)
x_q=cbind(x,x2)
```

Whitefish, with exponential covariance

```
wf_data_q <-  list(Dx = ncol(x_q),
                   N = nrow(x_q),
                   Ds = ncol(s),
                   x = x_q,
                   s = s,
                   y = y,
                   V = V)
# fit.w.q = stan(model_code =  GP_whitefishmodel_ire, data = wf_data_q, warmup=150, iter = 500, chains =
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
# saveRDS(fit.w.q, file = "fit_w2.rds")
fit.w.q = readRDS("fit_w2.rds")

m.wq = as.matrix(fit.w.q)
#print(fit.w.q)              # Rhat
summary(fit.w.q, pars = c("s2_matern", "l", "f[1]"))            # summary
```

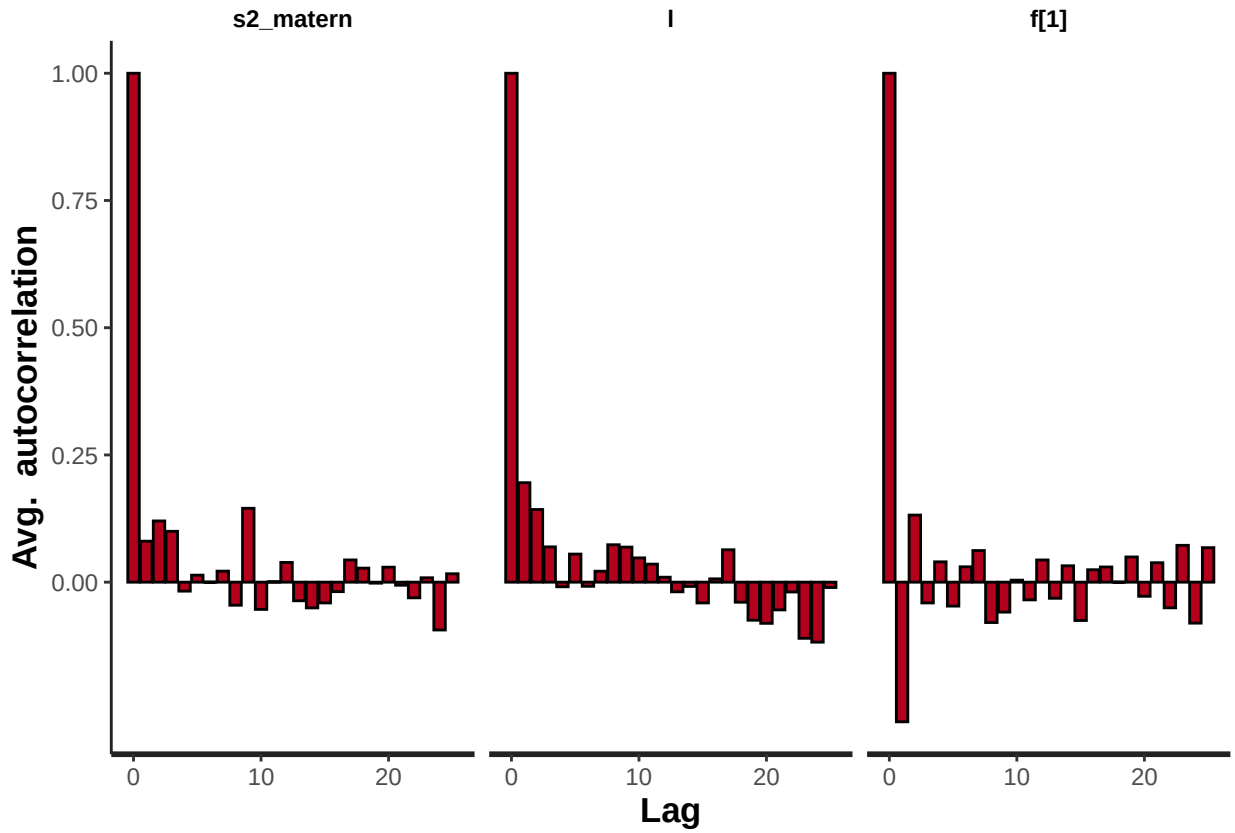
```
## $summary
##                mean    se_mean         sd      2.5%       25%       50%
## s2_matern  4.813869 0.12013220  1.7748090  2.268713  3.589664  4.496625
## l         30.538628 1.03798240 13.9813961 10.176357 19.243591 28.862361
## f[1]       4.425796 0.03373926  0.8035619  3.061619  3.827384  4.405241
##                75%     97.5%    n_eff      Rhat
## s2_matern  5.701611  8.389158 218.2651 0.9977153
## l         38.810804 63.830866 181.4350 1.0014107
## f[1]       4.940799  6.099453 567.2410 0.9973556
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter        mean         sd      2.5%       25%       50%       75%
##   s2_matern  4.813869  1.7748090  2.268713  3.589664  4.496625  5.701611
##   l         30.538628 13.9813961 10.176357 19.243591 28.862361 38.810804
##   f[1]       4.425796  0.8035619  3.061619  3.827384  4.405241  4.940799
##            stats
## parameter       97.5%
##   s2_matern  8.389158
##   l         63.830866
##   f[1]       6.099453
```

```
stan_trace(fit.w.q, pars = c("s2_matern", "l", "f[1]"))         # traceplot
```

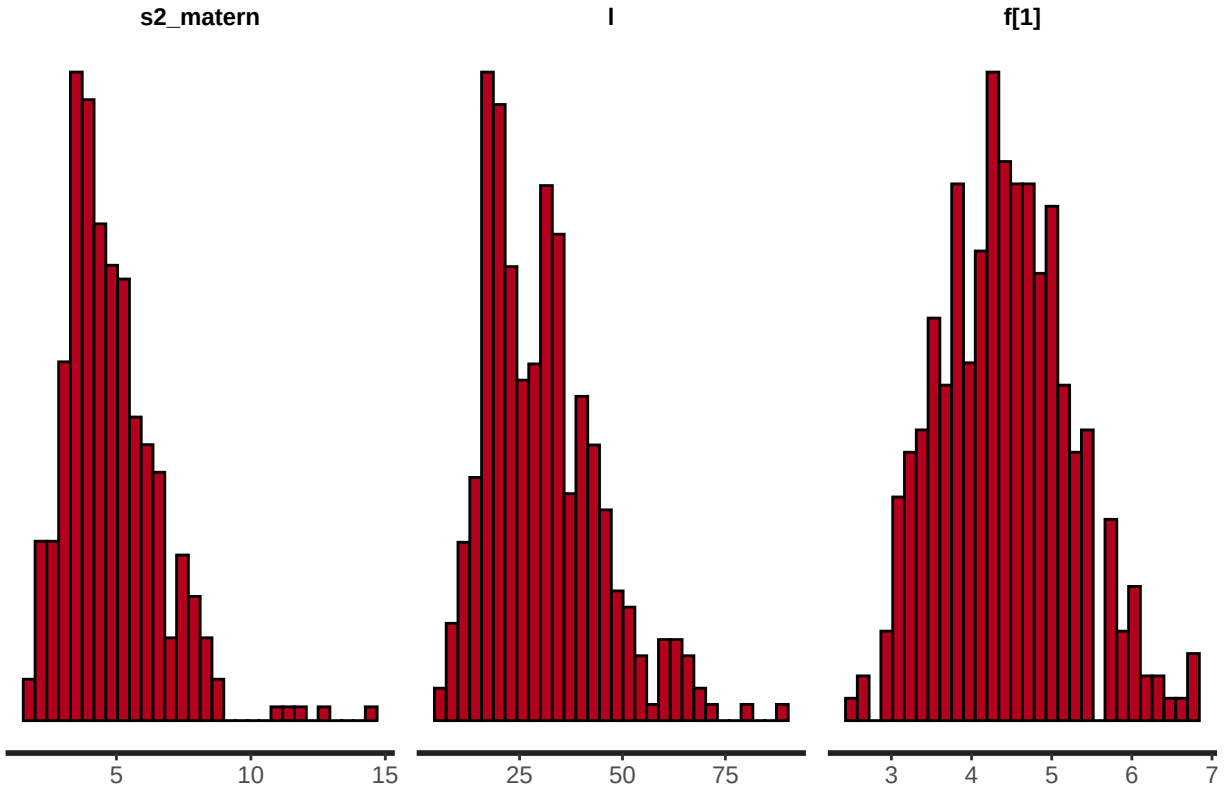



```
stan_ac(fit.w.q,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation
```

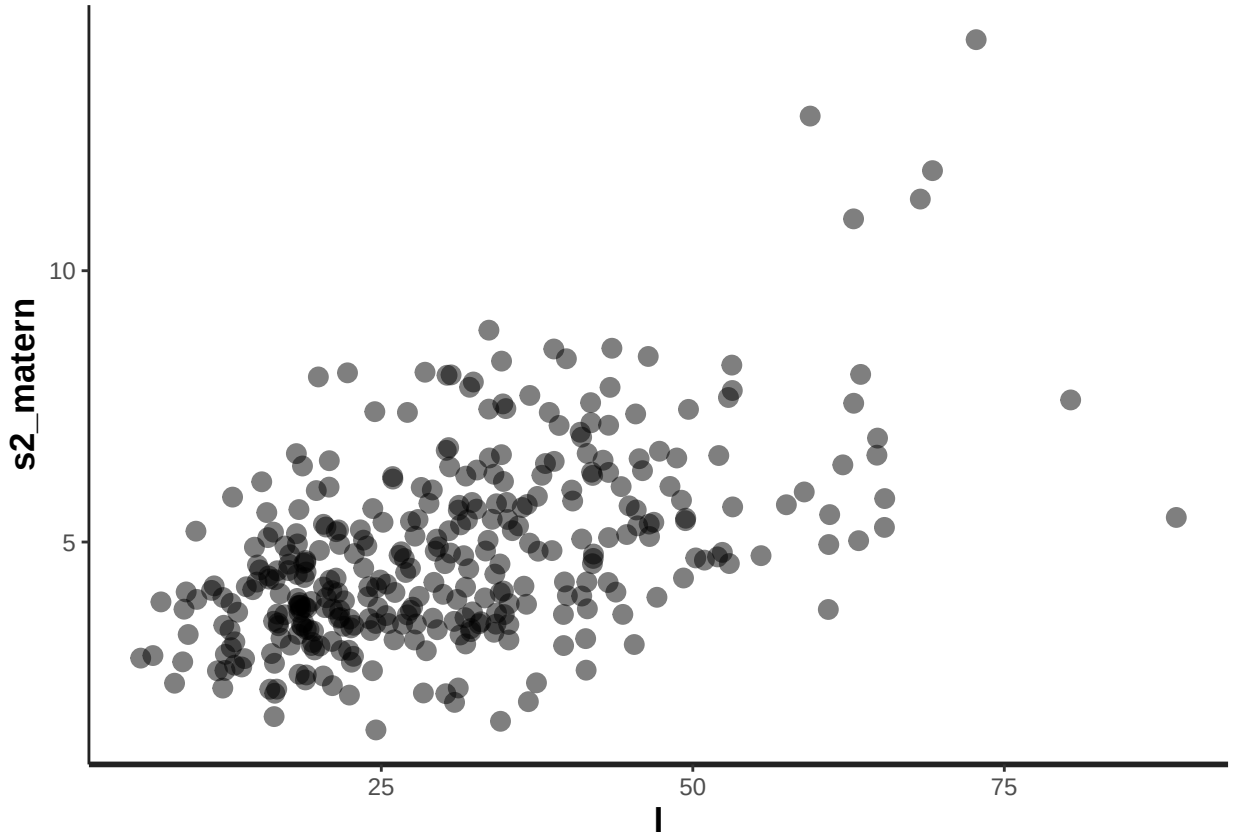
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit.w.q, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```r
# scatter plot of parameters of spatial random effect
stan_scat(fit.w.q, pars = c("l", "s2_matern"), color = "black", size = 3)
```



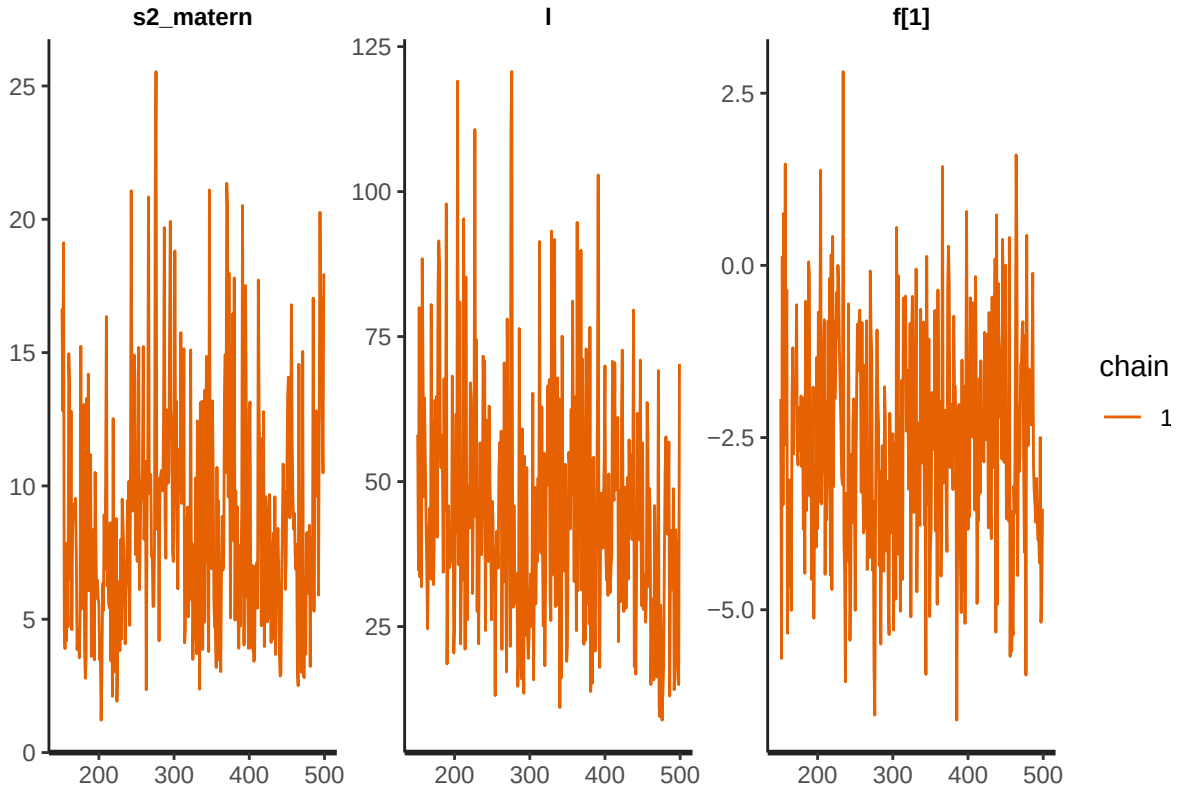
Vendace

```r
ven_data_q <-  list(Dx = ncol(x_q),
                    N = nrow(x_q),
                    Ds = ncol(s),
                    x = x_q,
                    s = s,
                    y = y.ven,
                    V = V)
# fit.v.q = stan(model_code =  GP_whitefishmodel_ire, data = ven_data_q, warmup=150, iter = 500, chains
#            #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
#            control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
# saveRDS(fit.v.q, file = "fit_v2.rds")
fit.v.q = readRDS( "fit_v2.rds")
m.vq = as.matrix(fit.v.q)


#print(fit.v.q)              # Rhat
summary(fit.v.q, pars = c("s2_matern", "l", "f[1]"))            # summary
```

```
## $summary
##                mean   se_mean        sd      2.5%       25%       50%       75%
## s2_matern  8.803954 0.4583028  4.351137  2.891335  5.568389  8.261277 10.783483
## l         44.684916 1.5392211 20.041679 14.073994 30.957053 42.277954 55.209516
## f[1]      -2.545171 0.1045828  1.642386 -5.528178 -3.695770 -2.625752 -1.392864
```

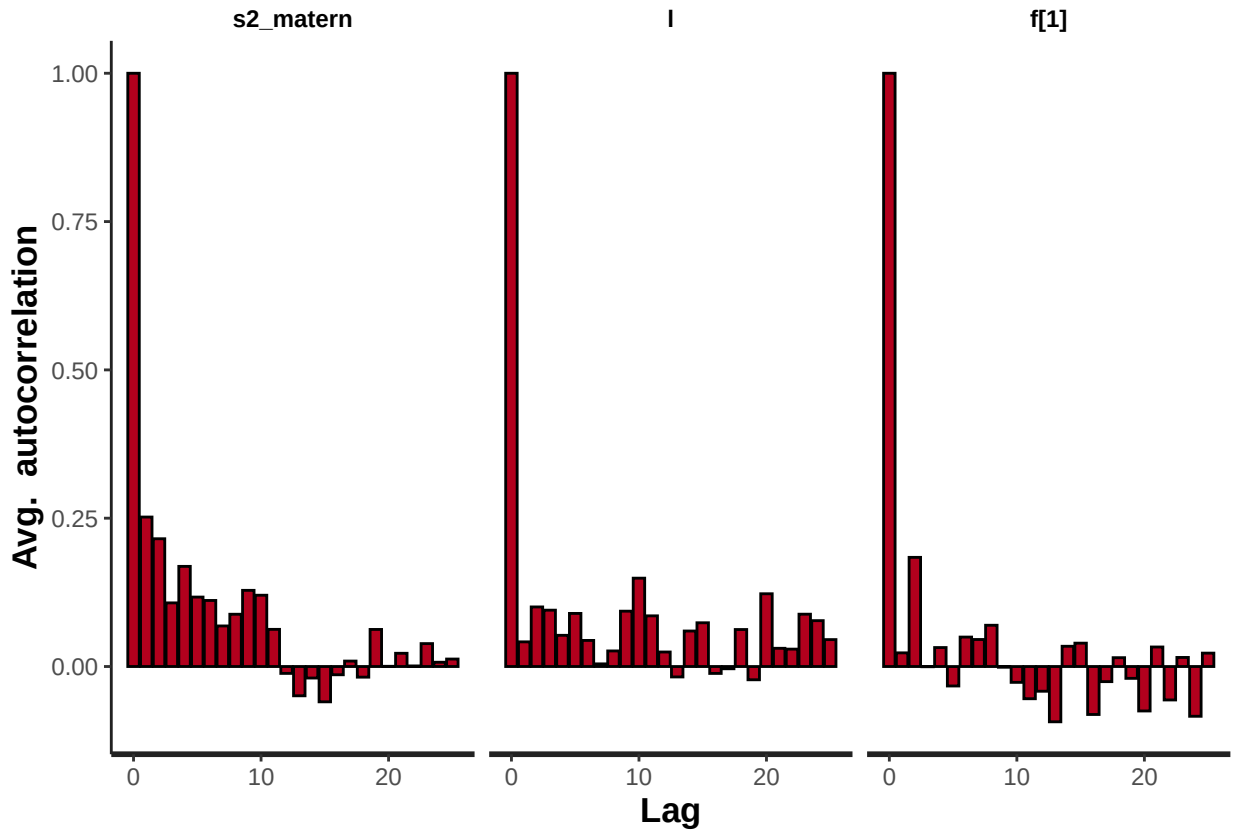
```
##                97.5%     n_eff      Rhat
## s2_matern 20.0166992  90.13646 0.9971522
## l         92.1669503 169.53766 1.0114810
## f[1]       0.4678788 246.62079 0.9980325
##
## $c_summary
## , , chains = chain:1
##
##            stats
## parameter        mean        sd      2.5%       25%       50%       75%
##   s2_matern  8.803954  4.351137  2.891335  5.568389  8.261277 10.783483
##   l         44.684916 20.041679 14.073994 30.957053 42.277954 55.209516
##   f[1]      -2.545171  1.642386 -5.528178 -3.695770 -2.625752 -1.392864
##            stats
## parameter        97.5%
##   s2_matern 20.0166992
##   l         92.1669503
##   f[1]       0.4678788
```

```
stan_trace(fit.v.q, pars = c("s2_matern", "l", "f[1]"))          # traceplot
```

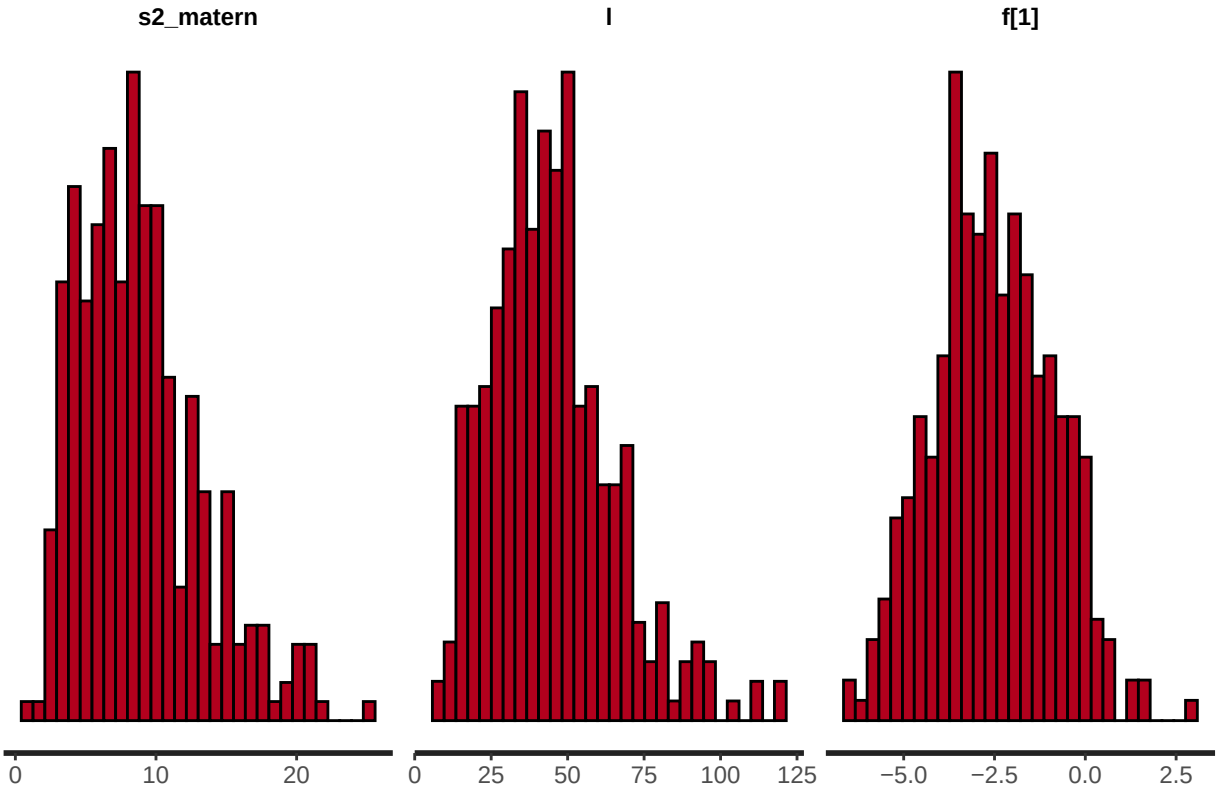



```
stan_ac(fit.v.q,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))    # autocorrelation
```

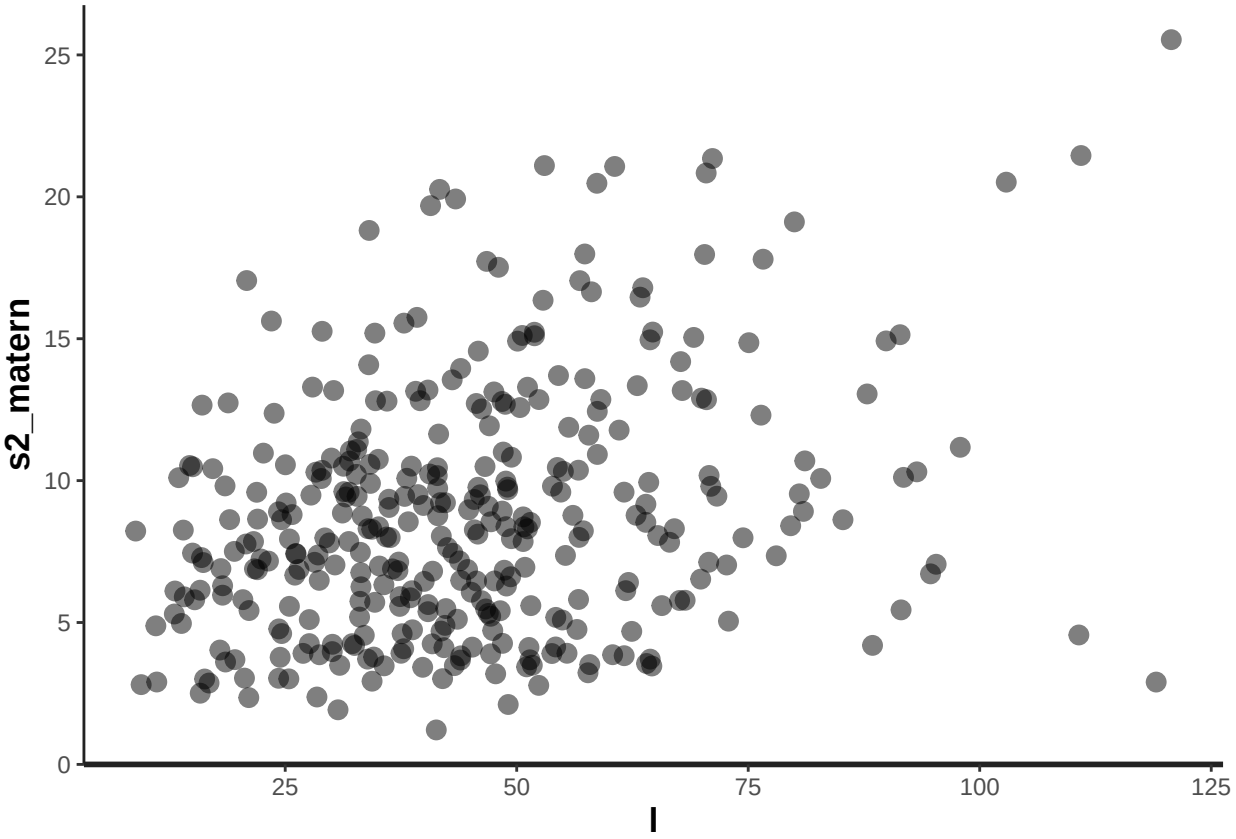
```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```




```
quietgg(stan_hist(fit.v.q, pars = c("s2_matern", "l", "f[1]")))   # posterior density
```



```r
# scatter plot of parameters of spatial random effect
stan_scat(fit.v.q, pars = c("l", "s2_matern"), color = "black", size = 3)
```



And do spatial predictions

```r
# Prediction variables
spred = as.matrix(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89)) / 1000  # spatial coordin
xpred.q = matrix(0,nrow=nrow(spred),ncol=22)      # intercept + 6 BOTTOMCLS classes + 5 continues covar
xpred.q[,1] = 1                                # Set the column corresponding to intercept to 1
xpred.q[whitefish.raster$BOTTOMCLS==1,2] = 1   # Set the elements corresponding to BOTTOMCLS = 1 to 1
xpred.q[whitefish.raster$BOTTOMCLS==2,3] = 1   # Set the elements corresponding to BOTTOMCLS = 2 to 1
xpred.q[whitefish.raster$BOTTOMCLS==3,4] = 1   # Set the elements corresponding to BOTTOMCLS = 3 to 1
xpred.q[whitefish.raster$BOTTOMCLS==4,5] = 1   # Set the elements corresponding to BOTTOMCLS = 4 to 1
xpred.q[whitefish.raster$BOTTOMCLS==5,6] = 1   # Set the elements corresponding to BOTTOMCLS = 5 to 1
xpredcont.q = as.matrix(cbind(whitefish.raster$DIS_SAND,
                              whitefish.raster$FE300ME,
                              whitefish.raster$ICELAST09,
                              whitefish.raster$RIVERS,
                              whitefish.raster$DIST20M,
                              whitefish.raster$CHL_A,
                              whitefish.raster$TEMP09M,
                              whitefish.raster$SALT09M,
                              whitefish.raster$DIS_SAND^2,
                              whitefish.raster$FE300ME^2,
                              whitefish.raster$ICELAST09^2,
                              whitefish.raster$RIVERS^2,
                              whitefish.raster$DIST20M^2,
                              whitefish.raster$CHL_A^2,
```

```
                            whitefish.raster$TEMP09M^2,
                            whitefish.raster$SALT09M^2))

stdxcont.q = apply(x_q[,7:22], 2, sd)
mxcont.q = apply(x_q[,7:22], 2, mean)
xpred.q[,7:22] = t( apply( t(apply(xpredcont.q,1,'-',mxcont.q)),1,'/',stdxcont.q) )    # "standardize"
```

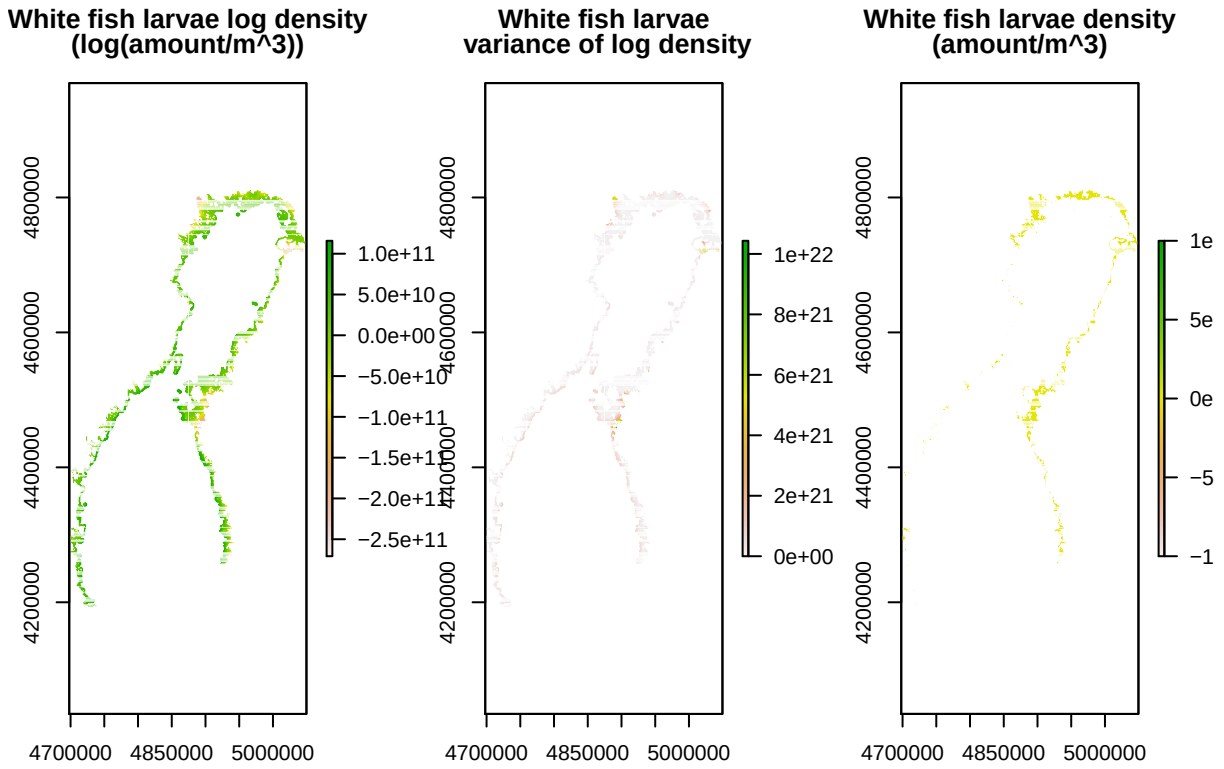
Whitefish predictions

```
#p.wq= prediction(m.wq, expCovariance, x_q, xpred.q , s, spred)
#saveRDS(p.wq, file="pred_w2.RData")
p.wq=readRDS("pred_w2.RData")

# linear coefficients(weights): beta estimates
p.wq$summary_beta
```

```
##                      Mean Stand_Dev    quant2.5% quant97.5%
## BOTTOMCLS_0 -17.6727260 4.5926044 -26.45443944 -9.5411006
## BOTTOMCLS_1   2.6110923 0.6355201   1.54184709  3.7942640
## BOTTOMCLS_2   3.4630910 1.0174538   1.99215844  5.3832906
## BOTTOMCLS_3  -2.8424631 2.1700794  -7.95172904 -0.2235535
## BOTTOMCLS_4   2.1454865 0.5386166   1.40561491  2.9397206
## BOTTOMCLS_5   2.1907872 0.4456723   1.42206691  3.1486821
## FE300ME     -18.3441299 5.4109261 -27.86737114 -9.0734003
## DIS_SAND     -9.3926399 4.3449506 -18.03297119 -2.5538301
## ICELAST09     4.5599798 1.2352904   2.57257575  7.6178335
## RIVERS       -2.1094822 0.8658796  -3.60848553 -0.8529943
## DIST20M       1.5020315 1.3331487  -0.14321767  4.1694152
## CHL_A        -1.7834004 1.5855775  -4.45234565  0.7701687
## TEMP09M       3.8958107 2.1131968   0.85143568  7.5274890
## SALT09M       6.7043679 2.6224031   1.53109924 10.8417073
## FE300ME^2     1.5047539 0.9057006  -0.21475017  3.3709709
## DIS_SAND^2   11.1875007 3.4206163   5.54950442 19.8302604
## ICELAST09^2   1.6325350 0.3765243   1.05198219  2.4692588
## RIVERS^2     -1.1615589 0.2884805  -1.74635883 -0.5708109
## DIST20M^2    -0.9478972 0.3597799  -1.67384070 -0.3999620
## CHL_A^2       1.8414325 1.0292233  -0.04670973  3.7574991
## TEMP09M^2    -1.1222172 0.5032864  -2.10302850 -0.1245565
## SALT09M^2     1.1142788 1.3528226  -1.25652214  3.6150412
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.wq$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.wq$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p.wq$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
```

```
    ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

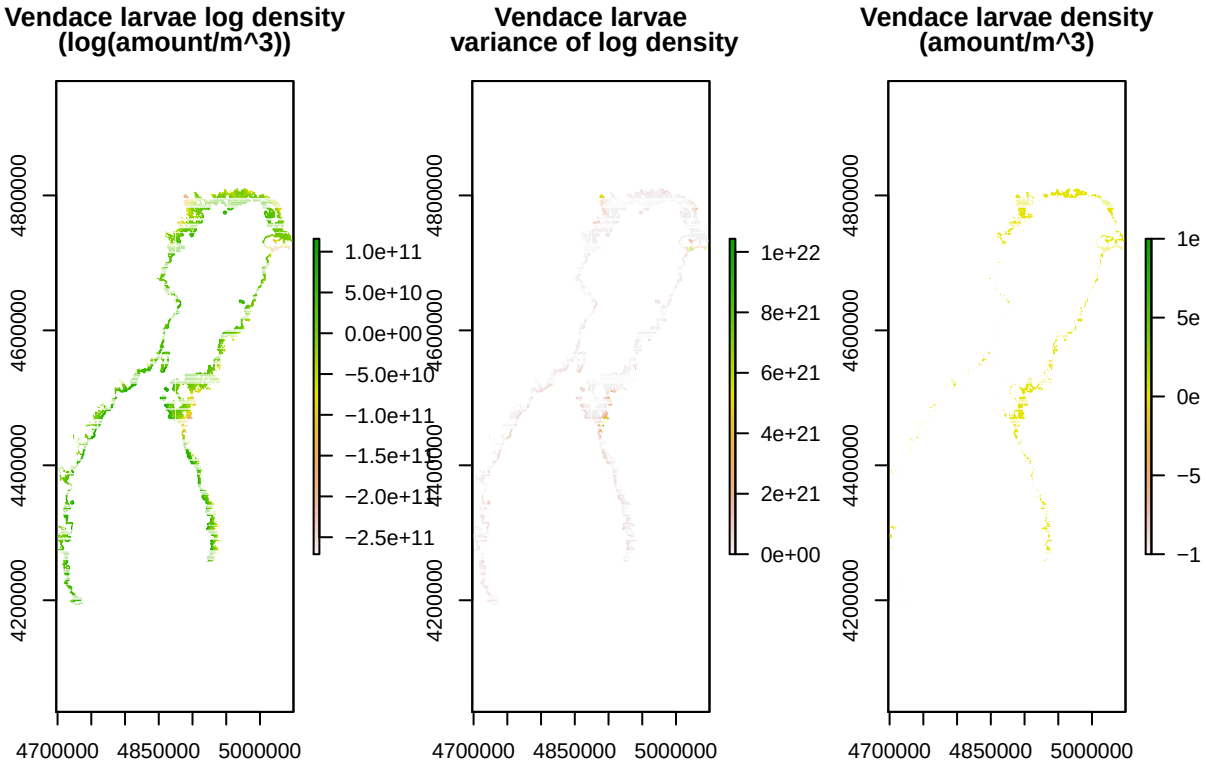
Vendace predictions

```
# p.vq= prediction(m.wq, expCovariance, x_q, xpred.q , s, spred)
#saveRDS(p.vq, file="pred_v2.RData")
p.vq=readRDS("pred_v2.RData")

# linear coefficients(weights): beta estimates
p.vq$summary_beta
```

```
##                     Mean Stand_Dev    quant2.5%    quant97.5%
## BOTTOMCLS_0 -17.4827646 4.4771977 -25.98527851 -9.5556953891
## BOTTOMCLS_1   2.6512299 0.6409622   1.69543025  3.7406179348
## BOTTOMCLS_2   3.4680578 1.0190128   1.89441446  5.2907666213
## BOTTOMCLS_3  -2.8643134 2.1883018  -8.05052750  0.0467084551
## BOTTOMCLS_4   2.1313659 0.5670753   1.33790523  2.8432515275
## BOTTOMCLS_5   2.2219159 0.4503197   1.36575959  3.1198081191
## FE300ME     -18.1776371 5.2912467 -27.57496132 -8.4046590472
## DIS_SAND     -9.3892015 4.3395136 -18.01401636 -2.5018321563
## ICELAST09     4.5219397 1.2661031   2.35151060  7.5851719227
## RIVERS       -2.0841047 0.8304721  -3.66595073 -0.8261130556
## DIST20M       1.5057187 1.2887184  -0.09056594  3.9909758549
## CHL_A        -1.7542949 1.7014464  -5.79581146  0.9400190815
## TEMP09M       3.9583358 2.0012270   0.61011548  7.4837563969
## SALT09M       6.6773175 2.6092337   1.46917861 10.1706765343
## FE300ME^2     1.4691001 0.8621986  -0.37002887  2.9871389056
## DIS_SAND^2   11.1883497 3.4427822   5.53570934 19.8491730657
```

```
## ICELAST09^2   1.6250658 0.3917744   1.00117803  2.5255889847
## RIVERS^2     -1.1342746 0.2732657  -1.61775319 -0.5833014096
## DIST20M^2    -0.9575645 0.3650242  -1.62226342 -0.4334263477
## CHL_A^2       1.8239810 1.0013824   0.03448317  3.3774124858
## TEMP09M^2    -1.0942654 0.5645200  -2.25144382 -0.0009094161
## SALT09M^2     1.1145876 1.3784182  -1.30872672  3.6098882679
```

```r
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.vq$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae l
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.vq$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae"
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p.vq$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae d
```




```r
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

Cross validation

Let's compare the two SDMs: the linear model and the one with additional quadratic effect. We use cross validation method: we calculate the k-fold-CV estimate, that is

$$\mathrm{CV} = \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, D_{\setminus k(i)}) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{1}{M} \log p(y_i|x_i, \theta^s)$$

where $\theta^s \sim p(\theta|D_{\setminus k(i)})$ and $D_{\setminus k(i)}$ denotes the data from where the block including the data point i is excluded. We set $k = 10$.

Kfold : computing neg.binomial lpmf for prediction set as a genereted quantity in the stan model.. problems.

```
GP_whitefishmodel_cv = "
  data {
    int<lower=1> N;   // training data
    int<lower=1> Dx;
    matrix[N,Dx] x;
    int<lower=1> Ds;
    matrix[N,Ds] s;
    int<lower=0> y[N];
    vector[N] V;
    int<lower=1> Np;  // test data
    matrix[Np,Dx] xp;
    matrix[Np,Ds] sp;
    int<lower=0> yp[Np];
    vector[Np] Vp;
  }
  transformed data {
    vector[N] mu;
    matrix[N, N] Dist_spatial;
    matrix[N, N] Sigma_lin;
    vector[Np] mup;
    matrix[Np, Np] Dist_spatialp;
    matrix[Np, Np] Sigma_linp;
    real s2_lin;
    s2_lin = 10;

    for (i in 1:N)
      mu[i] = 0;
    for (i in 1:Np)
      mup[i] = 0;

    // off-diagonal elements
    for (i in 1:(N-1)) {
      for (j in (i+1):N) {
        Dist_spatial[i, j] = pow(dot_self(s[i] - s[j]),0.5)  ;
        Sigma_lin[i, j] = s2_lin * dot_product(x[i],x[j]);     // linear covariance function

        // Fill in the other half
        Dist_spatial[j, i] = Dist_spatial[i, j];
        Sigma_lin[j, i] = Sigma_lin[i, j];
      }
    }
    // diagonal elements
```

```
  for (k in 1:N){
    Dist_spatial[k, k] = 0;
    Sigma_lin[k, k] = s2_lin * dot_product(x[k],x[k]) + 1e-6;   // add also some jitter
  }

      // off-diagonal elements
  for (i in 1:(Np-1)) {
    for (j in (i+1):Np) {
      Dist_spatialp[i, j] = pow(dot_self(sp[i] - sp[j]),0.5)  ;
      Sigma_linp[i, j] = s2_lin * dot_product(xp[i],xp[j]);     // linear covariance function

      // Fill in the other half
      Dist_spatialp[j, i] = Dist_spatialp[i, j];
      Sigma_linp[j, i] = Sigma_linp[i, j];
    }
  }
  // diagonal elements
  for (k in 1:Np){
    Dist_spatialp[k, k] = 0;
    Sigma_linp[k, k] = s2_lin * dot_product(xp[k],xp[k]) + 1e-6;   // add also some jitter
  }
}
parameters {
  real<lower=0> l;
  real<lower=0> s2_matern;
  vector[N] z;
    vector[Np] zp;
  real<lower=0> r;
}
transformed parameters {
  matrix[N, N] Sigma;
  matrix[N, N] L;
  matrix[Np, Np] Sigmap;
  matrix[Np, Np] Lp;
  real<lower=0> inv_l;


  inv_l = inv(l);
Sigma = s2_matern*exp(-inv_l*Dist_spatial ) + Sigma_lin; // Exponential
Sigmap = s2_matern*exp(-inv_l*Dist_spatialp ) + Sigma_linp; // Exponential

 //     Sigma = s2_matern*(1 + pow(3,0.5)*inv_l*Dist_spatial).*exp(-pow(3,0.5)*inv_l*Dist_spatial)  +
  L = cholesky_decompose(Sigma);
  Lp = cholesky_decompose(Sigmap);
}
model {
  vector[N] ff;
  vector[Np] ffp;

  // A weakly informative prior for magnitude
  s2_matern ~ student_t(4, 0, 1);

  // A weakly informative prior for l, that shrinks to 0 cor. among locations with more than 50km dis
```

```
    l ~ gamma(7, 0.1);

// A weakly informative prior for
    r ~ gamma(2, 0.1 );

    z ~ normal(0, 1);
    zp ~ normal(0, 1);
    ff = L*z;
    ffp = Lp*zp;

    for (n in 1:N) {
      y[n] ~ neg_binomial_2(V[n]*exp(ff[n]), r);
    }


  }
  generated quantities {
    vector[N] f;
    vector[Np] fp;
    vector[Np] Lik;

    // derived quantity (transform)
    f = L*z;
    fp = Lp*zp;
    //for(i in 1:Np) Lik[i] = neg_binomial_2_lpmf(yp[i] |Vp[i]*exp(fp[i]),r);

  }"


#stanStruct = stan_model(model_code = GP_whitefishmodel_cv)
```

```
# | K-fold Cross-Validation
cv= function(data, stanStruct,  K=10){
  # shuffle the data so that cross validation is done with random divisions

  rows = sample(nrow(data))
  df = data[rows,]
  x = as.matrix(subset(df, select=-c(y,V,s1,s2)))  # target first place
  y = df$y
  V = df$V
  s =cbind(df$s1, df$s2)

  B = ceiling(length(y)/K)  # the size of each fold
  n = length(y)
  logLikest = c();

  for (i in 1:K) {
    ind1 = (B*(i-1)+1):min(n, i*B)    # indexes of test data
    ind2 = which(y != y[ind1]);  # indexes of training data
    # dataset for the ith fold
    dataset = list(Dx = ncol(x), Ds = ncol(s),
                          N = length(ind2),
                          x = x[ind2,],
                          s = s[ind2,],
```

```
                        y = y[ind2],
                        V = V[ind2],
                        Np = length(ind1),
                        xp = x[ind1,],
                        sp = s[ind1,],
                        yp = y[ind1],
                        Vp = V[ind1])

    # posterior samples
    post = sampling(stanStruct, data = dataset, chains = 1, warmup = 100, thin = 1, iter = 600)
    M = extract(post)
    # log point-wise posterior predictive density estimates
    logLikest[ind1] = log(colMeans(M$Lik))
  }

  KfoldCV = mean(logLikest)
  return(KfoldCV)
}
```

Try a different approach: use the previous stan function for training data (ind2) and previous prediction function for test data (ind1).

```
cv1 = function(data, stanmodel, K=5) {
 rows = sample(nrow(data))
  df = data[rows,]
  x = as.matrix(subset(df, select=-c(y,V,s1,s2)))  # target first place
  y = df$y
  V = df$V
  s =cbind(df$s1, df$s2)

  B = ceiling(length(y)/K)  # the size of each fold
  n = length(y)
  logLikest = c();

  for (i in 1:K) {
    ind1 = (B*(i-1)+1):min(n, i*B)    # indexes of test data
    ind2 = which(y != y[ind1]);  # indexes of training data
    # dataset for the ith fold
    dataset = list(Dx = ncol(x),
                   Ds = ncol(s),
                   N = nrow(x[ind2,]),
                   x = x[ind2,],
                   s = s[ind2,],
                   y = y[ind2],
                   V = V[ind2])


    # posterior samples
    post = stan(model_code =stanmodel, data = dataset, chains = 1, warmup = 150, thin = 1, iter = 500,
                    control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l", "r"))
    r = as.matrix(post, pars="r")
    m = as.matrix(post)
    # log point-wise posterior predictive density estimates
    Np = length(ind1)
```

```
    xp = x[ind1,]
    sp = s[ind1,]
    yp = y[ind1]
    Vp = V[ind1]
    p = prediction(m, expCovariance, x[ind2,], xp , s[ind2,], sp, thin=1)

    lik.yp=matrix(nrow=length(r), ncol=length(ind1)) #row=MC iter, col=pred obs
    for (j in 1:length(r)) {
      lik.yp[j,] = dnbinom(yp, mu = Vp*exp(p$EfMC[,j]), size =r[j])
    }
    logLikest[ind1] = log(colMeans(lik.yp))
    print(logLikest)
  }
  logLikest[!is.finite(logLikest)] = -200
  KfoldCV = mean(logLikest, na.rm =T)
  return(KfoldCV)
}
```

```
y= whitefish.dat$WHISUM[-vol0]
#wf 1degree poly
data = data.frame(y, x,V,s1=s[,1], s2=s[,2])
Kfoldcv.wf =-7.081072 #cv1(data, GP_whitefishmodel_ire) #

#ven 1degree poly
data = data.frame(y=y.ven, x,V, s1=s[,1], s2=s[,2])
Kfoldcv.ven = -2.984913 #cv1(data, GP_whitefishmodel_ire)  #    (-4.460242 )

#wf 2degree poly
data = data.frame(y, x_q,V,s1=s[,1], s2=s[,2])
Kfoldcv.wf2 =-4.987135 #cv1(data, GP_whitefishmodel_ire)  #

#ven 2degree poly
data = data.frame(y=y.ven, x_q,V,s1=s[,1], s2=s[,2])
Kfoldcv.ven2 =-5.545475 #cv1(data, GP_whitefishmodel_ire) #

cv.df = data.frame(model=c("whitefish 1poly","vendace 1poly","whitefish 2poly","vendace 2poly"), Kfoldcv
cv.df
```

```
##             model   Kfoldcv
## 1 whitefish 1poly -7.081072
## 2   vendace 1poly -2.984913
## 3 whitefish 2poly -4.987135
## 4   vendace 2poly -5.545475
```

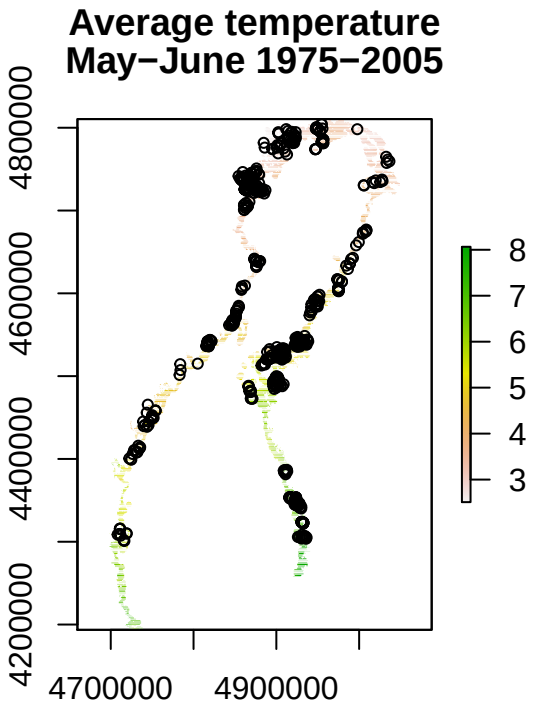
Future predictions

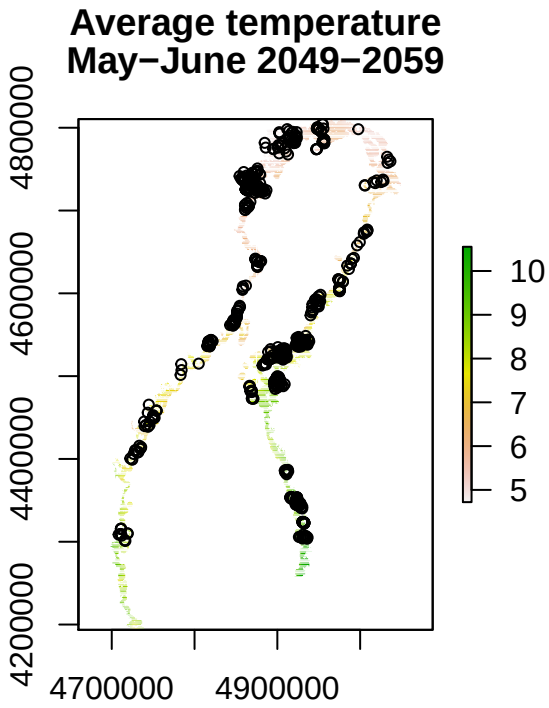
After comparing the models we do predictions for future years. We assume the only variables which will change significantly are salinity, temperature (last ice coverage, rivers). The values for future covariates have been recovered by mean of specific models, implemented in the SmartSea project. We consider as representative of future situation the mean value of the last available decade (2049-2059), predicted in April -June.

```
whitefish.raster = read.table("white_fishes_final_raster_fut.txt", header=TRUE)
whitefish.raster.past = read.table("white_fishes_final_raster.txt", header=TRUE)
```

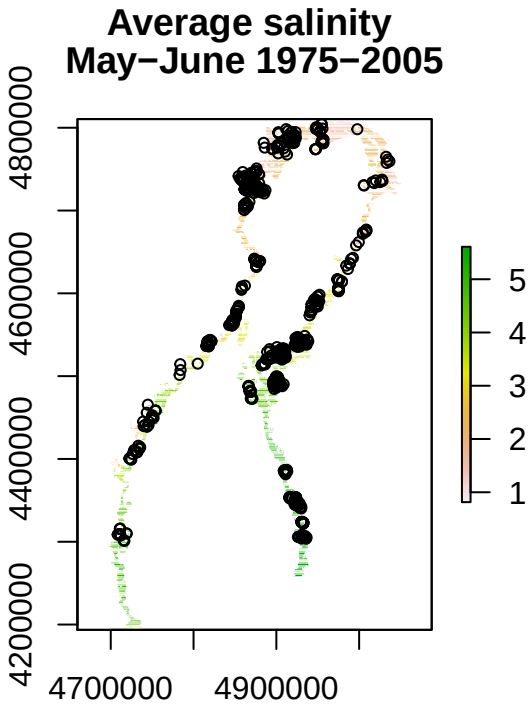
We now compare the environmental covariates for future years, and the one from past years, by plotting the raster layers. The temperature has an average increase of 2 degreres, while salinity generally decreases a bit, but its variation is less evident than the temperature gradient.

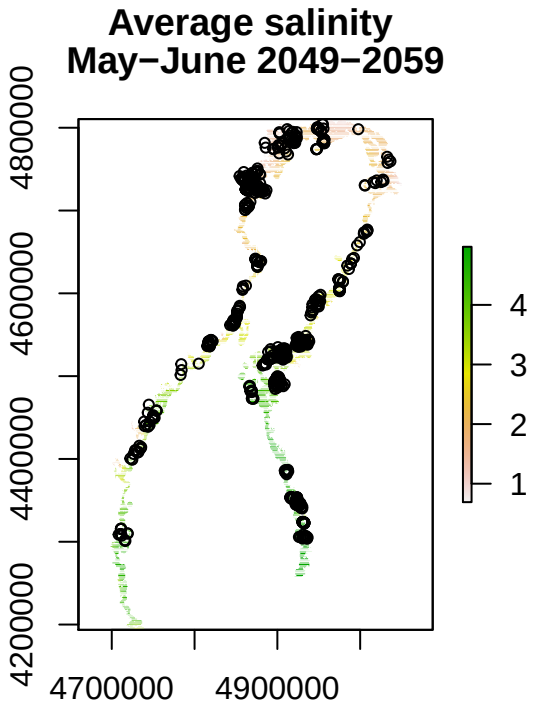
```
# Visualize few environmental covariates
e <- extent(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89))
r <- raster(e, ncol=length(unique(whitefish.raster$E_etrs89)), nrow=length(unique(whitefish.raster$N_et
# Visualize the study area

par(mfrow=c(1,2))

# temperature
z <- rasterize(cbind(whitefish.raster.past$E_etrs89,whitefish.raster.past$N_etrs89), r, whitefish.raste
plot(z, xlim=cbind(min(whitefish.raster.past$E_etrs89),max(whitefish.raster.past$E_etrs89)),
     ylim=cbind(min(whitefish.raster.past$N_etrs89),max(whitefish.raster.past$N_etrs89)), main=c("Averag
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)

z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$TEMP09M_
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average tempera
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
```



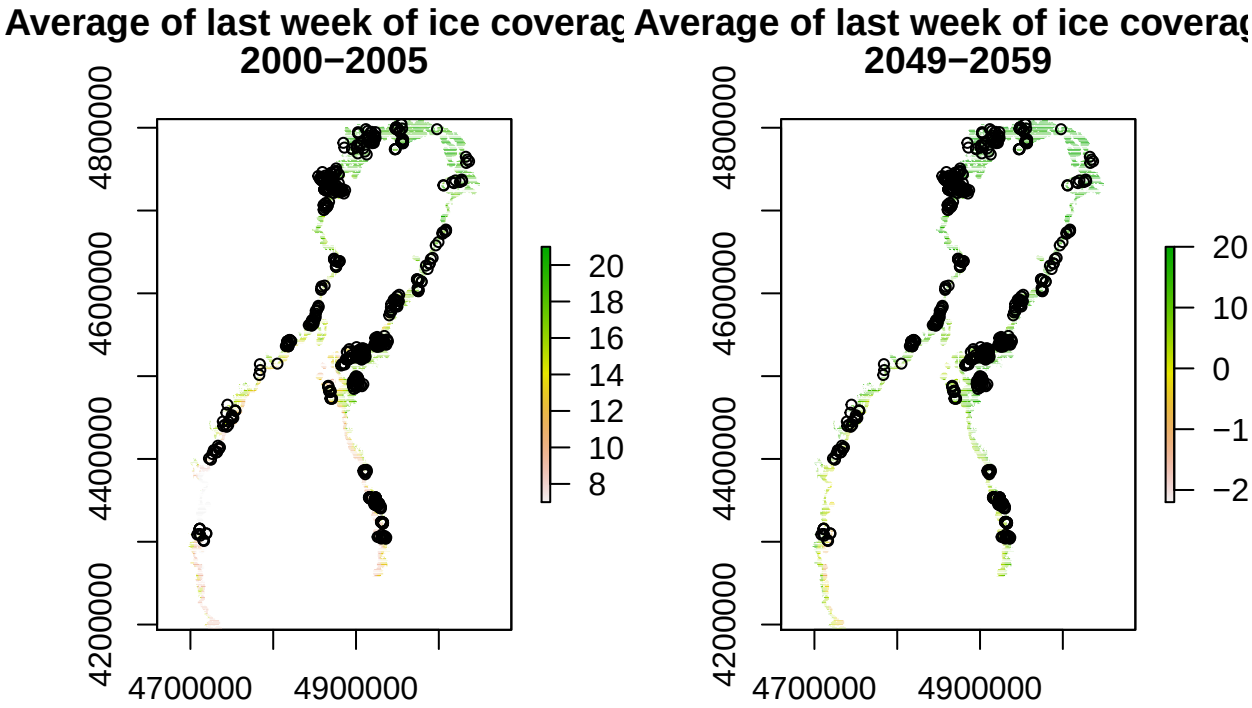



```
# salinity
z <- rasterize(cbind(whitefish.raster.past$E_etrs89,whitefish.raster.past$N_etrs89), r, whitefish.raste
plot(z, xlim=cbind(min(whitefish.raster.past$E_etrs89),max(whitefish.raster.past$E_etrs89)),
     ylim=cbind(min(whitefish.raster.past$N_etrs89),max(whitefish.raster.past$N_etrs89)), main=c("Avera
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)

z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$SALT09M_
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average salinit
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
```






```
# icelast
z <- rasterize(cbind(whitefish.raster.past$E_etrs89,whitefish.raster.past$N_etrs89), r, whitefish.raste
plot(z, xlim=cbind(min(whitefish.raster.past$E_etrs89),max(whitefish.raster.past$E_etrs89)),
     ylim=cbind(min(whitefish.raster.past$N_etrs89),max(whitefish.raster.past$N_etrs89)), main=c("Avera
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)

z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, whitefish.raster$ICELAST_
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average of last
# Plot the locations of sampling sites
points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89, cex=0.7)
```



We do predictions on larvae densities of the two species, by using as future covariates the new dataset, with future values for temperature and salinity. We use the model with quadratic effect which turns out to be the best one in terms of prediction accuracy, with exponential covariance function.

```
# Prediction variables
spred = as.matrix(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89)) / 1000  # spatial coordin
xpred.q = matrix(0,nrow=nrow(spred),ncol=22)      # intercept + 6 BOTTOMCLS classes + 5 continues covar
xpred.q[,1] = 1                                 # Set the column corresponding to intercept to 1
xpred.q[whitefish.raster$BOTTOMCLS==1,2] = 1   # Set the elements corresponding to BOTTOMCLS = 1 to 1
xpred.q[whitefish.raster$BOTTOMCLS==2,3] = 1   # Set the elements corresponding to BOTTOMCLS = 2 to 1
xpred.q[whitefish.raster$BOTTOMCLS==3,4] = 1   # Set the elements corresponding to BOTTOMCLS = 3 to 1
xpred.q[whitefish.raster$BOTTOMCLS==4,5] = 1   # Set the elements corresponding to BOTTOMCLS = 4 to 1
xpred.q[whitefish.raster$BOTTOMCLS==5,6] = 1   # Set the elements corresponding to BOTTOMCLS = 5 to 1
xpredcont.q = as.matrix(cbind(whitefish.raster$DIS_SAND,
                              whitefish.raster$FE300ME,
                              whitefish.raster$ICELAST_FUT,
                              whitefish.raster$RIVERS,
                              whitefish.raster$DIST20M,
                              whitefish.raster$CHL_A,
                              whitefish.raster$TEMP09M_FUT,
                              whitefish.raster$SALT09M_FUT,
                              whitefish.raster$DIS_SAND^2,
                              whitefish.raster$FE300ME^2,
                              whitefish.raster$ICELAST_FUT^2,
                              whitefish.raster$RIVERS^2,
                              whitefish.raster$DIST20M^2,
                              whitefish.raster$CHL_A^2,
                              whitefish.raster$TEMP09M_FUT^2,
                              whitefish.raster$SALT09M_FUT^2))

stdxcont.q = apply(x_q[,7:22], 2, sd)
mxcont.q = apply(x_q[,7:22], 2, mean)
```

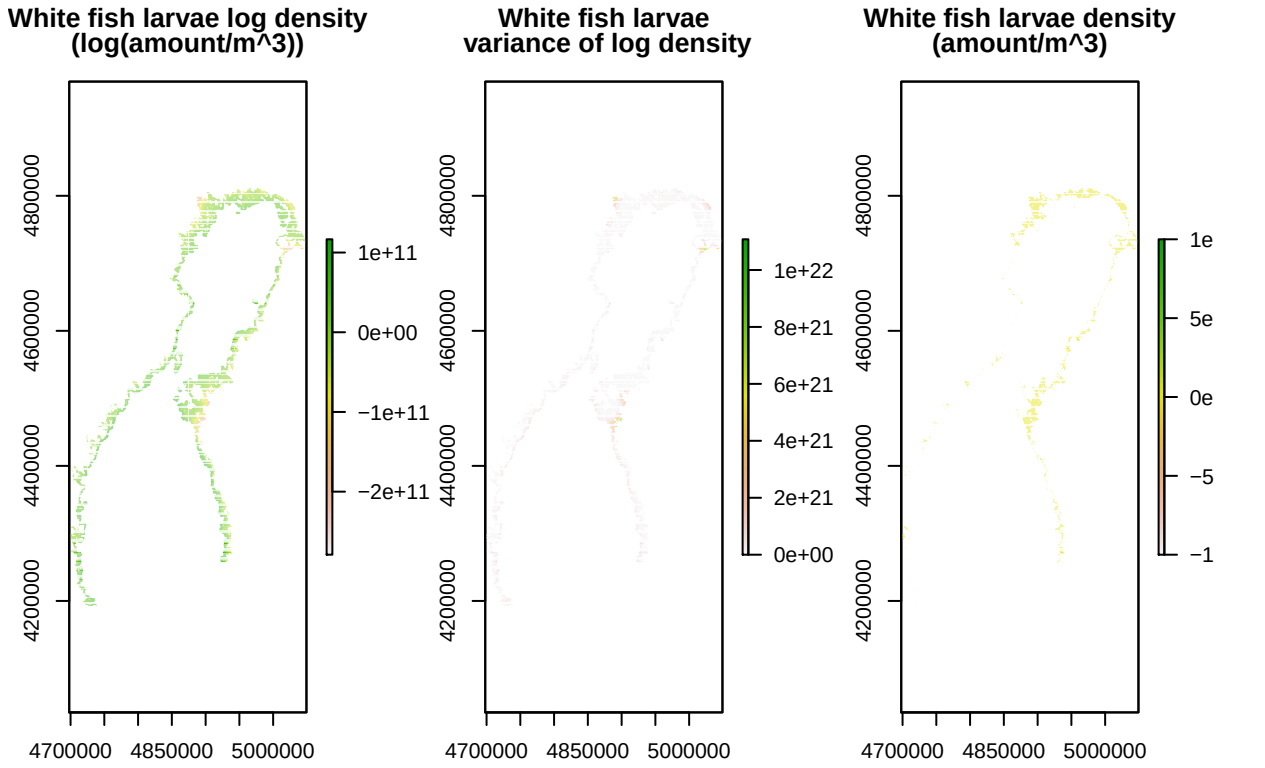
```
xpred.q[,7:22] = t( apply( t(apply(xpredcont.q,1,'-',mxcont.q)),1,'/',stdxcont.q) )    # "standardize"
```

Whitefish predictions

```
#p.wq.fut= prediction(m.wq, expCovariance, x_q, xpred.q , s, spred)
#saveRDS(p.wq.fut, file="pred_w2_fut.RData")
p.wq=readRDS("pred_w2_fut.RData")

# linear coefficients(weights): beta estimates
p.wq$summary_beta
```

```
##                     Mean Stand_Dev   quant2.5%  quant97.5%
## BOTTOMCLS_0 -17.6518200 4.7090685 -27.1313253 -9.65712201
## BOTTOMCLS_1   2.6224329 0.6166140   1.5773113  3.90898856
## BOTTOMCLS_2   3.4707125 0.9866796   1.8409806  5.21048860
## BOTTOMCLS_3  -2.8312872 2.1691481  -8.0345769  0.06107361
## BOTTOMCLS_4   2.1516502 0.5544885   1.3548164  3.05221898
## BOTTOMCLS_5   2.2026837 0.4708268   1.2913529  3.15074727
## FE300ME     -18.3260273 5.4528192 -27.6351145 -8.66543762
## DIS_SAND     -9.3887532 4.3482472 -17.9541819 -2.44248194
## ICELAST09     4.5869258 1.2061718   2.5312535  7.59048362
## RIVERS       -2.0978549 0.8595792  -3.7030306 -0.74632664
## DIST20M       1.5212732 1.3578440  -0.1149884  3.96513658
## CHL_A        -1.8446546 1.5969721  -5.2714937  0.27336894
## TEMP09M       3.8260605 2.2107027  -0.4707481  8.86172433
## SALT09M       6.8686244 2.6675187   2.1193324 10.44663218
## FE300ME^2     1.4886965 0.8865499  -0.1189024  3.00034601
## DIS_SAND^2   11.1848694 3.4285296   5.5733589 19.80458645
## ICELAST09^2   1.6322485 0.3831341   1.0531091  2.48341784
## RIVERS^2     -1.1556874 0.2748994  -1.7154198 -0.67010331
## DIST20M^2    -0.9507713 0.3450782  -1.6045915 -0.43677223
## CHL_A^2       1.8233986 1.0396921  -0.2023091  3.71475009
## TEMP09M^2    -1.0995164 0.5901903  -2.2588221 -0.05699018
## SALT09M^2     1.1325141 1.3893416  -1.3172574  3.30911869
```

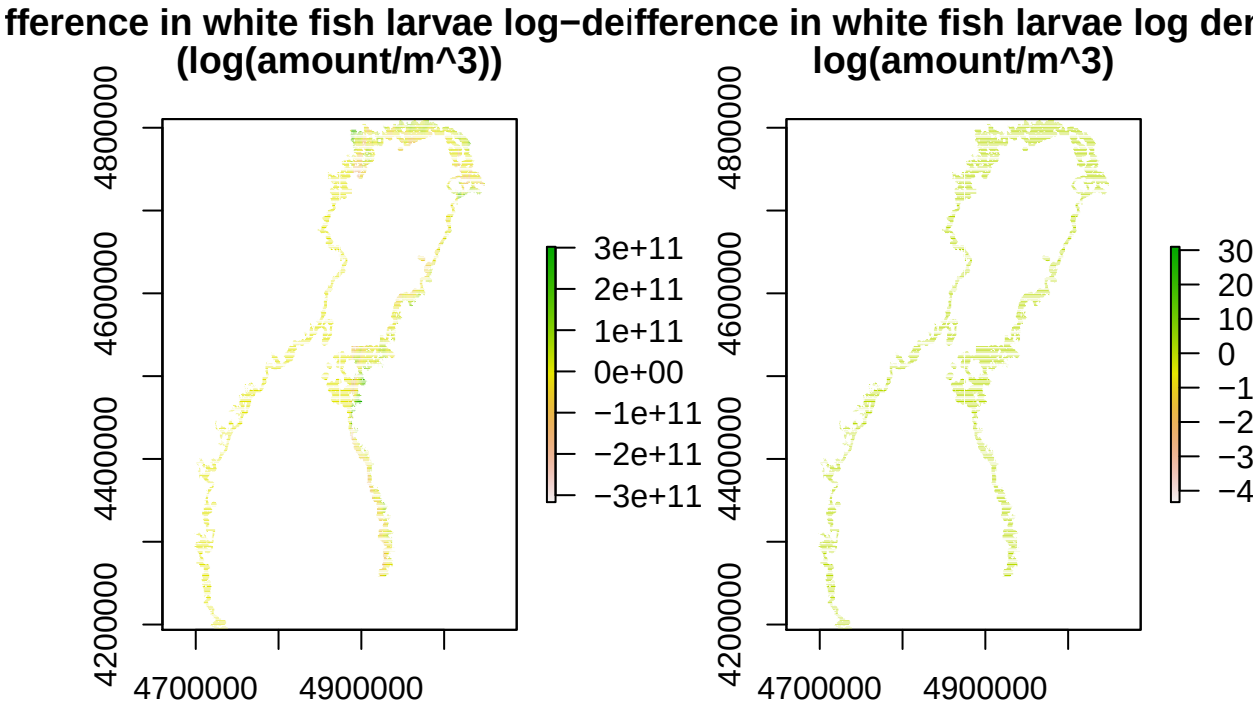
```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.wq$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.wq$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p.wq$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("White fish larva
```




```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

We compute and visualize the difference in larvae aboundance between past and future density

```
p.wq=readRDS("pred_w2.RData")
p.wq.fut=readRDS("pred_w2_fut.RData")

  par(mfrow=c(1,2))
# Posterior median of difference of past and future log density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.wq$Ef-p.wq.fut$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Difference in w
# Posterior median of difference of past and future density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.wq$Ef/p.wq.fut$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Difference in w
```

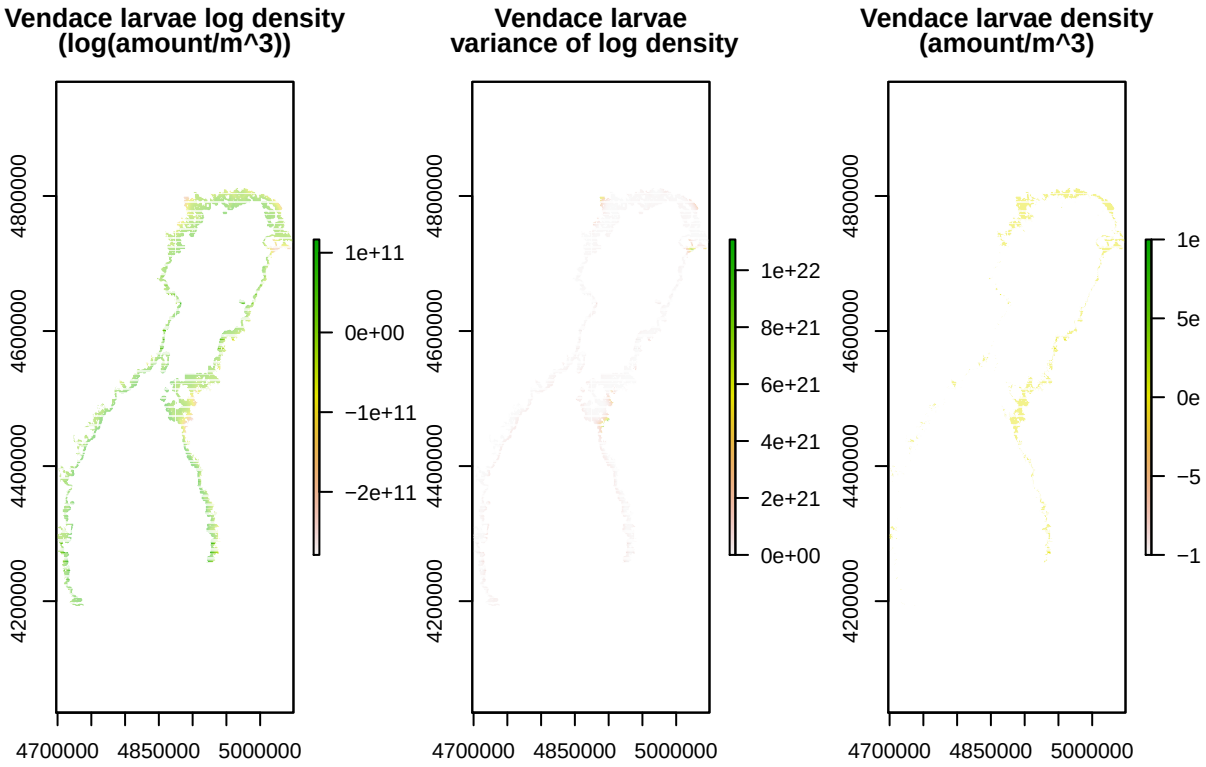


Vendace predictions

```
#p.vq.fut= prediction(m.wq, expCovariance, x_q, xpred.q , s, spred)
#saveRDS(p.vq.fut, file="pred_v2_fut.RData")
p.vq=readRDS("pred_v2_fut.RData")

# linear coefficients(weights): beta estimates
p.vq$summary_beta
```

```
##                    Mean Stand_Dev    quant2.5%  quant97.5%
## BOTTOMCLS_0 -17.5307996 4.4788223 -27.15071355 -9.99210152
## BOTTOMCLS_1   2.5927074 0.6534723   1.52368399  3.59760742
## BOTTOMCLS_2   3.4540003 1.0410919   1.96611033  5.23749263
## BOTTOMCLS_3  -2.8642281 2.2132708  -8.26690868 -0.13476303
## BOTTOMCLS_4   2.1424147 0.5587115   1.29318109  2.90512620
## BOTTOMCLS_5   2.2108177 0.4522977   1.43351749  3.18665488
## FE300ME     -18.3280977 5.4129517 -27.78630905 -8.72929555
## DIS_SAND     -9.3884697 4.3451259 -18.06468306 -2.51261994
## ICELAST09     4.5343120 1.2580653   2.25428366  7.68211097
## RIVERS       -2.0831593 0.8690825  -3.92239165 -0.87823248
## DIST20M       1.5253273 1.3199451  -0.10024641  4.09584202
## CHL_A        -1.7369064 1.6153312  -5.56335135  0.47570594
## TEMP09M       3.8730071 2.1526478   0.35372876  7.94177095
## SALT09M       6.6835097 2.6440422   1.91199449 10.29668028
## FE300ME^2     1.5124590 0.8801711  -0.40055674  2.92445746
## DIS_SAND^2   11.1854546 3.4252772   5.55169204 19.81586874
## ICELAST09^2   1.6331469 0.3896810   1.01682531  2.56763918
## RIVERS^2     -1.1581125 0.2691260  -1.70983230 -0.65664710
## DIST20M^2    -0.9551661 0.3513037  -1.59659776 -0.45158246
## CHL_A^2       1.8063654 0.9975724  -0.03664245  3.60820188
## TEMP09M^2    -1.1277086 0.5630382  -2.19752508 -0.07707782
## SALT09M^2     1.1430186 1.3729267  -1.43303268  3.77973396
```

```
# prediction of larvae density f:
par(mfrow=c(1,3))
# Posterior mean of f
  z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.vq$Ef)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae l
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior variance of f
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, p.vq$Varf)
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae"
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
# Posterior median of density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, exp(p.vq$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Vendace larvae d
```

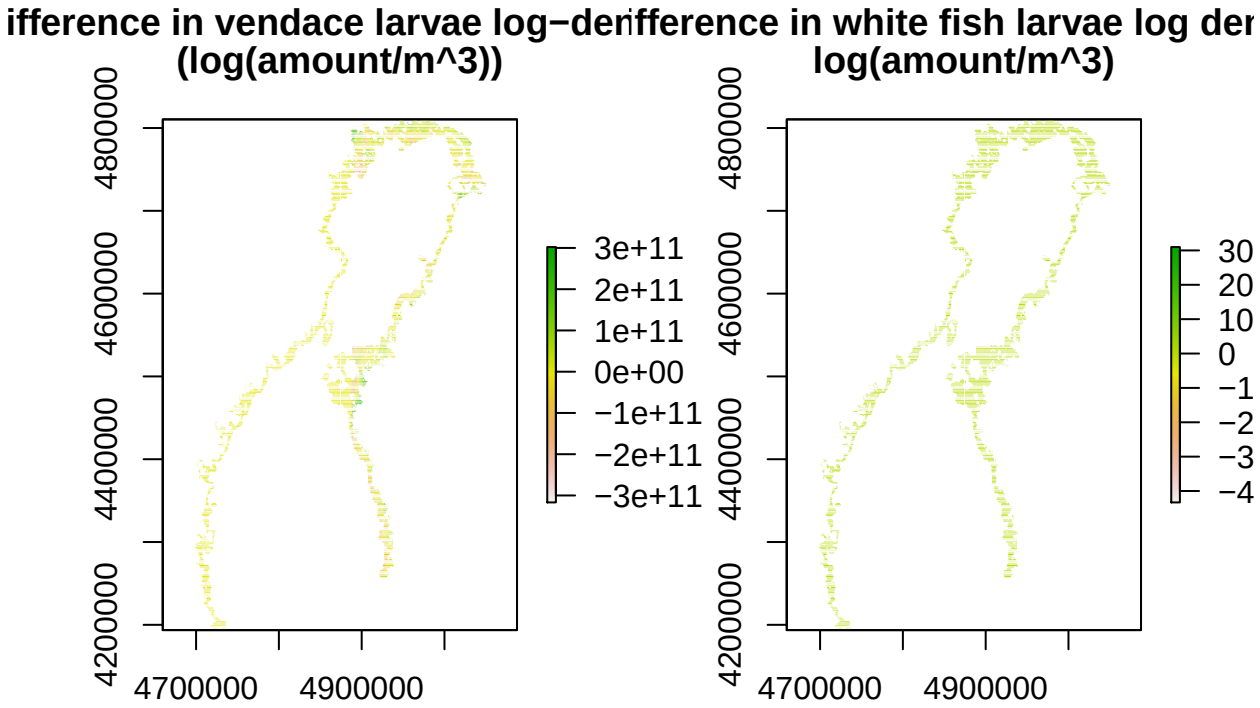



```
#points(whitefish.dat$E_etrs89,whitefish.dat$N_etrs89)
```

We compute and visualize the difference in larvae aboundance between past and future density

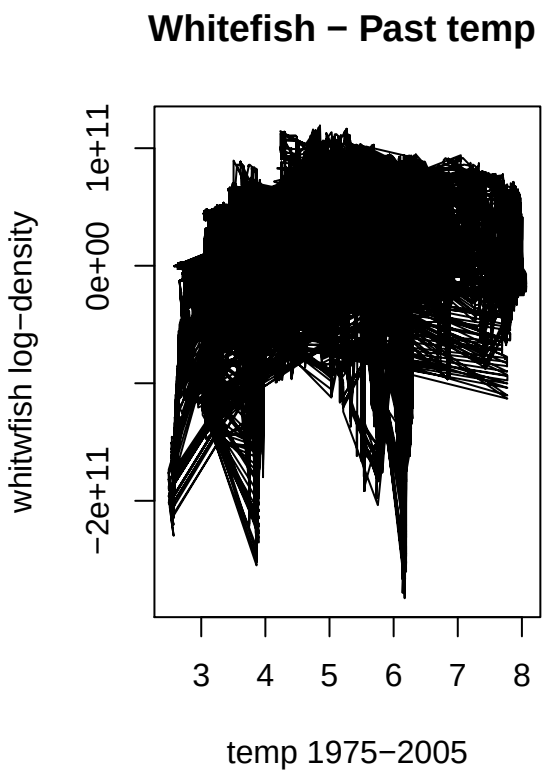
```
p.vq=readRDS("pred_v2.RData")
p.vq.fut=readRDS("pred_v2_fut.RData")

par(mfrow=c(1,2))
# Posterior median of difference of past and future density
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.vq$Ef-p.vq.fut$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Difference in v
# Posterior median of difference of past and future density
```

```
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.vq$Ef/p.vq.fut$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Difference in wh
```



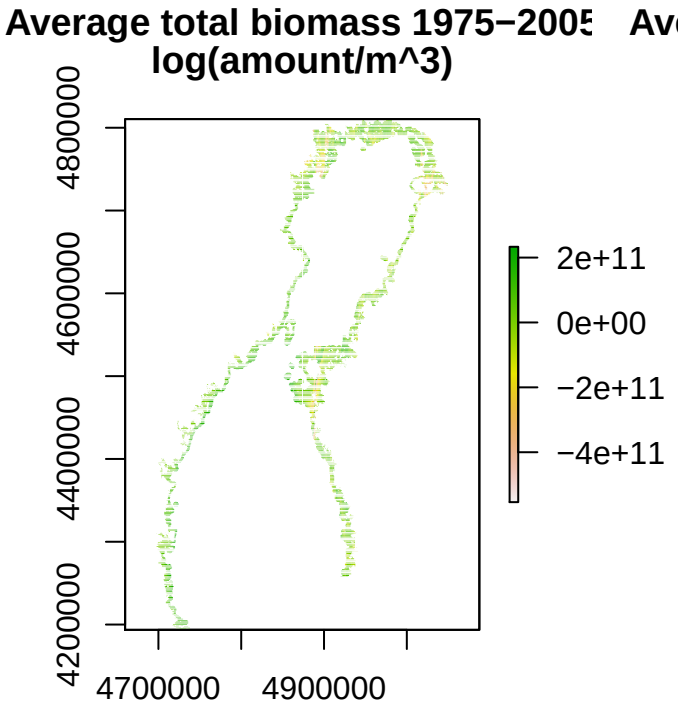
We draw the response curve along salinity, temperature and last ice cover day within the range of the current and future covariate values.

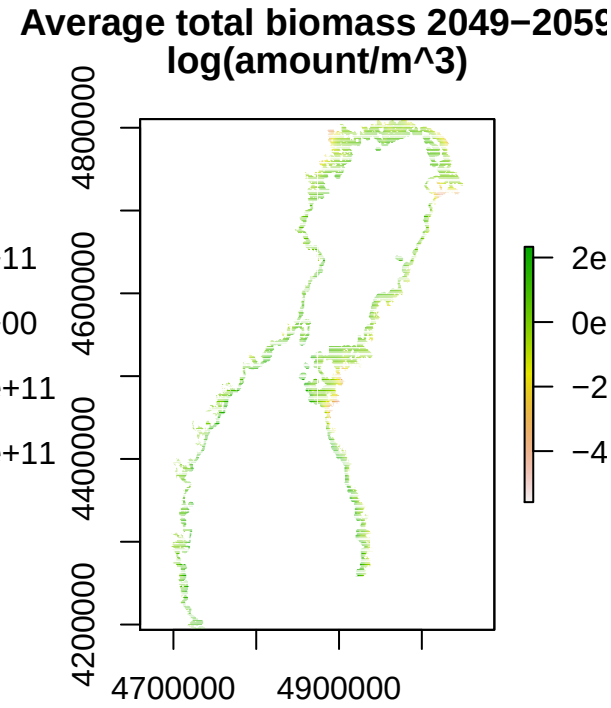
```
### Whitefish
par(mfrow=c(1,2))
plot(whitefish.raster.past$TEMP09M, p.wq$Ef, type="l",xlab="temp 1975-2005", ylab="whitwfish log-density
```



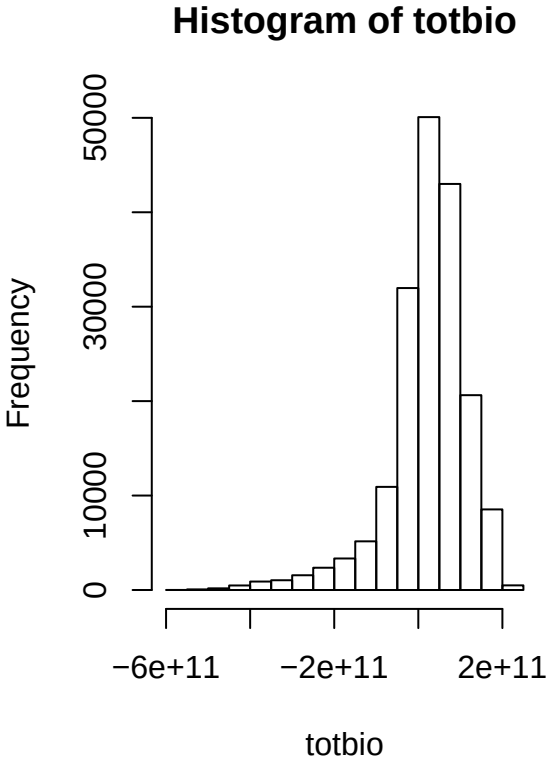
Finally we compute the total biomass in the GoB, for future larvae density of the two species

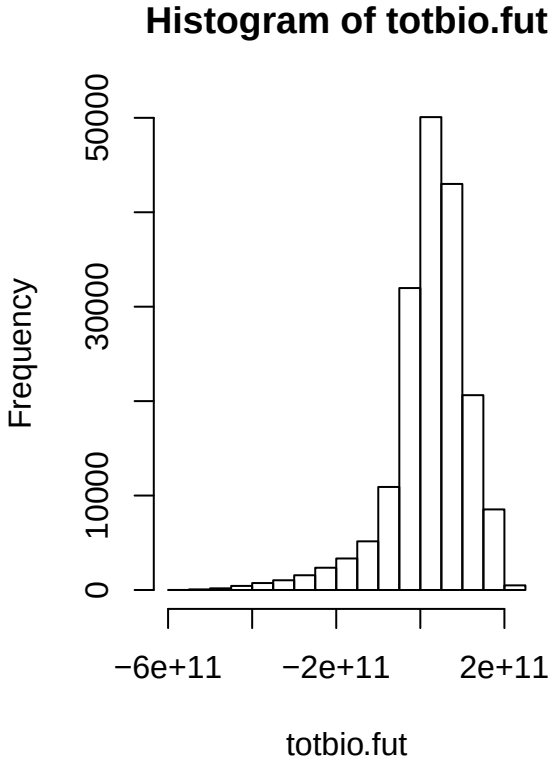
```
par(mfrow=c(1,2))

# Posterior median of tot biomass in the past
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.wq$Ef+p.vq$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average total bi

# Posterior median of tot biomass in the future
z <- rasterize(cbind(whitefish.raster$E_etrs89,whitefish.raster$N_etrs89), r, (p.wq.fut$Ef+p.vq.fut$Ef))
plot(z, xlim=cbind(min(whitefish.raster$E_etrs89),max(whitefish.raster$E_etrs89)),
     ylim=cbind(min(whitefish.raster$N_etrs89),max(whitefish.raster$N_etrs89)), main=c("Average total bi
```






```
totbio=p.wq$Ef+p.vq$Ef
totbio.fut=p.wq.fut$Ef+p.vq.fut$Ef
par(mfrow=c(1,2))
hist(totbio)
hist(totbio.fut)
```





```
# sum(totbio.fut) == sum(totbio.fut) = 3.793988e+15
```

The total biomass is of order 10e+15, while the differences in larvae log-density estimations between the past and the future decreases of maximum 20 units, hence they result not relevant. When computing the total biomass, we see no evidence of changes, due to temperature or salinity modification.

Joint species distribution model

Interactions among species are usually really relevant in estimating species distributions models. We implement a hierarchical joint species distribution model that attributes variation in species occurrence and co-occurrence to the influences of environmental variables and species-to-species associations, as done in r - package Hmsc implementation.

Assuming our response variable Y has a Poisson distribution, with rate parameter (expected value) μ_i, with $i = 1, \dots, n$ observation index, given V_i the offset variable, a simple abboundance HSDM has the following form:

$$\log(\frac{\mu_i}{V_i}) = x_i^T \beta + \phi(s_i)$$

where $\phi(s_i)$ is the random spatial effect.

A JSDM, given the species index $j = 1, 2$, at location s, will be

$$\log(\frac{\mu_{ij}}{V_{ij}}) = x_{ij}^T \beta_j + \phi(s)_{ij}$$

where ϕ_{ij} models variation in species occurrence and co-occurrences. Prior distributions for the latent varaiable's parameters remain the same as in the one-SDM. We assume that the random effect ϕ_{ij} has a GP prior $\phi_i \sim N(0, \Omega_\phi)$ where Ω_ϕ is a species-to-species variance–covariance matrix, following a separable model, that is

$$\Sigma_\phi = R \otimes T$$

where R is a $n \times N$ matrix whose elements account $\rho(s_i, s_j)$ for saptial correlation among sites s_i and s_j, following e.g. a Matern covariance function, whose parameters priors are the same as in SSDMs. While T is a 2×2 matrix that account for correlations amng the two species, for wich we assume the prior

$$T \sim IW(2, diag(0.001, 0.001))$$

The symbol $\otimes$ stands for the Kronecker product, that in our case, gives:

$$(R \otimes T)_{ij} = r_{\lfloor (i-1)/2 \rfloor + 1, \lfloor (j-1)/2 \rfloor + 1} t_{(i-1)\%N+1, (j-1)\%N+1}$$

We now implement the JSDM

```
GP_JSDM = "
  functions{
  matrix Kron.prod(matrix R, matrix T, data int N) {
    matrix[2*N,2*N] kron;
    int m;
    int n;
    int p;
    int q;

    for( i in 1:2*N){
      for(j in 1:2*N) {
        m = ((i-1)/2)+1; //implicit floor
        n = ((j-1)/2)+1;
        p = (i-1)%N+1;
        q = (j-1)%N+1;
        kron[i,j] = R[m,n] * T[p,q];
      }
    }
    return kron;
  }
```

```
}
data {
  int<lower=1> N;
  int<lower=1> Dx;
  matrix[2*N,Dx] x;
  int<lower=1> Ds;
  matrix[2*N,Ds] s;
  int<lower=0> y[2*N];
  vector[2*N] V;
}
transformed data {
  vector[2*N] mu;
  matrix[N, N] Dist_spatial;
  matrix[2*N, 2*N] Sigma_lin;
  real s2_lin;
  matrix[2,2] Sigma_T;

  Sigma_T = diag_matrix( rep_vector(0.001,2));
  s2_lin = 10;
  for (i in 1:2*N)
    mu[i] = 0;
  // off-diagonal elements
  for (i in 1:(2*N-1)) {
    for (j in (i+1):2*N) {
      Sigma_lin[i, j] = s2_lin * dot_product(x[i],x[j]);     // linear covariance function

      // Fill in the other half
      Sigma_lin[j, i] = Sigma_lin[i, j];
    }
  }
  // diagonal elements
  for (k in 1:2*N){
    Sigma_lin[k, k] = s2_lin * dot_product(x[k],x[k]) + 1e-6;   // add also some jitter
  }

  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      Dist_spatial[i, j] = pow(dot_self(s[i] - s[j]),0.5)  ;

      // Fill in the other half
      Dist_spatial[j, i] = Dist_spatial[i, j];
    }
  }
  // diagonal elements
  for (k in 1:N){
    Dist_spatial[k, k] = 0;
  }
}
parameters {
  real<lower=0> l;
  real<lower=0> s2_matern;
  vector[2*N] z;
  matrix[2,2] T;
```

```
  }
  transformed parameters {
    matrix[2*N, 2*N] Sigma_phi;
    matrix[2*N, 2*N] Sigma;
    matrix[2*N, 2*N] L;
    real<lower=0> inv_l;

    inv_l = inv(l);
    Sigma_phi = Kron.prod(s2_matern*exp(-inv_l*Dist_spatial ) , T, N) ; // Exponential
   //   Sigma_phi = Kron.prod(s2_matern*(1 + pow(3,0.5)*inv_l*Dist_spatial).*exp(-pow(3,0.5)*inv_l*Dist_
   Sigma = Sigma_phi + Sigma_lin;
    L = cholesky_decompose(Sigma);
  }
  model {
    vector[2*N] ff;

    // A weakly informative prior for magnitude
    s2_matern ~ student_t(4, 0, 1);

    // A weakly informative prior for l
    l ~ gamma(7, 0.1);

    // A weakly informative prior for T
    T ~ inv_wishart(2, Sigma_T  );

    z ~ normal(0, 1);
    ff = L*z;

    for (n in 1:2*N)
      y[n] ~ poisson(V[n]*exp(ff[n]));

  }
  generated quantities {
    vector[2*N] f;
    // derived quantity (transform)
    f = L*z;
  }"
```

In the previous model we use the choleski decomposition of the covariance matrix: $\Sigma_\phi = LL^T$, to recover the latent variable $f = Lz$, where $z \sim N(0, 1)$. We reimplement the model using the directly linear weight estimations, so that $\mu_f = X\beta$.

```
GP_JSDM1 = "
  functions{
  matrix Kron.prod(matrix R, matrix T, data int N) {
    matrix[2*N,2*N] kron;
    int m;
    int n;
    int p;
    int q;

    for( i in 1:2*N){
      for(j in 1:2*N) {
        m = ((i-1)/2)+1; //implicit floor
```

```
      n = ((j-1)/2)+1;
      p = (i-1)%N+1;
      q = (j-1)%N+1;
      kron[i,j] = R[m,n] * T[p,q];
    }
  }
  return kron;
}

}
data {
  int<lower=1> N;
  int<lower=1> Dx;
  matrix[2*N,Dx] x;
  int<lower=1> Ds;
  matrix[2*N,Ds] s;
  int<lower=0> y[2*N];
  vector[2*N] V;
}
transformed data {
  matrix[N, N] Dist_spatial;
  real s2_lin;
  matrix[2,2] Sigma_T;

  Sigma_T = diag_matrix( rep_vector(0.001,2));
  s2_lin = 10;


  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      Dist_spatial[i, j] = pow(dot_self(s[i] - s[j]),0.5)  ;

      // Fill in the other half
      Dist_spatial[j, i] = Dist_spatial[i, j];
    }
  }
  // diagonal elements
  for (k in 1:N){
    Dist_spatial[k, k] = 0;
  }
}
parameters {
  real<lower=0> l;
  real<lower=0> s2_matern;
  matrix[2,2] T;
  vector[2*N] b;
  vector[2*N] f;

}
transformed parameters {
  matrix[2*N, 2*N] Sigma_phi;
  real<lower=0> inv_l;
  vector[2*N] mu;
```

```
   inv_l = inv(l);
   Sigma_phi = Kron.prod(s2_matern*exp(-inv_l*Dist_spatial ) , T, N) ; // Exponential
  //   Sigma_phi = Kron.prod(s2_matern*(1 + pow(3,0.5)*inv_l*Dist_spatial).*exp(-pow(3,0.5)*inv_l*Dist_
   for(n in 1:2*N) {
     mu[n] =   dot_product(  x[n,], b) ;
     Sigma_phi[n,n] = Sigma_phi[n,n] + 1e-6; //jiitter
   }
 }
 model {

   // A weakly informative prior for magnitude
   s2_matern ~ student_t(4, 0, 1);

   // A weakly informative prior for l
   l ~ gamma(7, 0.1);

   // A weakly informative prior for T
   T ~ inv_wishart(2, Sigma_T  );

   f ~ multi_normal(mu, Sigma_phi);

   for (n in 1:2*N)
     y[n] ~ poisson(V[n]*exp(f[n]));

 }
"
```

Problems with the kronecker product function...

```
y.ven = whitefish.dat$VENSUM[-vol0]

jsdm_data <-  list(Dx = ncol(x),
                   N = nrow(x),
                   Ds = ncol(s),
                   x = rbind(x,x), #same covariates
                   s = rbind(s,s),
                   y = c(y,y.ven),
                   V = c(V,V))
fit.j = stan(model_code =  GP_JSDM, data = jsdm_data, warmup=150, iter = 500, chains = 1,
          #    init=list( list(f=as.vector(rep(0,nrow(x))), l=1, s2_matern=1)) ,
          control = list(adapt_delta = 0.99), pars=c("f", "s2_matern", "l")  )
m = as.matrix(fit.j)


print(fit.j)              # Rhat
summary(fit.j, pars = c("s2_matern", "l", "f[1]"))            # summary
stan_trace(fit.j, pars = c("s2_matern", "l", "f[1]"))         # traceplot
stan_ac(fit.j,inc_warmup = FALSE, lags = 25, pars = c("s2_matern", "l", "f[1]"))   # autocorrelation be
quietgg(stan_hist(fit.j, pars = c("s2_matern", "l", "f[1]")))   # posterior density
# scatter plot of parameters of spatial random effect
stan_scat(fit.j, pars = c("l", "s2_matern"), color = "black", size = 3)
```

Bottom type

Now we consider the bottom coverage type, stored in the categorical variable which were excluded from the model covariates. Each site was classified to 5 BOTTOMCOV and seven BOTTOM types as follows:

```
1 = pehmeä (mud),
2 = hiesu (silt),
3 = hiekka (sand),
4 = hiekka/kivi (Sa/st),
5 = kivi nyrkki (Stones),
6 = kivi lohkare (Rocks),
7 = kallio (Cliff).
```

We will further classify each site with COVERAGE: 0 = clear and 1 = covered with vegetation and use such variables in order to study whether there is difference in the probability of presence of white fish and vendace in clear and vegetated areas. Moreover, we want to compare the differences between different bottom types. Indeed former studies suggests that the survival of white fish larvae is decreased by algal or other bottom vegetation which have been increasing throughout Finnish and Swedish coastal region due to eutrophication.

Let's denote by $\theta_{b,c}$ the fraction of locations where white fish larvae are present out of all locations with bottom type $b \in \{1, \ldots, 7\}$ and coverage $c = \{0, 1\}$ throughout the study region. Let's assume that there is no prior information on $\theta_{b,c}$ so that their prior is uniform between 0 and 1. Let's also assume that the parameters $\theta_{b,c}$ are mutually independent.

First we select the data of interest in order to analyse how the bottom coverage affects white fishes pawns. The variables of interest are

- BOTTOM (bottom class, exclude class 0 from data),
- BOTTOMCOV, a classification of vegetation cover: exclude 0 and after that classify as clear all the sites where BOTTOMCOV < 5 and covered all the sites where BOTTOMCOV $= 5$
- WHIBIN (presence of the white fish)

```
library(readxl)
setwd("/home/piailari/Documents/BayesianDataAnalysis/exercises/week3/exercise2b")
data.full = read_xlsx("bsg653_3.xlsx")
data = data.full[, c( "WHIBIN", "BOTTOMCOV", "BOTTOM")]
data = data[data$BOTTOM!=0,]
# 0 : clear
# 1 : covered
data$BOTTOMCOV = ifelse(data$BOTTOMCOV == 5, 1, 0)

# write.table(data, file="white_fishes_data.txt", row.names=FALSE, col.names=TRUE)
```

We want to model the fraction of locations where white fish larvae are present out of all locations with bottom type $b \in \{1, \ldots, 7\}$ and coverage $c = \{0, 1\}$. The posterior of interest is $\theta_{b,c} \sim Beta(y_{b,c} + 1,\ N_{b,c} - y_{b,c} + 1)$ for all the possible combinations of b and c.

Let's have a look at the data, and store the variables $y_{b,c}$, $N_{b,c}$ in two arrays

```
table(data)
```

```
## , , BOTTOM = 1
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0   3  44
##      1   7  45
##
```

```
## , , BOTTOM = 2
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0   4   8
##      1  19  15
##
## , , BOTTOM = 3
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0  26   6
##      1 109   6
##
## , , BOTTOM = 4
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0   6   8
##      1  16   7
##
## , , BOTTOM = 5
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0   5  11
##      1  12  11
##
## , , BOTTOM = 6
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0  18  25
##      1  43  34
##
## , , BOTTOM = 7
##
##       BOTTOMCOV
## WHIBIN   0   1
##      0   3   2
##      1   6   3
```

```
y = table(data$BOTTOM[data$WHIBIN ==1], data$BOTTOMCOV[data$WHIBIN ==1])
N = table(data$BOTTOM, data$BOTTOMCOV)
```

Now we sample from the posterior distributions of $\theta_{b,c}$ $\forall b, c$:

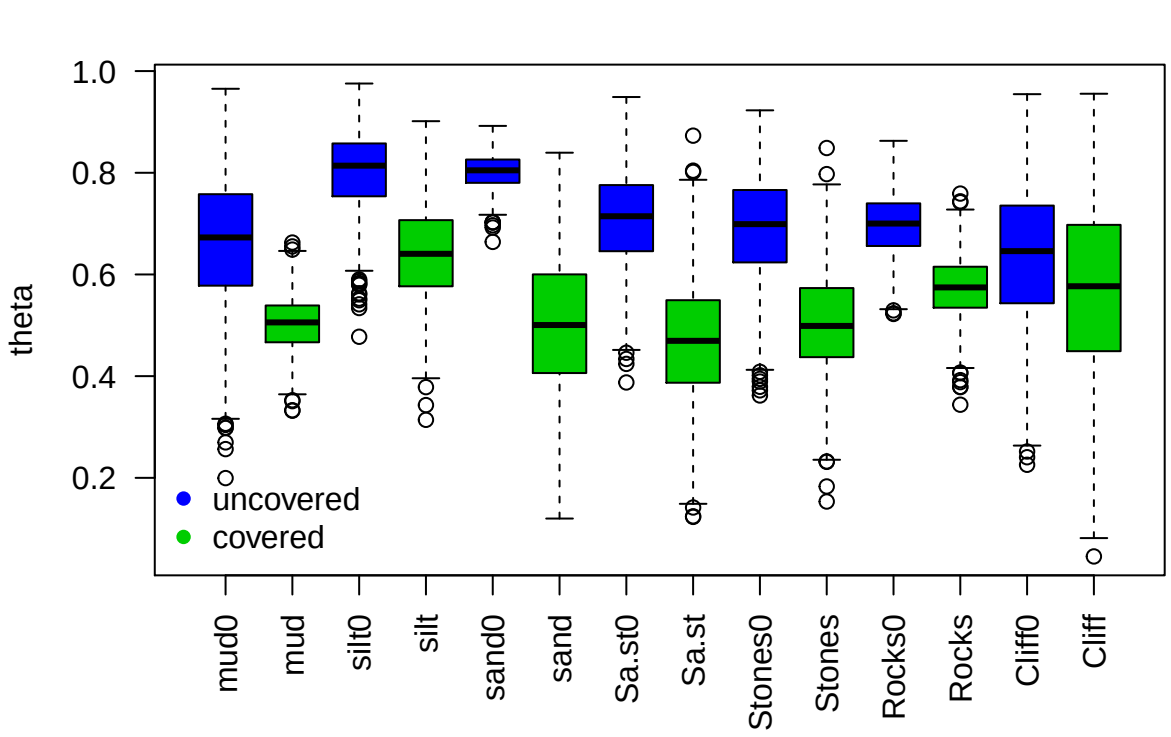
```
set.seed(123)
theta =  array(rep(NA, 7*2*1000), dim=c(7, 2, 1000))
for (i in 1:7) {
   for (j in 1:2) {
      theta[i,j,] = rbeta(1000, y[i,j]+1, N[i,j]-y[i,j]+1)
   }
}
```

Observe the following plot.

The different coastal types are coded as follows:

1 = mud
2 = silt
3 = sand
4 = Sa/st
5 = Stones
6 = Rocks
7 = Cliff

```
theta.df = data.frame(
 "mud0" = theta[1,1,],"mud" = theta[1,2,],
 "silt0" = theta[2,1,],"silt" = theta[2,2,],
 "sand0"= theta[3,1,], "sand"= theta[3,2,],
 "Sa/st0"= theta[4,1,], "Sa/st"= theta[4,2,],
"Stones0"= theta[5,1,],"Stones"= theta[5,2,],
 "Rocks0"= theta[6,1,],"Rocks"= theta[6,2,],
"Cliff0"= theta[7,1,],"Cliff"= theta[7,2,]
)

boxplot(theta.df, col= 4:3 , las =2, ylab= "theta", main ="Posterior of theta.bc")
legend("bottomleft", legend = c("uncovered", "covered"), col=4:3, pch = 16,bty = "n")
```



We now compute the probability that $\theta_{b,1} > \theta_{b,0}$ for all bottom classes $b \in \{1, \ldots, 7\}$.

```
# Probabilities per costal type
pr = c()
for (i in 1:7) {
   pr[i] = sum(theta[i,2,]>theta[i,1,])/length(theta[i,1,])
}
names(pr) = 1:7
```

```
pr
```

```
##     1     2     3     4     5     6     7
## 0.132 0.091 0.007 0.069 0.087 0.066 0.393
```