R script for mapping zero-inflated autocorrelated species abundance data

Olga Lyashevska

10th August 2015

Contents

List of Tables

List of Figures

1 Session info

```
rm(list = ls())
sessionInfo()

## R version 3.2.0 (2015-04-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu precise (12.04.5 LTS)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8    LC_NUMERIC=C
##  [3] LC_TIME=C               LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=C           LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=C              LC_NAME=C
##  [9] LC_ADDRESS=C            LC_TELEPHONE=C
## [11] LC_MEASUREMENT=C        LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
## [6] methods   base
##
## other attached packages:
##  [1] reshape2_1.4.1  coda_0.17-1     mgcv_1.8-4
##  [4] nlme_3.1-120    stargazer_5.2   corrplot_0.73
##  [7] pscl_1.4.9      lattice_0.20-30 MASS_7.3-39
## [10] ggplot2_1.0.1   geoRglm_0.9-8   rgdal_0.9-3
## [13] raster_2.3-40   sp_1.1-0        geoR_1.7-5.1
## [16] gstat_1.0-22    vimcom_1.2-5    setwidth_1.0-3
## [19] colorout_1.1-0  knitr_1.10.5
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.11.6          formatR_1.2
##  [3] highr_0.5            plyr_1.8.2
##  [5] xts_0.9-7            tools_3.2.0
##  [7] digest_0.6.8         evaluate_0.7
##  [9] gtable_0.1.2         Matrix_1.1-5
## [11] proto_0.3-10         stringr_1.0.0
## [13] grid_3.2.0           spacetime_1.1-4
## [15] tcltk_3.2.0          magrittr_1.5
## [17] scales_0.2.4         splancs_2.01-37
## [19] codetools_0.2-10     intervals_0.15.0
## [21] RandomFields_3.0.62 colorspace_1.2-6
## [23] labeling_0.3         stringi_0.4-1
## [25] munsell_0.4.2        FNN_1.1
## [27] zoo_1.7-12
```

```r
set.seed(5021)
```

2 Custom functions

2.1 Function to classify zeros either as a Bernoulli or a Poisson

```r
myfun.sep.large <- function(sam) {
    samB <- sam
    samB$macoma <- ifelse(samB$macoma > 0, 1, 0)
    samP <- sam
    glmB <- glm(formula = macoma ~ mgs + mgs2 + silt + silt2 +
        depth + depth2 + oost + noord, family = "binomial", data = subset(samB,
        select = -c(x, y)))
    glmZ <- zeroinfl(formula = macoma ~ mgs + mgs2 + silt + silt2 +
        depth + depth2 + oost + noord, link = "logit", dist = "poisson",
        data = samP)
    p.glmB <- predict(glmB, type = "response")
    summary(p.glmB)
    lp.glmZ <- predict(glmZ, type = "zero")
    # prob of 1
    p.glmZ <- 1 - lp.glmZ
    summary(p.glmZ)
    mu.glmZ <- predict(glmZ, type = "count")
    p.glmZ.tot <- p.glmZ * exp(-mu.glmZ)
    p.rat <- (1 - p.glmZ)/(1 - p.glmZ + p.glmZ.tot)
    p.test <- runif(length(samP$macoma))
    echte <- p.rat > p.test
    keep <- ifelse(echte == TRUE & sam$macoma == 0, FALSE, TRUE)
    samP0 <- samP[keep, ]
    samB0 <- samB
    samB0$macoma <- ifelse(keep == TRUE, 1, samB$macoma)
    return(list(bin = samB0, pois = samP0))
}
```

```r
myfun.sep.small <- function(sam) {
    samB <- sam
    samB$macoma <- ifelse(samB$macoma > 0, 1, 0)
    samP <- sam
    glmB <- glm(formula = macoma ~ silt + silt2 + depth, family = "binomial",
        data = subset(samB, select = -c(x, y)))
    glmZ <- zeroinfl(formula = macoma ~ silt + silt2 + depth,
        link = "logit", dist = "poisson", data = samP)
    p.glmB <- predict(glmB, type = "response")
    summary(p.glmB)
    lp.glmZ <- predict(glmZ, type = "zero")
    summary(lp.glmZ)
    p.glmZ <- 1 - lp.glmZ
    summary(p.glmZ)
    mu.glmZ <- predict(glmZ, type = "count")
    p.glmZ.tot <- p.glmZ * exp(-mu.glmZ)
    p.rat <- (1 - p.glmZ)/(1 - p.glmZ + p.glmZ.tot)
    p.test <- runif(length(samP$macoma))
    echte <- p.rat > p.test
    keep <- ifelse(echte == TRUE & sam$macoma == 0, FALSE, TRUE)
    samP0 <- samP[keep, ]
    samB0 <- samB
    samB0$macoma <- ifelse(keep == TRUE, 1, samB$macoma)
    return(list(bin = samB0, pois = samP0))
}
```

2.2 Functions to back-transform predictions and simulations

```r
# back transforming predictions when nsim=0, method of
# Christensen
# https://github.com/cran/geoRglm/blob/master/R/binom.pred.R
# https://github.com/cran/geoRglm/blob/master/R/extensions.R
# for binomial
transf.predB <- function(var1.pred, var1.var) {
    plogis(var1.pred) + 0.5 * (exp(var1.pred) * (-expm1(var1.pred))/(1 +
        exp(var1.pred))^3) * var1.var
}
transf.varB <- function(var1.pred, var1.var) {
    (exp(var1.pred)/(1 + exp(var1.pred))^2)^2 * var1.var + (1/2) *
        (exp(var1.pred) * (-expm1(var1.pred))/(1 + exp(var1.pred))^3)^2 *
        var1.var^2
}
# for poisson
transf.predP <- function(var1.pred, var1.var) {
    exp(var1.pred + 0.5 * var1.var)
}
transf.varP <- function(var1.pred, var1.var) {
    (exp(2 * var1.pred + var1.var)) * expm1(var1.var)
}
# back-transforming simulations when nsim=1 or when using
# output from glsm.mcmc for binomial
antilogit <- function(x) {
    exp(x)/(1 + exp(x))
}
# for poisson use `exp'
```

2.3 Function to perform scaling of variables

```r
myscale <- function(x) {
    (x - mean(x))/sd(x)
}
```

2.4 Function to generate counts from prevalence and intensity

```
myfun.count <- function(pred.pi, pred.mu) {
    pred.pi <- data.frame(pred.pi)
    pred.mu <- data.frame(pred.mu)
    datb <- numeric()
    for (i in 1:ncol(pred.pi)) {
        # generate random samples from bin dist
        b <- rbinom(n = nrow(pred.pi), size = 1, p = pred.pi[,
            i])
        datb <- cbind(datb, b)
    }
    datp <- numeric()
    for (j in 1:ncol(pred.mu)) {
        # generate random samples from pois dist
        d <- rpois(n = nrow(pred.mu), lambda = pred.mu[, j])
        datp <- cbind(datp, d)
    }
    res <- ifelse(datb == 0, 0, datp)
    res <- data.frame(res)
    names(res) <- c(paste(rep("X", ncol(pred.pi)), c(1:ncol(pred.pi)),
        sep = ""))
    res <- res[complete.cases(res), ]
}
```

2.5 Function to sample data

```
myfun.sample <- function(x, ...) {
    if (all(is.na(x))) {
        return(NA)
    }
    return(sample(x[!is.na(x)], ...))
}
```

3 Data import and screening

```
## ----dat.sep
dat <- read.csv("input/dat.csv", header = T)
# remove 4 values in macoma >100
dat <- subset(dat, macoma < 100)
dat$mgs2 <- dat$mgs^2
dat$silt2 <- dat$silt^2
dat$depth2 <- dat$depth^2
dat <- subset(dat, select = -grid)
dat <- subset(dat, select = -X)
meancov <- apply(dat, 2, mean)
sdcov <- apply(dat, 2, sd)
stargazer(subset(dat, select = -c(x, y)), summary = TRUE, rownames = FALSE,
    title = "Original data, 4026 observations", nobs = FALSE)
```

Table 1: Original data, 4026 observations

Statistic	Mean	St. Dev.	Min	Max
macoma	1.408	4.898	0	84
mgs	151.770	42.182	20.400	369.900
silt	13.711	14.702	0.000	78.800
depth	−44.275	45.314	−199	96
oost	1,730.550	386.552	1,155.229	2,603.047
noord	5,917.922	171.126	5,454.327	6,170.868
mgs2	24,813.130	12,505.370	416.160	136,826.000
silt2	404.072	805.920	0.000	6,209.440
depth2	4,013.121	5,315.251	0	39,601

```r
dat <- cbind(dat[, c("macoma", "x", "y")], apply(dat[, c("mgs",
    "mgs2", "silt", "silt2", "depth", "depth2", "oost", "noord")],
    2, myscale))
stargazer(subset(dat, select = -c(x, y)), summary = TRUE, rownames = FALSE,
    title = "Data after scaling, 4026 observations", nobs = FALSE)
```

Table 2: Data after scaling, 4026 observations

Statistic	Mean	St. Dev.	Min	Max
macoma	1.408	4.898	0	84
mgs	0.000	1.000	−3.114	5.171
mgs2	−0.000	1.000	−1.951	8.957
silt	−0.000	1.000	−0.933	4.427
silt2	−0.000	1.000	−0.501	7.203
depth	−0.000	1.000	−3.415	3.096
depth2	−0.000	1.000	−0.755	6.695
oost	−0.000	1.000	−1.488	2.257
noord	0.000	1.000	−2.709	1.478

```r
# datsc<-dat write.csv(datsc, 'input/datsc.csv')
```

```r
df <- read.csv("./input/dat.csv")
df <- subset(df, macoma < 100)
print(ggplot(df, aes(macoma)) + geom_histogram(binwidth = 2,
    position = "dodge") + scale_x_continuous(name = "Species abundance") +
    scale_y_continuous(name = "Counts"))
# to find skewness library(agricolae) skewness(df$macoma)
```

```r
corrplot(cor(dat), method = "ellipse", type = "lower")
```

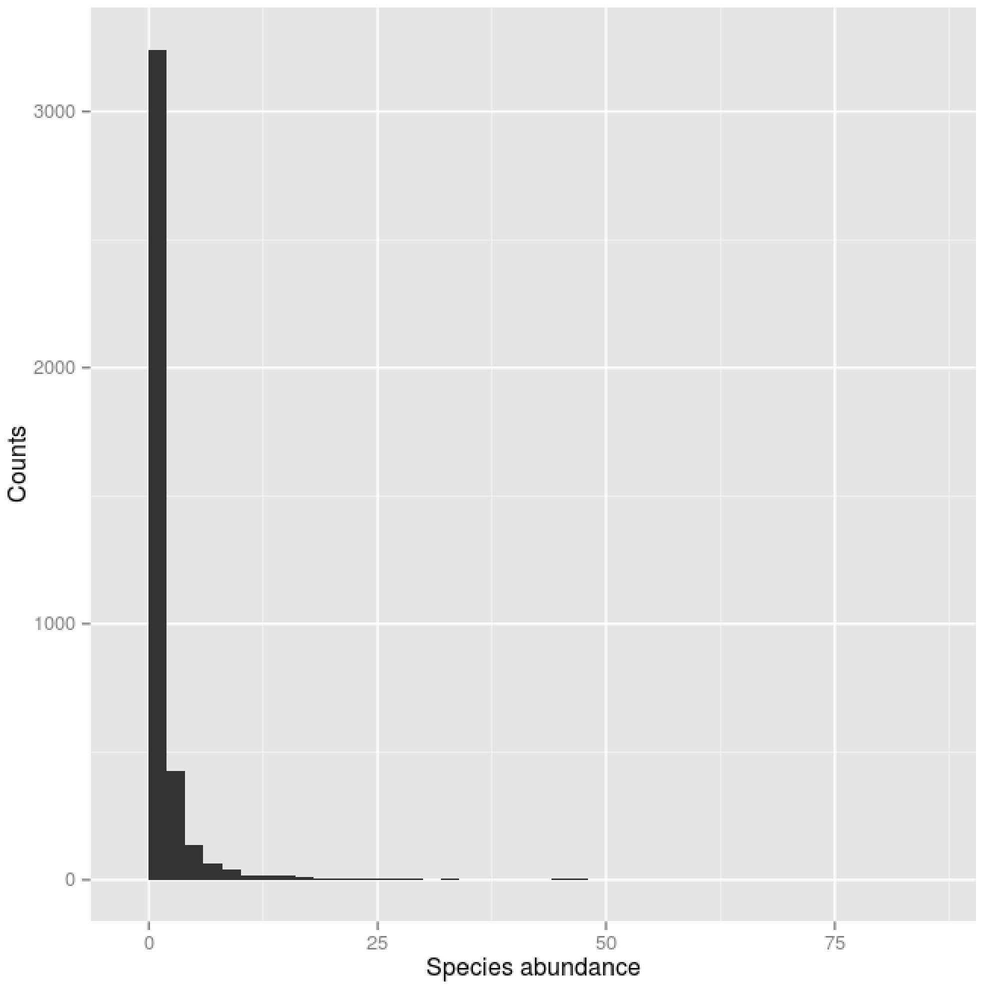


Figure 1: Histogram of counts of Macoma balthica.

```
intertidal.shp<-readOGR("./input/intertidal", "intertidal")

## OGR data source with driver: ESRI Shapefile
## Source: "./input/intertidal", layer: "intertidal"
## with 141 features
## It has 3 fields

RD<-CRS("+proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889
+k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bessel
+towgs84=565.2369,50.0087,465.658,-0.406857330322398,
0.350732676542563,-1.8703473836068,4.0812 +units=m +no_defs")
intertidal.shp<-spTransform(intertidal.shp, RD)
gg.intertidal<-ggplot(intertidal.shp) +
geom_polygon(aes(x = long/1000, y = lat/1000, group=group), alpha=0.3)+
scale_y_continuous(name = "Northing (km)") +
scale_x_continuous(name = "Easting (km)") +
coord_equal()
b<-c(-Inf,1,2, Inf)
l<-c("[0-1)","[1-2)", "[2-84)")
colo<-c( "#FFFF00", "#7FC400", "#008B00")
#breaks for data
```

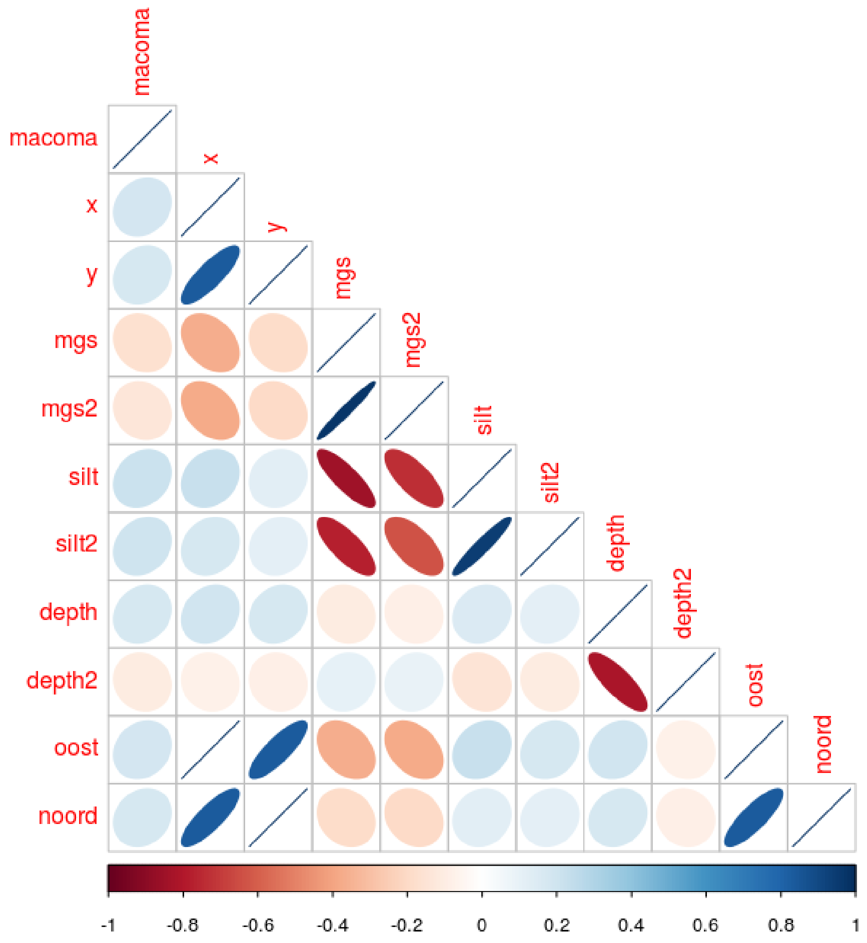


Figure 2: A graphical display of the correlation matrix of environmental variables considered for inclusion in model

```
df$group<-cut(df$macoma,breaks=b, labels=l)
print(gg.intertidal+
      geom_point(data = df, aes( x = x* 1.0e-3, y = y* 1.0e-3, colour=group),size=0.5)+
      scale_colour_manual(name = "", values = colo)+
      theme(legend.position="bottom")+
      guides(colour = guide_legend(override.aes = list(size=3))))
```

4 Model selection and zero classification

4.1 Bernoulli

```
# binomial model
datB <- dat
datB$macoma <- ifelse(datB$macoma > 0, 1, 0)
glmB <- glm(formula = macoma ~ mgs + mgs2 + silt + silt2 + depth +
    depth2 + oost + noord, family = binomial, data = datB)
dropterm(step(glmB), test = "Chisq")

## Start:  AIC=4533.41
## macoma ~ mgs + mgs2 + silt + silt2 + depth + depth2 + oost +
```

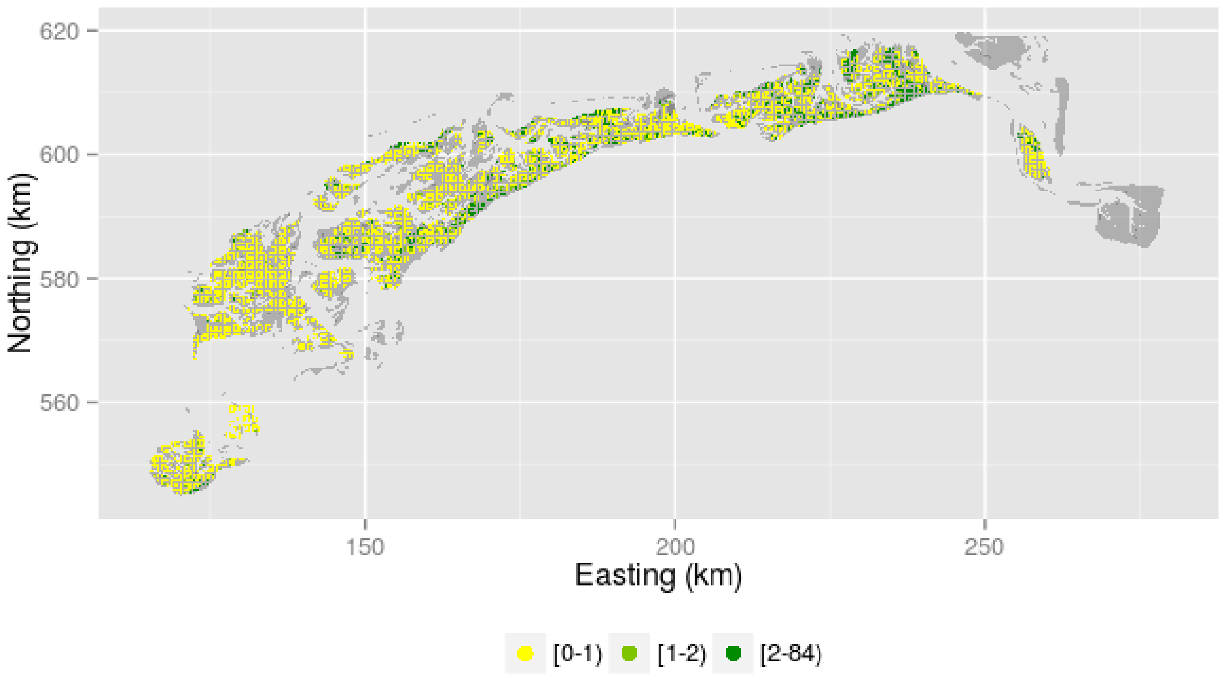


Figure 3: Empirical species abundance map of Macoma balthica. At many locations (yellow dots) the counts equal zero, thus assuming Gaussian distribution is inappropriate.

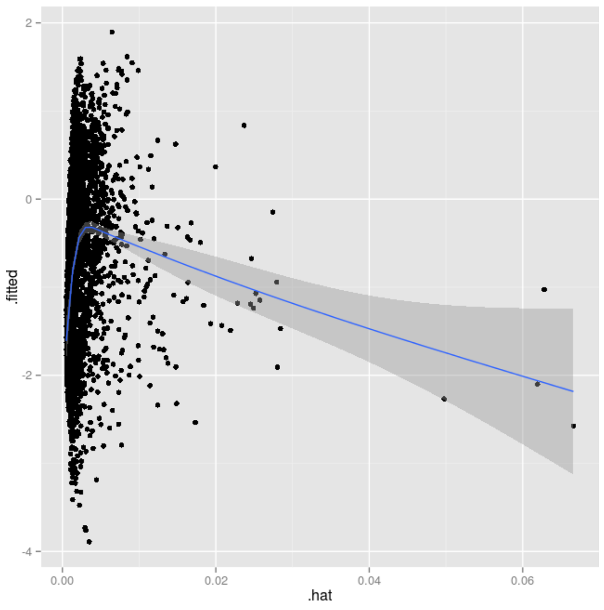
```
##     noord
##
##          Df Deviance    AIC
## - mgs     1   4515.4 4531.4
## - mgs2    1   4515.6 4531.6
## - depth2  1   4516.3 4532.3
## - oost    1   4517.3 4533.3
## <none>        4515.4 4533.4
## - noord   1   4533.7 4549.7
## - silt2   1   4534.0 4550.0
## - silt    1   4555.3 4571.3
## - depth   1   4611.1 4627.1
##
## Step:  AIC=4531.42
## macoma ~ mgs2 + silt + silt2 + depth + depth2 + oost + noord
##
##          Df Deviance    AIC
## - depth2  1   4516.3 4530.3
## - oost    1   4517.4 4531.4
## <none>        4515.4 4531.4
## - mgs2    1   4518.2 4532.2
## - noord   1   4534.3 4548.3
## - silt2   1   4545.7 4559.7
## - silt    1   4555.6 4569.6
## - depth   1   4611.1 4625.1
##
## Step:  AIC=4530.32
## macoma ~ mgs2 + silt + silt2 + depth + oost + noord
```

```
##
##          Df Deviance    AIC
## <none>       4516.3 4530.3
## - oost   1   4518.6 4530.6
## - mgs2   1   4518.9 4530.9
## - noord  1   4534.8 4546.8
## - silt2  1   4546.4 4558.4
## - silt   1   4556.3 4568.3
## - depth  1   4731.1 4743.1
## Single term deletions
##
## Model:
## macoma ~ mgs2 + silt + silt2 + depth + oost + noord
##         Df Deviance    AIC     LRT   Pr(Chi)
## <none>      4516.3 4530.3
## mgs2    1   4518.9 4530.9   2.554    0.1100
## silt    1   4556.3 4568.3  39.990 2.552e-10 ***
## silt2   1   4546.4 4558.4  30.077 4.152e-08 ***
## depth   1   4731.1 4743.1 214.804 < 2.2e-16 ***
## oost    1   4518.6 4530.6   2.243    0.1342
## noord   1   4534.8 4546.8  18.502 1.698e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

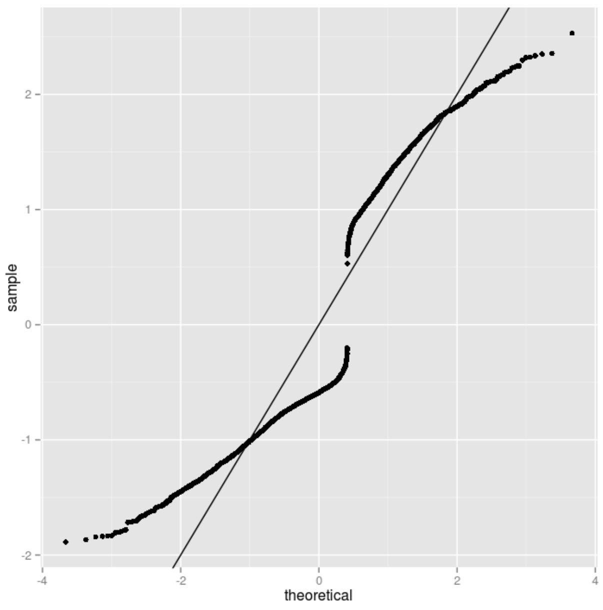
```
ggplot(glmB, aes(.hat, .fitted)) + geom_point() + geom_smooth(se = TRUE)
qplot(sample = .stdresid, data = glmB, stat = "qq") + geom_abline()
qplot(.hat, .stdresid, data = glmB, size = .cooksd) + geom_smooth(se = FALSE,
    size = 0.5)
ggplot(glmB, aes(x = .resid)) + geom_histogram()
```

4.2 Poisson

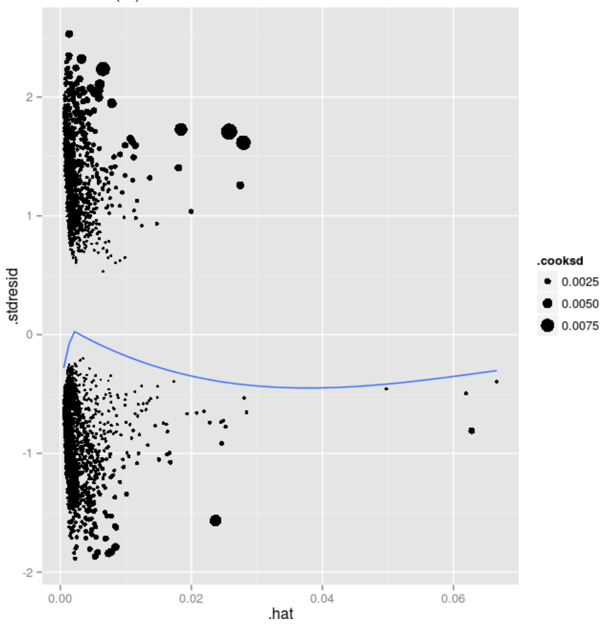
```
datP <- dat
glmP <- glm(formula = macoma ~ mgs + mgs2 + silt + silt2 + depth +
    depth2 + oost + noord, family = poisson(link = log), data = datP)
dropterm(step(glmP), test = "Chisq")

## Start:  AIC=18543.8
## macoma ~ mgs + mgs2 + silt + silt2 + depth + depth2 + oost +
##     noord
##
##           Df Deviance   AIC
## - mgs2    1    14766 18542
## - depth2  1    14767 18543
## - oost    1    14767 18543
## <none>         14766 18544
## - mgs     1    14776 18552
## - silt2   1    14805 18581
```



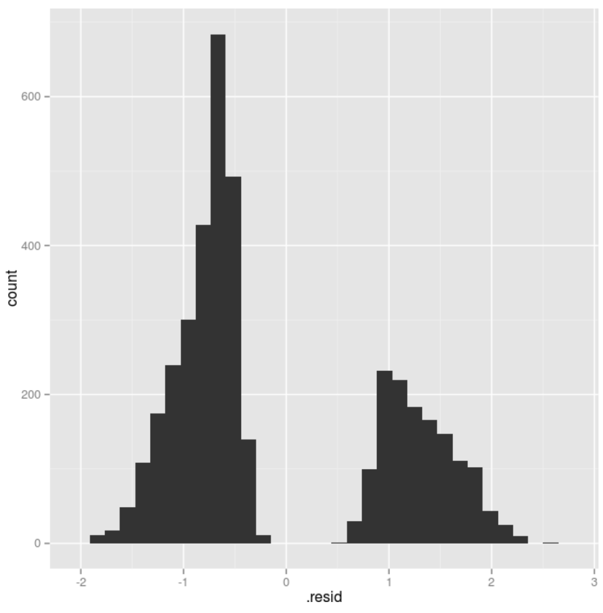
(a) Predicted versus fitted values



(b) QQ plot



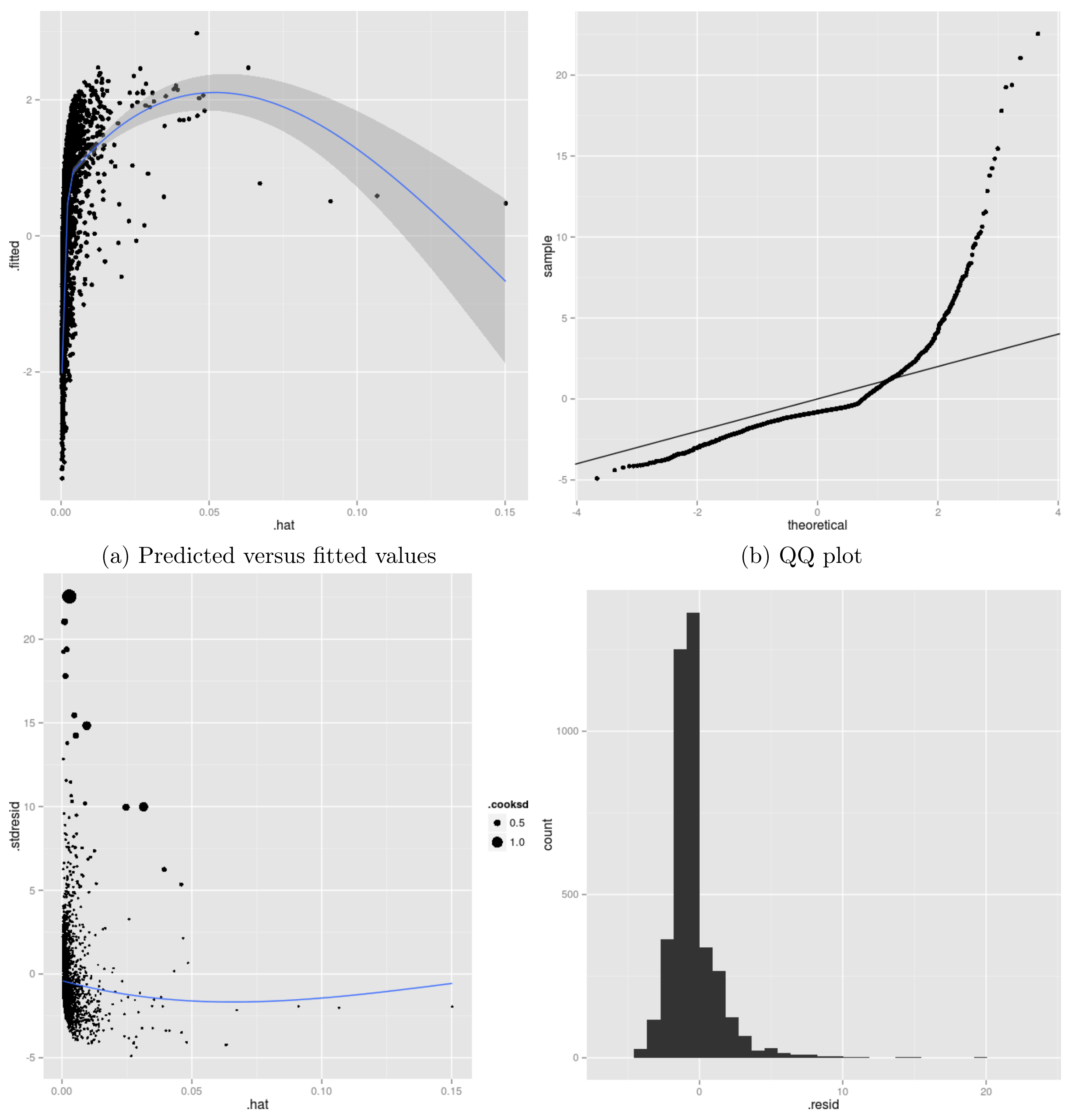
(c) Predicted vs standardized residuals



(d) Histogram of residuals

Figure 4: Regression diagnostics plots, Bernoulli model

```
## - noord   1    15191 18968
## - silt    1    15307 19083
## - depth   1    15385 19161
##
## Step:  AIC=18541.82
## macoma ~ mgs + silt + silt2 + depth + depth2 + oost + noord
##
##          Df Deviance   AIC
## - depth2  1    14767 18541
## - oost    1    14767 18541
## <none>          14766 18542
## - silt2   1    14836 18610
## - mgs     1    14904 18678
## - noord   1    15209 18983
## - silt    1    15308 19082
## - depth   1    15385 19159
##
## Step:  AIC=18540.83
## macoma ~ mgs + silt + silt2 + depth + oost + noord
##
##         Df Deviance   AIC
## - oost   1    14768 18540
## <none>         14767 18541
## - silt2  1    14836 18608
## - mgs    1    14905 18677
## - noord  1    15210 18982
## - silt   1    15312 19084
## - depth  1    15781 19553
##
## Step:  AIC=18540.13
## macoma ~ mgs + silt + silt2 + depth + noord
##
##         Df Deviance   AIC
## <none>         14768 18540
## - silt2  1    14841 18611
## - mgs    1    14924 18695
## - silt   1    15316 19086
## - depth  1    15808 19578
## - noord  1    16505 20275
## Single term deletions
##
## Model:
## macoma ~ mgs + silt + silt2 + depth + noord
##        Df Deviance   AIC     LRT   Pr(Chi)
## <none>       14768 18540
## mgs     1    14924 18695  156.43 < 2.2e-16 ***
## silt    1    15316 19086  547.98 < 2.2e-16 ***
## silt2   1    14841 18611   73.26 < 2.2e-16 ***
## depth   1    15808 19578 1039.72 < 2.2e-16 ***
## noord   1    16505 20275 1736.74 < 2.2e-16 ***
## ---
```



(a) Predicted versus fitted values

(b) QQ plot

(c) Predicted vs standardized residuals

(d) Histogram of residuals

Figure 5: Regression diagnostics plots, Poisson model

```
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ggplot(glmP, aes(.hat, .fitted)) + geom_point() + geom_smooth(se = TRUE)
qplot(sample = .stdresid, data = glmP, stat = "qq") + geom_abline()
qplot(.hat, .stdresid, data = glmP, size = .cooksd) + geom_smooth(se = FALSE,
    size = 0.5)
ggplot(glmP, aes(x = .resid)) + geom_histogram()
```

```r
stargazer(glmB, glmP, no.space=TRUE, single.row=TRUE,
          title='Generalised linear models for Bernoulli and Poisson datasets',
          object.names=FALSE, model.numbers = FALSE)
```

Table 3: Generalised linear models for Bernoulli and Poisson datasets

	Dependent variable:	
	macoma	
	logistic	*Poisson*
mgs	0.016 (0.375)	0.354*** (0.111)
mgs2	−0.123 (0.311)	0.013 (0.091)
silt	0.846*** (0.135)	1.023*** (0.044)
silt2	−0.612*** (0.141)	−0.250*** (0.040)
depth	0.606*** (0.063)	0.475*** (0.017)
depth2	0.068 (0.071)	0.026 (0.026)
oost	0.105 (0.076)	0.034 (0.031)
noord	0.318*** (0.076)	0.757*** (0.041)
Constant	−0.784*** (0.038)	−0.280*** (0.022)
Observations	4,026	4,026
Log Likelihood	−2,257.707	−9,262.898
Akaike Inf. Crit.	4,533.415	18,543.800
Note:		*p<0.1; **p<0.05; ***p<0.01

4.3 Zero classification

```r
trendP <- glmP$formula
trendB <- glmB$formula
dat.sep <- myfun.sep.large(dat)
table(dat.sep$bin$macoma == 0)

##
## FALSE  TRUE
##  1633  2393

table(dat.sep$pois$macoma == 0)

##
## FALSE  TRUE
##  1370   263
```

5 Estimation of variogram parameters

5.1 Method of moments

5.1.1 Bernoulli

```
bin <- dat.sep$bin
bin.glm <- glm(formula = trendB, family = "binomial", data = bin)
bin$residB <- resid(bin.glm, type = "response")
bin.glm.beta <- as.numeric(bin.glm$coefficients)
coordinates(bin) <- c("x", "y")
samplevarB <- variogram(residB ~ 1, data = bin, cutoff = 3000)
modelvarB <- vgm(psill = 0.03, model = "Sph", range = 2500, nugget = 0.2)
modelvarB <- fit.variogram(samplevarB, model = modelvarB)
linevarB <- variogramLine(modelvarB, maxdist = max(samplevarB$dist))
```

5.1.2 Poisson

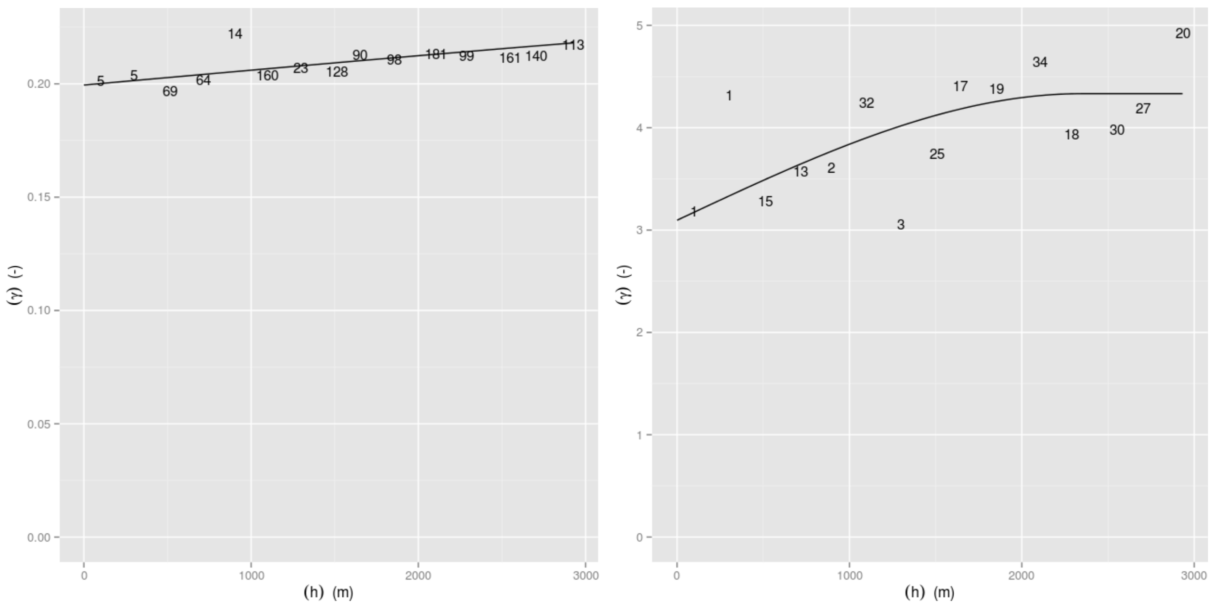
```
pois <- dat.sep$pois
pois.glm <- glm(formula = trendP, family = "poisson", data = pois)
pois$residP <- resid(pois.glm, type = "deviance")
pois.glm.beta <- as.numeric(pois.glm$coefficients)
coordinates(pois) <- c("x", "y")
samplevarP <- variogram(residP ~ 1, data = pois, cutoff = 3000)
modelvarP <- vgm(psill = 2, model = "Sph", range = 2500, nugget = 3)
modelvarP <- fit.variogram(samplevarP, model = modelvarP)
linevarP <- variogramLine(modelvarP, maxdist = max(samplevarP$dist))
```

```
ggplot(data = as.data.frame(samplevarB)) + geom_line(data = linevarB,
    aes(x = dist, y = gamma)) + geom_text(mapping = aes(x = dist,
    y = gamma, label = round(np/100)), size = 4) + scale_x_continuous(name = expression((h) *
    "  (m)")) + scale_y_continuous(name = expression((gamma) *
    "  (-)"), limits = c(0, max(samplevarB$gamma))) + theme(legend.position = "none")
ggplot(data = as.data.frame(samplevarP)) + geom_line(data = linevarP,
    aes(x = dist, y = gamma)) + geom_text(mapping = aes(x = dist,
    y = gamma, label = round(np/100)), size = 4) + scale_x_continuous(name = expression((h) *
    "  (m)")) + scale_y_continuous(name = expression((gamma) *
    "  (-)"), limits = c(0, max(samplevarP$gamma))) + theme(legend.position = "none")
```

5.2 Restricted maximum likelihood

5.2.1 Bernoulli

```
bin.geo <- as.geodata(obj = as.data.frame(bin), header = TRUE,
    coords.col = c("x", "y"), data.col = "residB", data.names = NULL,
    covar.col = c("mgs", "mgs2", "silt", "silt2", "depth", "depth2",
        "oost", "noord"))
bin.reml <- likfit(geodata = bin.geo, trend = "cte", cov.model = "spherical",
    ini.cov.pars = c(modelvarB[2, 2], modelvarB[2, 3]), nugget = modelvarB[1,
        2], lik.method = "REML")
saveRDS(bin.reml, "output/bin.reml.rds")
```



(a) Bernoulli model (b) Poisson model

Figure 6: Empirical and fitted theoretical variograms, semivariances are given by the number of point pairs multiplied by 0.01

```
bin.reml <- readRDS("output/bin.reml.rds")
bin.reml

## likfit: estimated model parameters:
##         beta       tausq     sigmasq         phi
## "   0.0049" "   0.1914" "   0.0242" "3405.0120"
## Practical Range with cor=0.05 for asymptotic range: 3405.012
##
## likfit: maximised log-likelihood = -2562
```

5.2.2 Poisson

```
pois.geo <- as.geodata(obj = as.data.frame(pois), header = TRUE,
    coords.col = c("x", "y"), data.col = "residP", data.names = NULL,
    covar.col = c("mgs", "mgs2", "silt", "silt2", "depth", "depth2",
        "oost", "noord"))
pois.reml <- likfit(geodata = pois.geo, trend = "cte", cov.model = "spherical",
    ini.cov.pars = c(modelvarP[2, 2], modelvarP[2, 3]), nugget = modelvarP[1,
        2], lik.method = "REML")
saveRDS(pois.reml, "output/pois.reml.rds")
```

```
pois.reml <- readRDS("output/pois.reml.rds")
pois.reml

## likfit: estimated model parameters:
##         beta       tausq     sigmasq         phi
## "  -0.3862" "   2.8593" "   1.3472" "2346.1920"
## Practical Range with cor=0.05 for asymptotic range: 2346.192
##
## likfit: maximised log-likelihood = -3428
```

6 Markov chain Monte Carlo

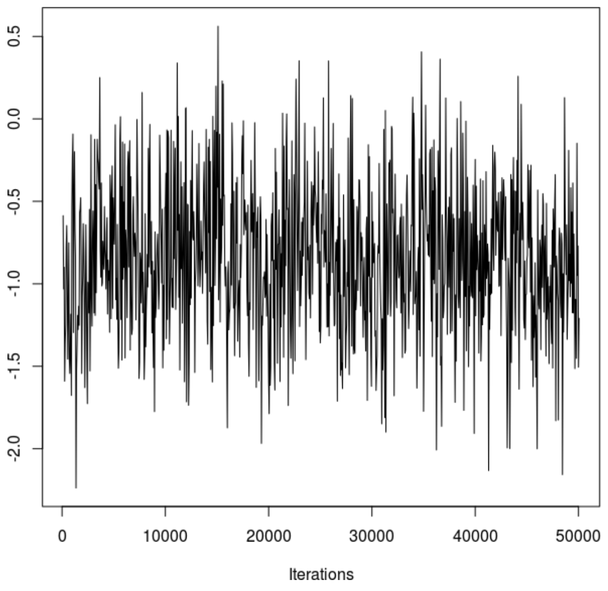
6.1 Bernoulli

```
bin.mcmc.geo<-as.geodata(
    obj = as.data.frame(bin),
    header = TRUE,
    coords.col = c("x","y"),
    data.col = "macoma",
    data.names = NULL,
    covar.col = c("mgs","mgs2", "silt","silt2","depth","depth2", "oost", "noord")
)
mcmcSetB <- mcmc.control(S.scale = 0.15, thin = 50, burn.in = 100)
glgmB <- list(
    family='binomial',
    trend=trend.spatial(trend=trendB, geodata=as.data.frame(bin)),
    cov.model='spherical',
    cov.pars=c(bin.reml$sigmasq, bin.reml$phi),
    nugget =bin.reml$tausq,
    beta=bin.glm.beta)
bin.simFtilde <- glsm.mcmc(
    geodata = bin.mcmc.geo,
    model = glgmB,
    mcmc.input = mcmcSetB,
    messages=TRUE)
saveRDS(bin.simFtilde,"output/bin.simFtilde.rds")
```

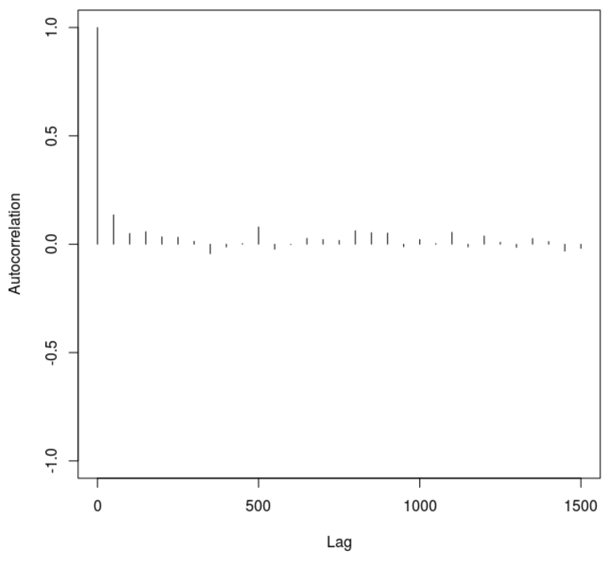
```
mcmcSetB <- mcmc.control(S.scale = 0.15, thin = 50, burn.in = 100)
bin.simFtilde <- readRDS("output/bin.simFtilde.rds")
bin.chainConv1 <- create.mcmc.coda(x = bin.simFtilde$simulations[round(runif(1,
    min = 1, max = nrow(bin.simFtilde$simulations)), 0), ], mcmc.input = mcmcSetB)
traceplot(bin.chainConv1)
autocorr.plot(bin.chainConv1, auto.layout = FALSE)
densplot(bin.chainConv1)
geweke.plot(bin.chainConv1, auto.layout = FALSE)
```

6.2 Poisson

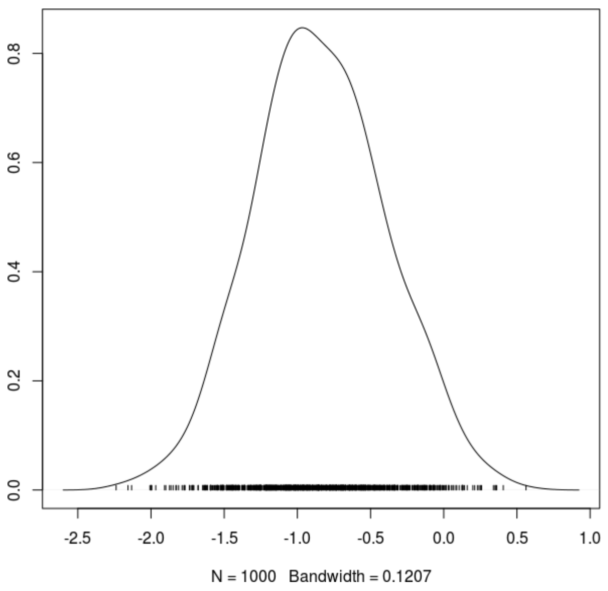
```
pois.mcmc.geo<-as.geodata(
    obj = as.data.frame(pois),
    header = TRUE,
    coords.col = c("x","y"),
    data.col = "macoma",
```



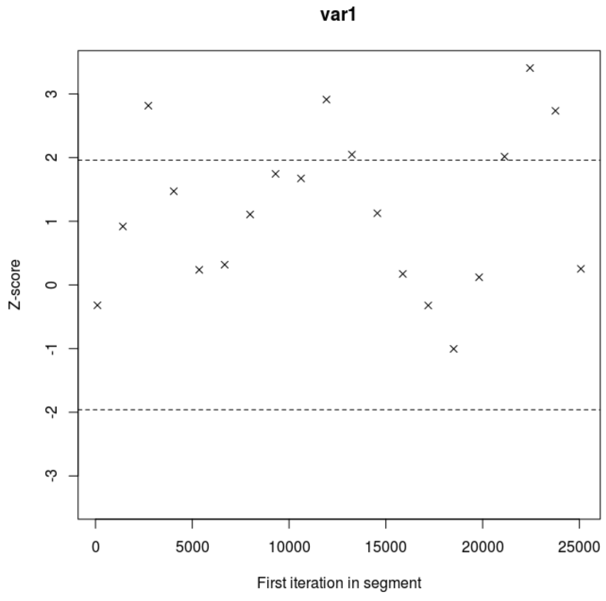
(a) Trace plot of iterations versus sampled values after the burn-in period



(b) The autocorrelation function in each chain



(c) The density plot



(d) Geweke-Brooks plot

Figure 7: MCMC convergence diagnostics, Bernoulli model.

```r
    data.names = NULL,
    covar.col = c("mgs","mgs2", "silt","silt2","depth","depth2", "oost", "noord")
)
mcmcSetP <- mcmc.control(S.scale = 0.1, thin = 50, burn.in = 100)
glgmP <- list(
    family='poisson',
    trend =trend.spatial(trendP, as.data.frame(pois)),
    cov.model='spherical',
    cov.pars=c(pois.reml$sigmasq, pois.reml$phi),
    nugget =pois.reml$tausq,
    beta=pois.glm.beta)
pois.simFtilde <- glsm.mcmc(
    geodata = pois.mcmc.geo,
    model = glgmP,
    mcmc.input = mcmcSetP,
    messages=TRUE)
saveRDS(pois.simFtilde,"output/pois.simFtilde.rds")
```

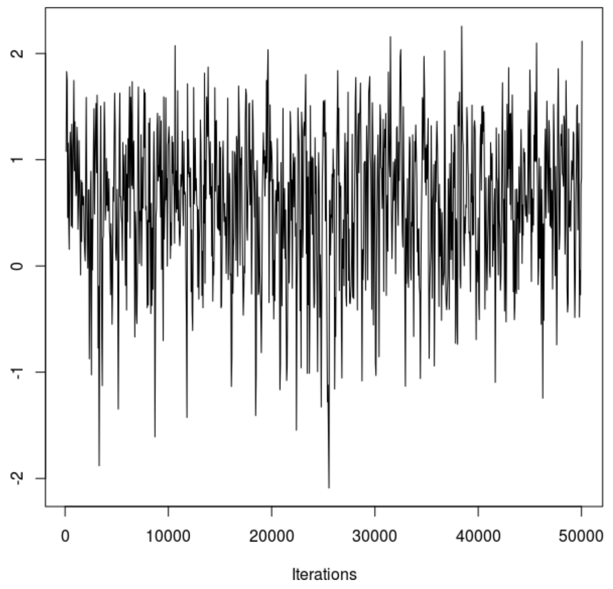
```r
mcmcSetP <- mcmc.control(S.scale = 0.1, thin = 50, burn.in = 100)
pois.simFtilde <- readRDS("output/pois.simFtilde.rds")
pois.chainConv1 <- create.mcmc.coda(x = pois.simFtilde$simulations[round(runif(1,
    min = 1, max = nrow(pois.simFtilde$simulations)), 0), ],
    mcmc.input = mcmcSetP)   #
traceplot(pois.chainConv1)
autocorr.plot(pois.chainConv1, auto.layout = FALSE)
densplot(pois.chainConv1)
geweke.plot(pois.chainConv1, auto.layout = FALSE)
```

7 Markov chain maximum likelihood

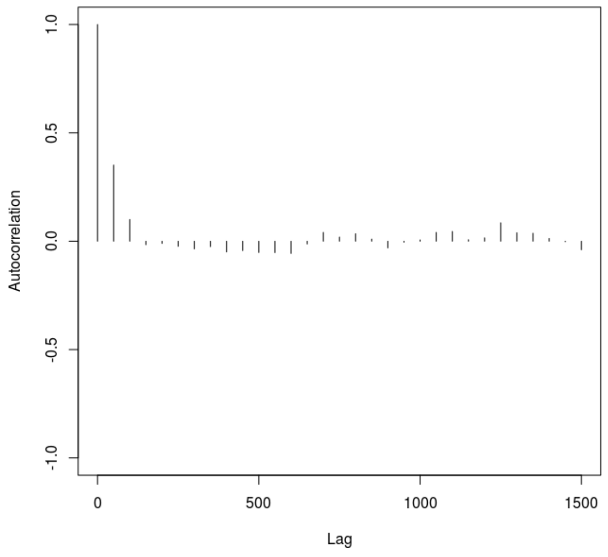
7.1 Bernoulli

```r
bin.mcmlPrep1 <- prepare.likfit.glsm(bin.simFtilde)
bin.mcml1 <- likfit.glsm(mcmc.obj = bin.mcmlPrep1, cov.model = "spherical",
    ini.phi = bin.reml$phi, nugget.rel = (bin.reml$tausq/bin.reml$sigmasq),
    fix.nugget.rel = FALSE, messages = FALSE)
saveRDS(bin.mcml1, "output/bin.mcml1.rds")
bin.simF <- glsm.mcmc(geodata = bin.mcmc.geo, units.m = "default",
    model = bin.mcml1, mcmc.input = mcmcSetB, messages = FALSE)
saveRDS(bin.simF, "output/bin.simF.rds")
```

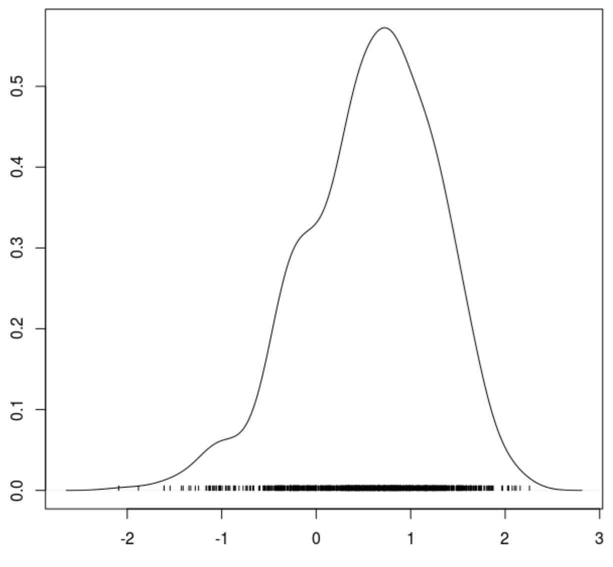
```r
bin.simF <- readRDS("output/bin.simF.rds")
bin.chainConv2 <- create.mcmc.coda(x = bin.simF$simulations[round(runif(1,
    min = 1, max = nrow(bin.simF$simulations)), 0), ], mcmc.input = mcmcSetB)
traceplot(bin.chainConv2)
autocorr.plot(bin.chainConv2, auto.layout = FALSE)
densplot(bin.chainConv2)
geweke.plot(bin.chainConv2, auto.layout = FALSE)
```



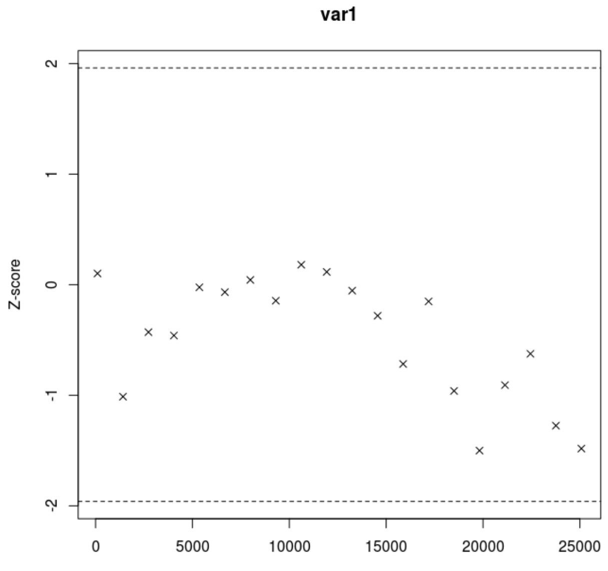
(a) Trace plot of iterations versus sampled values after the burn-in period



(b) The autocorrelation function in each chain

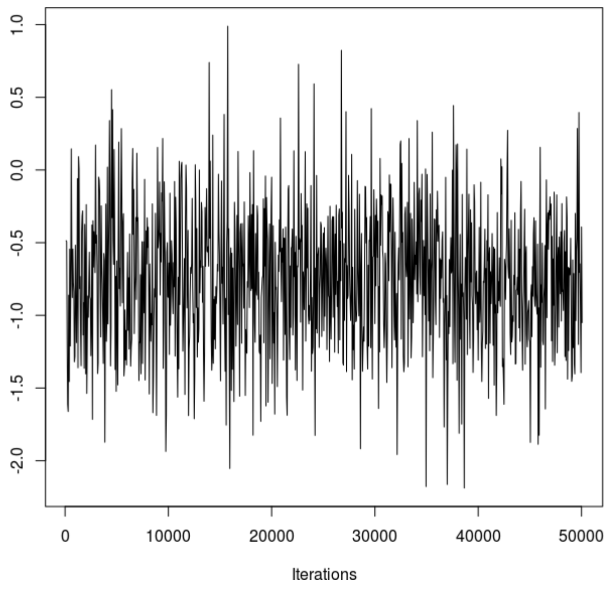


(c) The density plot

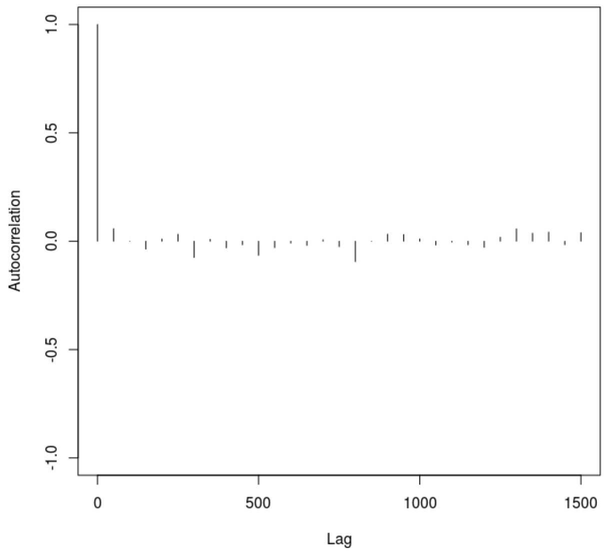


(d) Geweke-Brooks plot

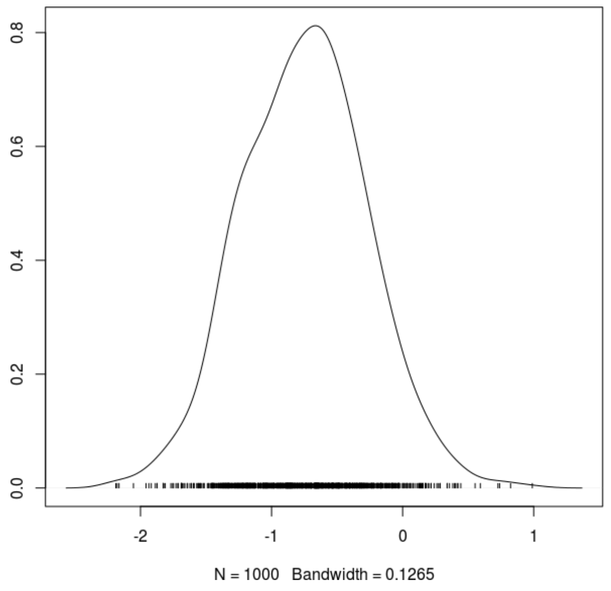
Figure 8: MCMC convergence diagnostics, Poisson model.



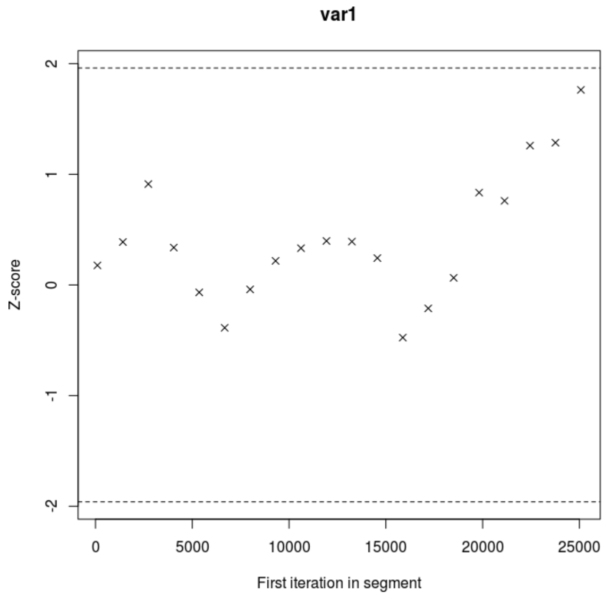
(a) Trace plot of iterations versus sampled values after the burn-in period



(b) The autocorrelation function in each chain



(c) The density plot



(d) Geweke-Brooks plot

Figure 9: MCMC convergence diagnostics, Bernoulli model (second run).

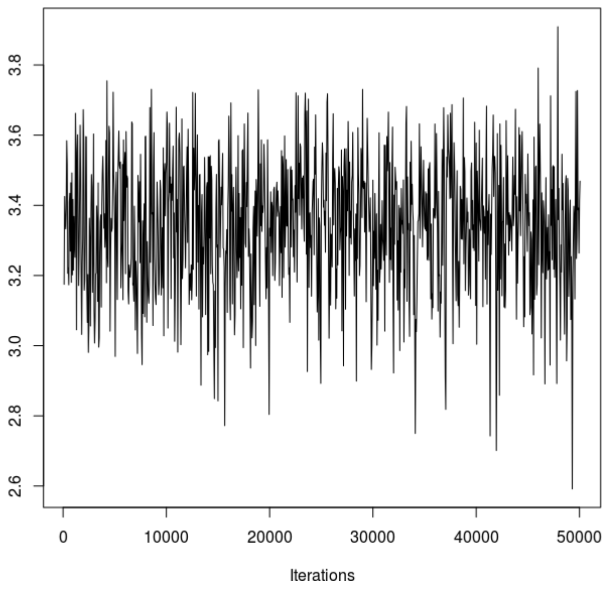
```
bin.mcml1 <- readRDS("output/bin.mcml1.rds")
bin.mcml1

## likfit.glsm: estimated model parameters:
##       beta0       beta1       beta2       beta3       beta4
## "  -0.4659" "   0.0447" "  -0.3249" "   0.4644" "  -0.4337"
##       beta5       beta6       beta7       beta8     sigmasq
## "   0.5401" "   0.0395" "   0.1600" "  -0.0356" "   0.0257"
##         phi   tausq.rel
## "3888.5581" "   7.6885"
##
##  likfit.glsm : maximised log-likelihood = 9.092
```

```
bin.mcmlPrep2 <- prepare.likfit.glsm(bin.simF)
bin.mcml2 <- likfit.glsm(mcmc.obj = bin.mcmlPrep2, cov.model = "spherical",
    ini.phi = bin.mcml1$cov.pars[2], nugget.rel = bin.mcml1$nugget.rel,
    fix.nugget.rel = FALSE, messages = TRUE)
saveRDS(bin.mcml2, "output/bin.mcml2.rds")
bin.simF2 <- glsm.mcmc(geodata = bin.mcmc.geo, units.m = "default",
    model = bin.mcml2, mcmc.input = mcmcSetB, messages = TRUE)
saveRDS(bin.simF2, "output/bin.simF2.rds")
```

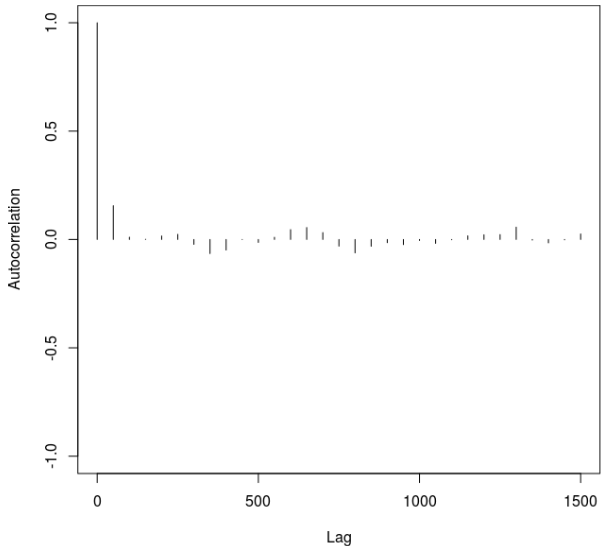
```
bin.mcml2 <- readRDS("output/bin.mcml2.rds")
bin.mcml2

## likfit.glsm: estimated model parameters:
##       beta0       beta1       beta2       beta3       beta4
## "  -0.5009" "  -0.0795" "  -0.2017" "   0.5138" "  -0.4875"
##       beta5       beta6       beta7       beta8     sigmasq
## "   0.5449" "   0.0429" "   0.1295" "  -0.0218" "   0.0419"
##         phi   tausq.rel
## "4294.4802" "   4.9351"
##
##  likfit.glsm : maximised log-likelihood = 15.33
```

7.2 Poisson

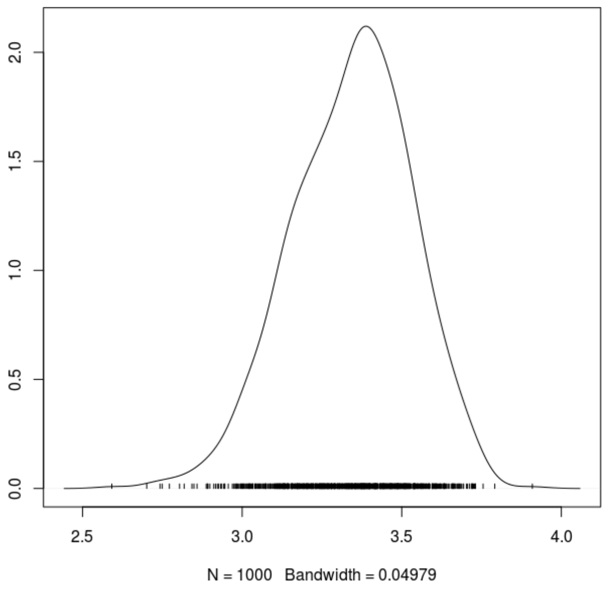
```
pois.mcmlPrep1 <- prepare.likfit.glsm(pois.simFtilde)
pois.mcml1 <- likfit.glsm(mcmc.obj = pois.mcmlPrep1, cov.model = "spherical",
    ini.phi = pois.reml$phi, nugget.rel = (pois.reml$tausq/pois.reml$sigmasq),
    fix.nugget.rel = FALSE, messages = FALSE)
saveRDS(pois.mcml1, "output/pois.mcml1.rds")
pois.simF <- glsm.mcmc(geodata = pois.mcmc.geo, units.m = "default",
    model = pois.mcml1, mcmc.input = mcmcSetP, messages = FALSE)
saveRDS(pois.simF, "output/pois.simF.rds")
```



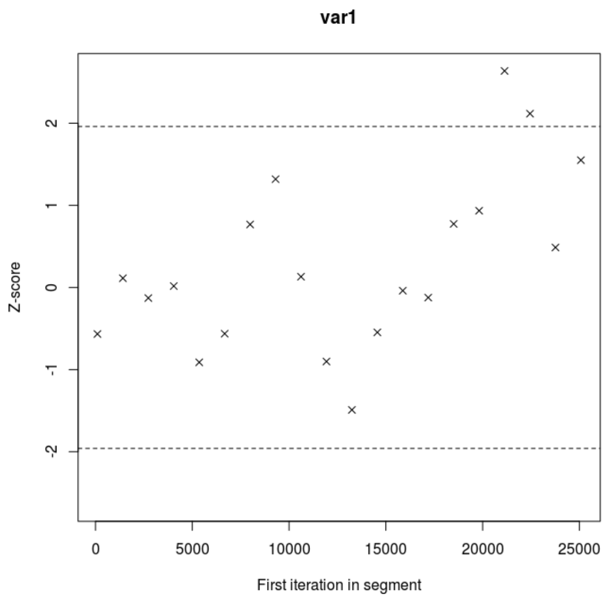
(a) Trace plot of iterations versus sampled values after the burn-in period



(b) The autocorrelation function in each chain



(c) The density plot



(d) Geweke-Brooks plot

Figure 10: MCMC convergence diagnostics, Poisson model (second run).

```
pois.simF <- readRDS("output/pois.simF.rds")
pois.chainConv2 <- create.mcmc.coda(x = pois.simF$simulations[round(runif(1,
    min = 1, max = nrow(pois.simF$simulations)), 0), ], mcmc.input = mcmcSetP)   #
traceplot(pois.chainConv2)
autocorr.plot(pois.chainConv2, auto.layout = FALSE)
densplot(pois.chainConv2)
geweke.plot(pois.chainConv2, auto.layout = FALSE)
```

```
pois.mcml1 <- readRDS("output/pois.mcml1.rds")
pois.mcml1

## likfit.glsm: estimated model parameters:
```

```
##       beta0       beta1       beta2       beta3       beta4
## "  -0.0005" "  -0.0646" "   0.4507" "   0.9892" "  -0.3697"
##       beta5       beta6       beta7       beta8     sigmasq
## "   0.4092" "   0.1392" "   0.1442" "   0.4635" "   0.5701"
##         phi   tausq.rel
## "1979.9621" "   1.7124"
##
##  likfit.glsm : maximised log-likelihood = 341.1
```

```
pois.mcmlPrep2 <- prepare.likfit.glsm(pois.simF)
pois.mcml2 <- likfit.glsm(mcmc.obj = pois.mcmlPrep2, cov.model = "spherical",
    ini.phi = pois.mcml1$cov.pars[2], nugget.rel = pois.mcml1$nugget.rel,
    fix.nugget.rel = FALSE, messages = TRUE)
saveRDS(pois.mcml2, "output/pois.mcml2.rds")
pois.simF2 <- glsm.mcmc(geodata = pois.mcmc.geo, units.m = "default",
    model = pois.mcml2, mcmc.input = mcmc.control(S.scale = 0.2,
        thin = 100, burn.in = 100), messages = TRUE)
saveRDS(pois.simF2, "output/pois.simF2.rds")
```

```
pois.mcml2 <- readRDS("output/pois.mcml2.rds")
pois.mcml2

## likfit.glsm: estimated model parameters:
##       beta0       beta1       beta2       beta3       beta4
## "   0.1015" "  -0.1456" "   0.4634" "   0.9399" "  -0.3767"
##       beta5       beta6       beta7       beta8     sigmasq
## "   0.3665" "   0.1233" "   0.0531" "   0.5536" "   0.3498"
##         phi   tausq.rel
## "2889.6908" "   2.0055"
##
##  likfit.glsm : maximised log-likelihood = 62.01
```

```
pois.mcmlPrep3 <- prepare.likfit.glsm(pois.simF2)
pois.mcml3 <- likfit.glsm(mcmc.obj = pois.mcmlPrep3, cov.model = "spherical",
    ini.phi = pois.mcml2$cov.pars[2], nugget.rel = pois.mcml2$nugget.rel,
    fix.nugget.rel = FALSE, messages = TRUE)
saveRDS(pois.mcml3, "output/pois.mcml3.rds")
pois.simF3 <- glsm.mcmc(geodata = pois.mcmc.geo, units.m = "default",
    model = pois.mcml3, mcmc.input = mcmc.control(S.scale = 0.2,
        thin = 100, burn.in = 100), messages = TRUE)
saveRDS(pois.simF3, "output/pois.simF3.rds")
# rename simulated mcmc and mcml
simF2B <- bin.simF2
mcmlEstimation2B <- bin.mcml2
simF2P <- pois.simF3
mcmlEstimation2P <- pois.mcml3
```

```
pois.mcml3 <- readRDS("output/pois.mcml3.rds")
pois.mcml3

## likfit.glsm: estimated model parameters:
##       beta0       beta1       beta2       beta3       beta4
## "   0.2006" "   0.2475" "   0.0935" "   0.8956" "  -0.2598"
##       beta5       beta6       beta7       beta8     sigmasq
## "   0.3607" "   0.1499" "  -0.0425" "   0.5826" "   0.3067"
##         phi   tausq.rel
## "2603.0983" "   1.6544"
##
##  likfit.glsm : maximised log-likelihood = 25.21
```

8 Prediction

8.1 Prediction grid

```
sgrid <- spsample(intertidal.shp, cellsize = 100, type = "regular",
    offset = c(0.5, 0.5))
proj4string(sgrid) <- RD
depth <- raster("input/depth2.grd")
mindepth <- -200
maxdepth <- 100
grddepth <- over(sgrid, as(depth, "SpatialGridDataFrame"))
ids <- which(grddepth$layer > mindepth & grddepth$layer < maxdepth)
sgrid <- sgrid[ids, ]
grddepth <- grddepth[ids, ]
tmp <- dat
coordinates(tmp) <- ~x + y
proj4string(tmp) <- RD
silt.idw <- idw(silt ~ 1, tmp, sgrid)
silt2.idw <- idw(silt2 ~ 1, tmp, sgrid)
mgs.idw <- idw(mgs ~ 1, tmp, sgrid)
mgs2.idw <- idw(mgs2 ~ 1, tmp, sgrid)
sgrid <- as.data.frame(sgrid)
sgrid$mgs <- mgs.idw$var1.pred
sgrid$mgs2 <- mgs2.idw$var1.pred
sgrid$silt <- silt.idw$var1.pred
sgrid$silt2 <- silt2.idw$var1.pred
sgrid$depth <- (grddepth - meancov["depth"])/sdcov["depth"]
sgrid$depth2 <- (grddepth^2 - meancov["depth2"])/sdcov["depth2"]
sgrid$oost <- (sgrid$x/100 - meancov["oost"])/sdcov["oost"]
sgrid$noord <- (sgrid$y/100 - meancov["noord"])/sdcov["noord"]
sgrid <- sgrid[complete.cases(sgrid), ]
coordinates(sgrid) <- c("x", "y")
proj4string(sgrid) <- RD
```

8.2 Prediction models

```
vgmB <- vgm(model = "Sph", psill = mcmlEstimation2B$cov.pars[1],
    range = mcmlEstimation2B$cov.pars[2], nugget = mcmlEstimation2B$nugget.rel *
        mcmlEstimation2B$cov.pars[1])

vgmP <- vgm(model = "Sph", psill = mcmlEstimation2P$cov.pars[1],
    range = mcmlEstimation2P$cov.pars[2], nugget = mcmlEstimation2P$nugget.rel *
        mcmlEstimation2P$cov.pars[1])
```

8.3 Bernoulli

```
nsm<-100
predB<-varB<-NULL
for (i in 1:nsm){
S.sibesB<-as.data.frame(cbind(
                              simF2B$geodata$coords,
                              simF2B$simulations[,i],
                              simF2B$geodata$covariate))
names(S.sibesB)<-c("x","y","S", names(simF2B$geodata$covariate))
coordinates(S.sibesB)<-~x+y
proj4string(S.sibesB)<-RD
S.krige.predB <- krige(
    formula = S~mgs+mgs2+silt+silt2+depth+depth2+oost+noord,
    locations = S.sibesB,
    newdata=sgrid,
    model = vgmB,
    beta = mcmlEstimation2B$beta,
    nmax=100,
    nsim=0, #use kriging interpolation
    debug.level=1
)
#prediction
predB<-cbind(predB,S.krige.predB$var1.pred)
#kriging var
varB<-cbind(varB,S.krige.predB$var1.var)
}
colnames(predB)<-paste(rep("S",ncol(predB)),c(1:ncol(predB)),sep="")
colnames(varB)<-paste(rep("S",ncol(varB)),c(1:ncol(varB)),sep="")
predB<-data.frame(sgrid@coords, predB)
varB<-data.frame(sgrid@coords, varB)
saveRDS(object=predB,file="./output/predB.rds")
saveRDS(object=varB,file="./output/varB.rds")
```

8.4 Poisson

```
nsm<-100
predP<-varP<-NULL
for (i in 1:nsm){
S.sibesP<-as.data.frame(cbind(
                              simF2P$geodata$coords,
                              simF2P$simulations[,i],
                              simF2P$geodata$covariate))
names(S.sibesP)<-c("x","y","S", names(simF2P$geodata$covariate))
coordinates(S.sibesP)<-~x+y
proj4string(S.sibesP)<-RD
S.krige.predP <- krige(
    formula = S~mgs+mgs2+silt+silt2+depth+depth2+oost+noord,
locations = S.sibesP,
newdata=sgrid,
model = vgmP,
beta = mcmlEstimation2P$beta,
nmax=100,
nsim=0,
debug.level=1
)
predP<-cbind(predP,S.krige.predP$var1.pred)
varP<-cbind(varP,S.krige.predP$var1.var)
}
colnames(predP)<-paste(rep("S",ncol(predP)),c(1:ncol(predP)),sep="")
colnames(varP)<-paste(rep("S",ncol(varP)),c(1:ncol(varP)),sep="")
predP<- data.frame(sgrid@coords, predP)
varP<-data.frame(sgrid@coords, varP)
saveRDS(object=predP,file="./output/predP.rds")
saveRDS(object=varP,file="./output/varP.rds")
```

8.5 Transformation of kriging prediction and kriging variance

```
## ----pi.pred.mean pi.var.mean
pi.pred <- mapply(transf.predB, var1.pred = predB[, -c(1:2)],
    var1.var = varB[, -c(1:2)])
# dim(pi.pred) 115145 100
pi.var <- mapply(transf.varB, var1.pred = predB[, -c(1:2)], var1.var = varB[,
    -c(1:2)])
# take average over 100 S
pi.pred.mean <- data.frame(predB[, c(1:2)], pi.pred.mean = rowMeans(pi.pred))
# predicted variance is obtained by taking the mean of the
# predicted variance plus the variance of the predicted means
pi.var.mean <- data.frame(varB[, c(1:2)], pi.var.mean = rowMeans(pi.var) +
    apply(pi.pred, 1, var))
saveRDS(pi.pred.mean, file = "./output/pi.pred.mean.rds")
saveRDS(pi.var.mean, file = "./output/pi.var.mean.rds")
## ----mu.pred.mean mu.var.mean
mu.pred <- mapply(transf.predP, var1.pred = predP[, -c(1:2)],
    var1.var = varP[, -c(1:2)])
# dim(mu.pred) 115145 100
mu.var <- mapply(transf.varP, var1.pred = predP[, -c(1:2)], var1.var = varP[,
    -c(1:2)])
# take average over 100 S
mu.pred.mean <- data.frame(predP[, c(1:2)], mu.pred.mean = rowMeans(mu.pred))
```

```r
mu.var.mean <- data.frame(varP[, c(1:2)], mu.var.mean = rowMeans(mu.var) +
    apply(mu.pred, 1, var))
saveRDS(mu.pred.mean, file = "./output/mu.pred.mean.rds")
saveRDS(mu.var.mean, file = "./output/mu.var.mean.rds")
## ----pimu.pred pimu.var
pimu.pred <- pi.pred * mu.pred
pimu.pred <- rowMeans(pimu.pred)
pimu.pred <- data.frame(predP[, c(1:2)], pimu.pred)
varpi <- apply(pi.pred, 1, var)
varmu <- apply(mu.pred, 1, var)
Epi2 <- apply(pi.pred, 1, function(x) {
    mean(x)^2
})
Emu2 <- apply(mu.pred, 1, function(x) {
    mean(x)^2
})
pimu.var <- varpi * varmu + varpi * Emu2 + varmu * Epi2
# coefficient of variation sqrt(var)/mean=sd/mean
pimu.cv <- sqrt(pimu.var)/pimu.pred[, 3]
pimu.cv <- data.frame(predP[, c(1:2)], pimu.cv)
saveRDS(pimu.pred, file = "./output/pimu.pred.rds")
saveRDS(pimu.cv, file = "./output/pimu.cv.rds")
```

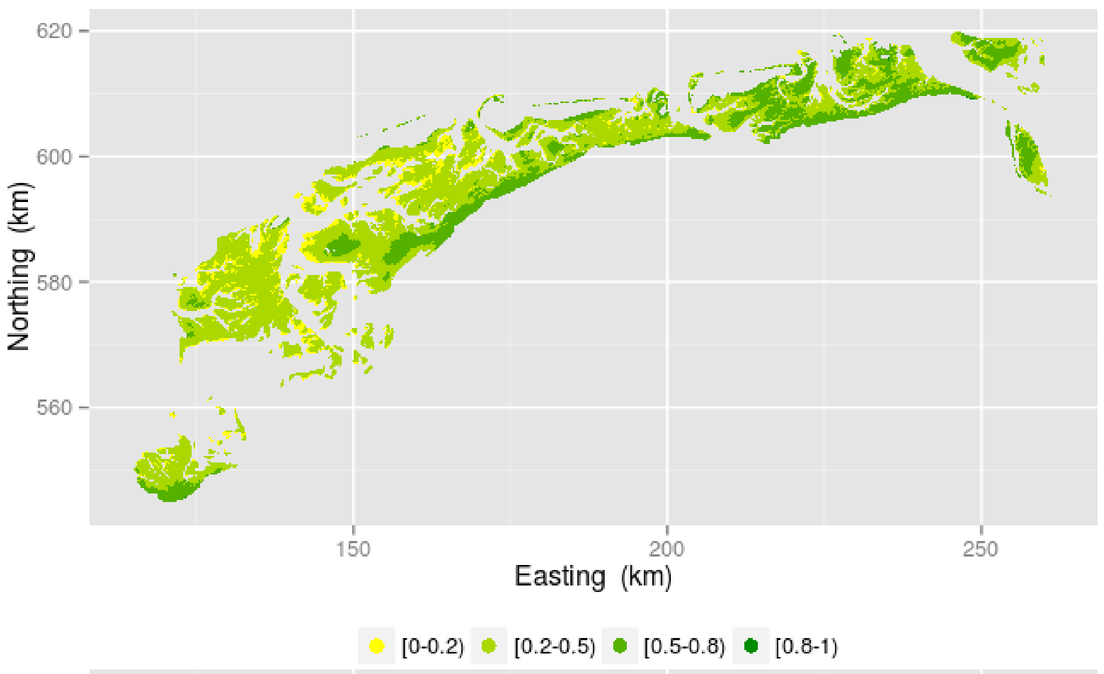
8.6 Prediction maps

```r
pi.pred.mean <- readRDS("./output/pi.pred.mean.rds")
pi.var.mean <- readRDS("./output/pi.var.mean.rds")
mu.pred.mean <- readRDS("./output/mu.pred.mean.rds")
mu.var.mean <- readRDS("./output/mu.var.mean.rds")
pimu.pred <- readRDS("./output/pimu.pred.rds")
pimu.cv <- readRDS("./output/pimu.cv.rds")
```

```r
##----pi.pred.mean
df<-pi.pred.mean
mean.pi.pred.mean<-(df[,3])
quantile(df$pi.pred.mean)

##        0%       25%       50%       75%      100%
## 0.0485152 0.2619845 0.3586531 0.5059025 0.8269089

b<-c(-Inf,0.2,0.5,0.8,Inf)
l<-c("[0-0.2)", "[0.2-0.5)","[0.5-0.8)", "[0.8-1)")
colo<-c("#FFFF00", "#AAD800", "#55B100", "#008B00")
df$group<-cut(pi.pred.mean[,3],breaks=b, labels=l, right=F) #[)
table(df$group)

##
##   [0-0.2) [0.2-0.5) [0.5-0.8)   [0.8-1)
##      9805     75360     29820        38
```

```
gg.pi.pred<-ggplot(
data = df,
aes( x = x* 1.0e-3, y = y* 1.0e-3, color=group))+
geom_point(size=0.2)+
scale_color_manual(name="", values=colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.pi.pred
##----pi.var.mean
df<-pi.var.mean
df$pi.var.mean<-(sqrt(df$pi.var.mean))/mean.pi.pred.mean
quantile(df$pi.var.mean)

##         0%        25%        50%        75%       100%
## 0.08381421 0.24232223 0.31361468 0.35986880 0.45291174

b<-c(-Inf,0.2,0.3,Inf)
l<-c("[0-0.01)","[0.01-0.1)", "[0.1-Inf)")
colo=c("#CDC673", "#AC744B", "#8B2323")
df$group<-cut(df$pi.var.mean,breaks=b, labels=l, right=F)
table(df$group)

##
##   [0-0.01) [0.01-0.1)  [0.1-Inf)
##      13586      37530      63907

gg.pi.var<-ggplot(
                  data = df,
                  aes( x = x* 1.0e-3, y = y* 1.0e-3, color=group))+
geom_point(size=0.2)+
scale_color_manual(name="", values=colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.pi.var
```

```
##----mu.pred.mean
df<-mu.pred.mean
mean.mu.pred.mean<-df[,3]
quantile(mu.pred.mean[,3])

##         0%        25%        50%        75%       100%
##  0.2234698  1.0226871  1.7785331  2.9108882 46.5276863

b<-c(-Inf,1,2,5, Inf)
l<-c("[0-1)", "[1-2)","[2-5)", "[5-46)")
```



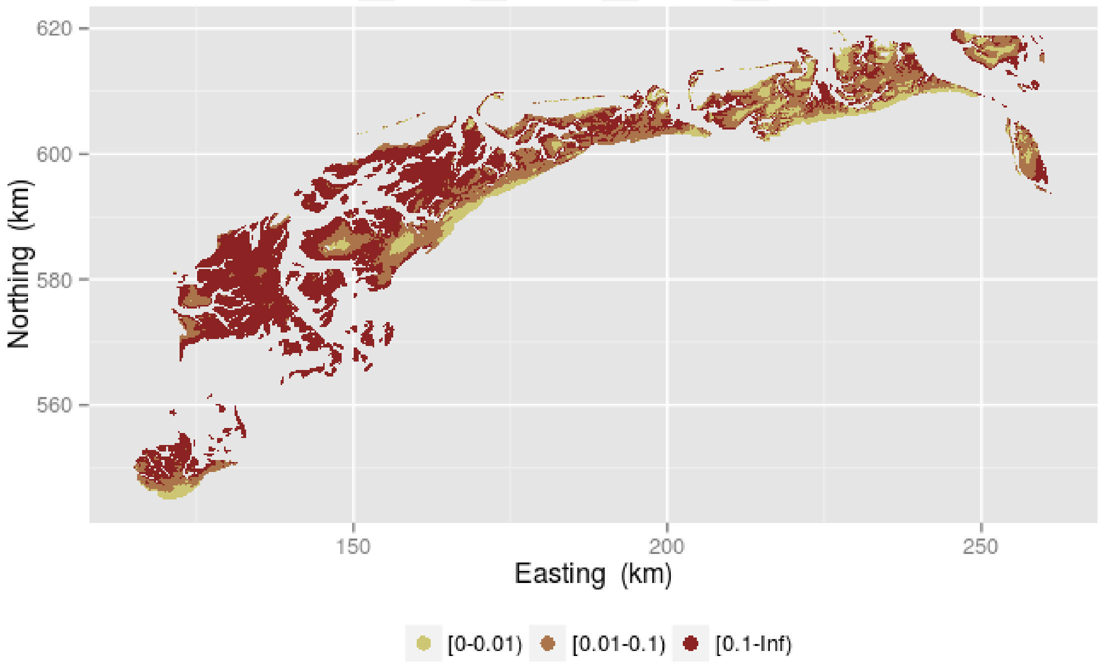
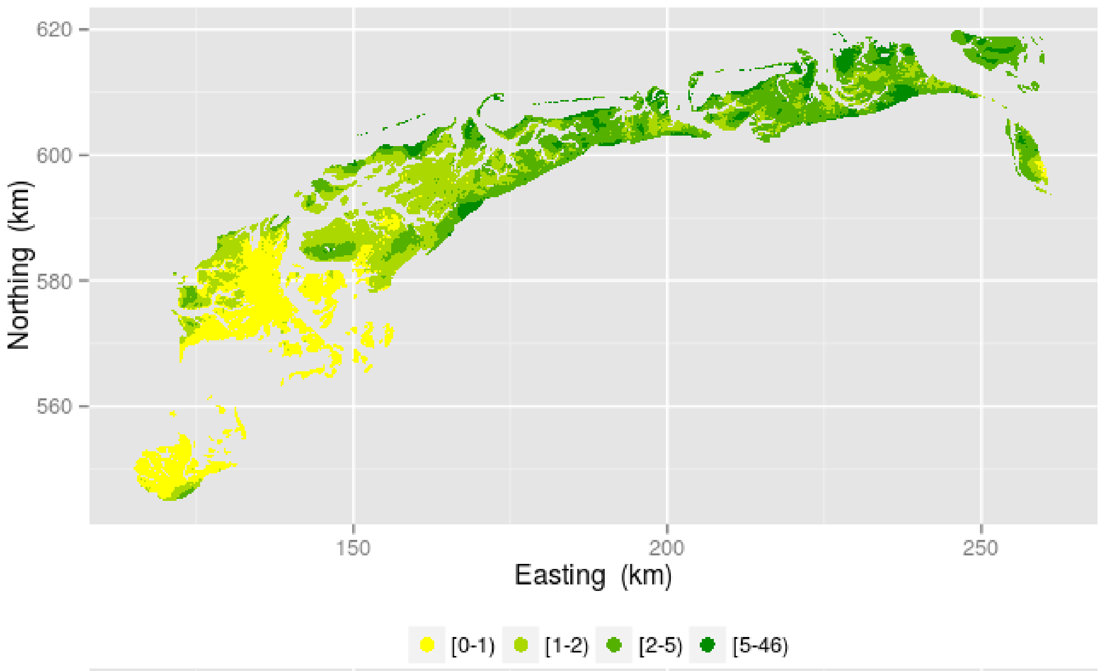


Figure 11: Predicted prevalence (a) and standard deviation of predicted prevalence divided by mean of predicted prevalence (b). Average of 100 realisations.

```
colo<-c("#FFFF00", "#AAD800", "#55B100", "#008B00")
df$group<-cut(mu.pred.mean[,3],breaks=b, labels=l, right=F) #[)
table(df$group)

##
##  [0-1)  [1-2)  [2-5) [5-46)
##  28004  36790  41071   9158

gg.mu.pred<-ggplot(
                   data = df,
                   aes( x = x* 1.0e-3, y = y* 1.0e-3, colour=group))+
geom_point(size=0.2)+
scale_colour_manual(name="", values = colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.mu.pred
df<-mu.var.mean
df$mu.var.mean<-(sqrt(df$mu.var.mean))/mean.mu.pred.mean
quantile(df$mu.var.mean)

##        0%       25%       50%       75%      100%
## 0.9109574 1.0053214 1.0495971 1.0913604 1.1367460

b<-c(-Inf,1,1.1,Inf)
l<-c("[0-1)","[1-1.1)", "[1.1-Inf)")
colo<-c("#CDC673","#AC744B", "#8B2323")
df$group<-cut(df$mu.var.mean,breaks=b, labels=l, right=F)
table(df$group)

##
##     [0-1)   [1-1.1) [1.1-Inf)
##     25437     66525     23061

gg.mu.var<-ggplot(
                  data = df,
                  aes( x = x* 1.0e-3, y = y* 1.0e-3, colour=group))+
geom_point(size=0.2)+
scale_colour_manual(name="", values = colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.mu.var
```

```
#pi*mu pred
df<-pimu.pred
quantile(df[,3])
```



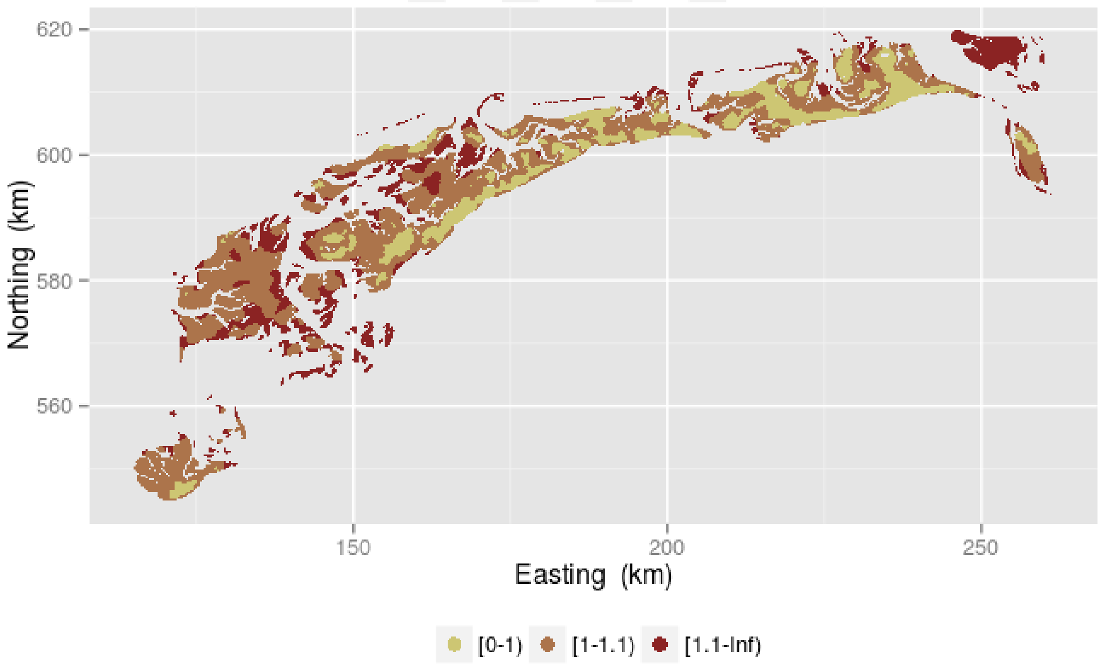


Figure 12: Predicted intensity and standard deviation of predicted intensity divided by mean of predicted intensity Average of 100 realisations.

```
##          0%          25%          50%          75%         100%
##  0.02571282  0.26532174  0.60909577  1.36936668 29.80746344

b<-c(-Inf,0.3,0.7,1.5, Inf)
l<-c("[0-0.3)", "[0.3-0.7)","[0.7-1.5)", "[1.5-30)")
colo<-c("#FFFF00", "#AAD800", "#55B100", "#008B00")
df$group<-cut(df[,3],breaks=b, labels=l, right=F)
table(df$group)

##
##   [0-0.3) [0.3-0.7) [0.7-1.5)  [1.5-30)
##     33308     28759     27423     25533

gg.pimu.pred<-ggplot(data = df,
                     aes( x = x* 1.0e-3, y = y* 1.0e-3, colour=group))+
geom_point(size=0.2)+
scale_colour_manual(name="", values = colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.pimu.pred
#pi*mu var
df<-pimu.cv
quantile(df[,3])

##        0%       25%       50%       75%      100%
## 0.0000000 0.1574969 0.1974478 0.2365296 0.3973716

b<-c(-Inf,0.20,0.25,0.30, Inf)
l<-c("[0-0.20)", "[0.20-0.25)","[0.25-0.30)", "[0.30-0.40)")
colo<-c("#CDC673","#B78F58","#A1593D","#8B2323")
df$group<-cut(df[,3],breaks=b, labels=l, right=F)
table(df$group)

##
##    [0-0.20) [0.20-0.25) [0.25-0.30) [0.30-0.40)
##       59504       34799       17276        3444

gg.pimu.cv<-ggplot(data = df,
                   aes( x = x* 1.0e-3, y = y* 1.0e-3, colour=group))+
geom_point(size=0.2)+
scale_colour_manual(name="", values = colo)+
scale_x_continuous(name = "Easting  (km)") +
scale_y_continuous(name = "Northing  (km)") +
coord_equal(ratio = 1)+
theme(legend.position="bottom")+
guides(colour = guide_legend(override.aes = list(size=3)))
gg.pimu.cv
```

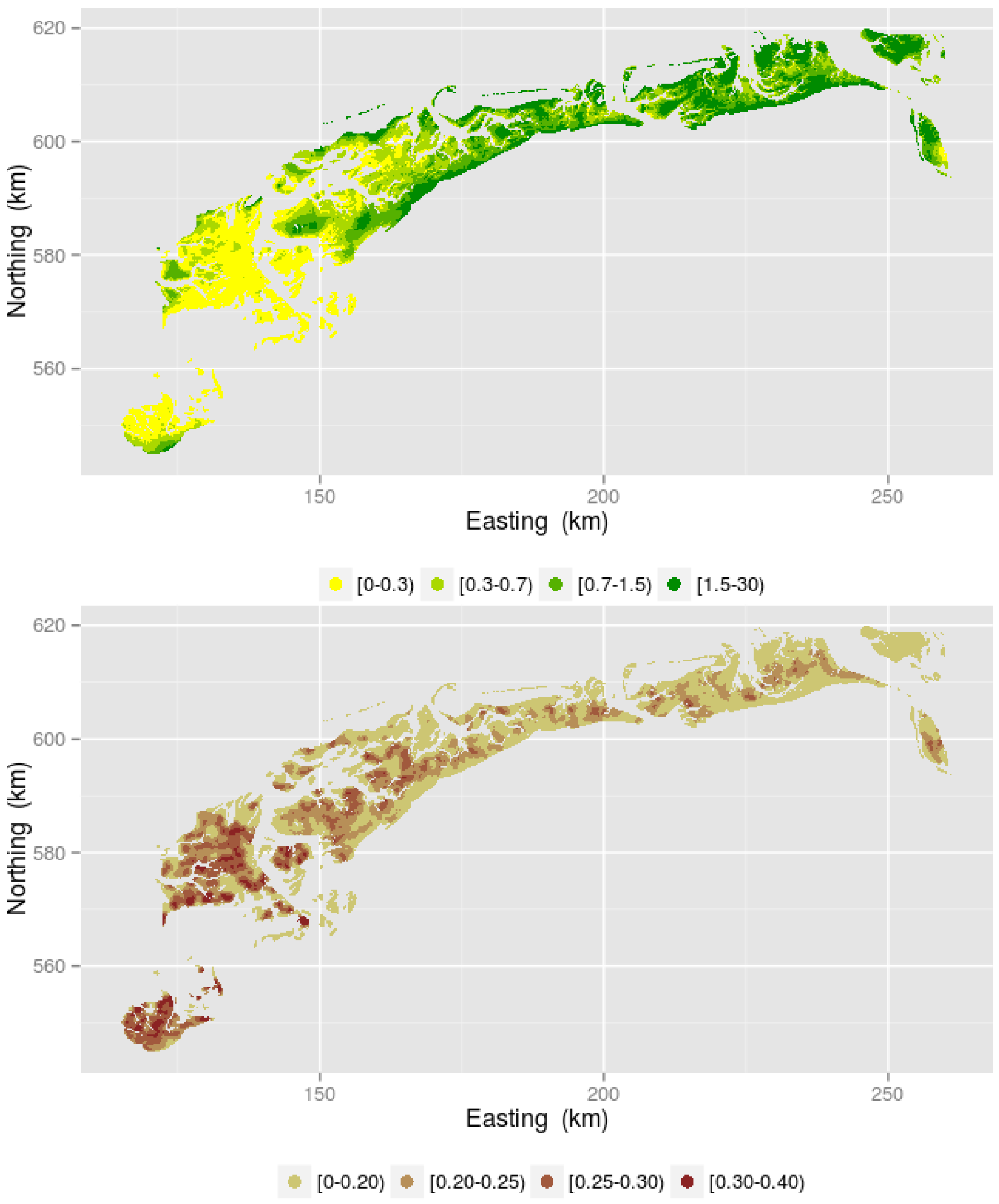


Figure 13: Predicted unconditional intensity and standard deviation of predicted unconditional intensity divided by mean of predicted unconditonal intensity. Average of 100 realisations.

9 Cross-validation

9.1 Predicted prevalence

```
nsm <- 100
xvalB <- NULL
for (i in 1:nsm) {
    S.sibesB <- as.data.frame(cbind(simF2B$geodata$coords, simF2B$simulations[,
        i], simF2B$geodata$covariate))
    names(S.sibesB) <- c("x", "y", "S", names(simF2B$geodata$covariate))
    xvalB[[i]] <- krige.cv(formula = S ~ mgs + mgs2 + silt +
        silt2 + depth + depth2 + oost + noord, data = S.sibesB,
        locations = ~x + y, model = vgmB, beta = mcmlEstimation2B$beta,
        nmax = 100, nfold = nrow(S.sibesB), debug.level = 1,
        verbose = TRUE)
}
# save a list of dataframes
saveRDS(object = xvalB, file = "./output/xvalB.rds")
```

```
xvalB <- readRDS("./output/xvalB.rds")
nsm <- 100
pi.obs <- pi.pr <- sb.obs <- sb.pr <- matrix(nrow = nrow(xvalB[[1]]),
    ncol = nsm)
for (i in 1:nsm) {
    # signal
    sb.obs[, i] <- xvalB[[i]]$observed
    sb.pr[, i] <- xvalB[[i]]$var1.pred
    # pi and mu
    pi.obs[, i] <- antilogit(xvalB[[i]]$observed)
    pi.pr[, i] <- transf.predB(xvalB[[i]]$var1.pred, xvalB[[i]]$var1.var)
}
# identify bernoulli locations
bin.dat.obs <- readRDS("output/dat.bin.rds")
bin.dat.obs <- bin.dat.obs$data
bin.pr <- ifelse(rowMeans(pi.pr) > 0.5, 1, 0)
# confusion matrix
table(bin.dat.obs, bin.pr)

##            bin.pr
## bin.dat.obs    0    1
##           0 1978  415
##           1  901  732

# x obs pred
x00 <- table(bin.dat.obs, bin.pr)[[1]]
x01 <- table(bin.dat.obs, bin.pr)[[3]]
x10 <- table(bin.dat.obs, bin.pr)[[2]]
x11 <- table(bin.dat.obs, bin.pr)[[4]]
# correctly predicted 0
x00/(x00 + x01)
```

```
## [1] 0.8265775

# correctly predicted 1
x11/(x11 + x10)

## [1] 0.4482547

# overall purity (accuracy)
(x11 + x00)/(x00 + x01 + x10 + x11)

## [1] 0.6731247

# map unit purity (users accuracy) present
x11/(x11 + x01)

## [1] 0.6381866

# absent
x00/(x00 + x10)

## [1] 0.6870441

# class representation (producer's accuracy) present
x11/(x10 + x11)

## [1] 0.4482547

# absent
x00/(x00 + x01)

## [1] 0.8265775
```

9.2 Predicted intensity

```
nsm <- 100
xvalP <- vector("list", length = nsm)
for (i in 1:nsm) {
    S.sibesP <- as.data.frame(cbind(simF2P$geodata$coords, simF2P$simulations[,
        i], simF2P$geodata$covariate))
    names(S.sibesP) <- c("x", "y", "S", names(simF2P$geodata$covariate))
    xvalP[[i]] <- krige.cv(formula = S ~ mgs + mgs2 + silt +
        silt2 + depth + depth2 + oost + noord, data = S.sibesP,
        locations = ~x + y, model = vgmP, beta = mcmlEstimation2P$beta,
        nmax = 100, nfold = nrow(S.sibesP), debug.level = 1,
        verbose = TRUE)
}
saveRDS(object = xvalP, file = "./output/xvalP.rds")
```

```
xvalP <- readRDS("./output/xvalP.rds")
mu.obs <- mu.pr <- sp.obs <- sp.pr <- matrix(nrow = nrow(xvalP[[1]]),
    ncol = nsm)
for (i in 1:nsm) {
    sp.obs[, i] <- xvalP[[i]]$observed
    sp.pr[, i] <- xvalP[[i]]$var1.pred
    mu.obs[, i] <- exp(xvalP[[i]]$observed)
    mu.pr[, i] <- transf.predP(xvalP[[i]]$var1.pred, xvalP[[i]]$var1.var)
}
pois.dat.obs <- readRDS("output/dat.pois.rds")
pois.dat.obs <- pois.dat.obs$data
var(pois.dat.obs)

## [1] 52.0103

me.mu <- mean(rowMeans(mu.pr) - pois.dat.obs)
me.mu

## [1] -0.1937722

mse.mu <- mean((rowMeans(mu.pr) - pois.dat.obs)^2)
mse.mu

## [1] 34.91769

rpd <- sqrt(var(pois.dat.obs))/sqrt(mse.mu)
rpd

## [1] 1.220455
```

9.3 Predicted unconditional intensity

```
# Cross-validation of pi*mu (predicted unconditional expected
# count)
dat.obs <- read.csv("input/dat.csv", header = T)
# remove 4 values in macoma >100
dat.obs <- subset(dat.obs, macoma < 100)
# variance in data, we hope that mse smaller than var(obs),
# meaning that model is useful, if not a simple average is
# enough
var(dat.obs$macoma)
# find locations of bernoulli zeros
bin.dat.obs <- readRDS("output/dat.bin.rds")
# covert to df
bin.dat.obs <- cbind(bin.dat.obs$coords, bin.dat.obs$data, bin.dat.obs$covariate)
names(bin.dat.obs)[3] <- "macoma"
xy.sp.pr2 <- subset(bin.dat.obs, (!bin.dat.obs$x %in% xvalP[[1]]$x &
    !bin.dat.obs$y %in% xvalP[[1]]$y), select = -3)
# read signal and model
simF2P <- readRDS("output/pois.simF3.rds")
mcmlEstimation2P <- readRDS("output/pois.mcml3.rds")
vgmP <- vgm(model = "Sph", psill = mcmlEstimation2P$cov.pars[1],
    range = mcmlEstimation2P$cov.pars[2], nugget = mcmlEstimation2P$nugget.rel *
        mcmlEstimation2P$cov.pars[1])
# predict poisson signal at bernoulli locations
nsm <- 100
xvalP2 <- vector("list", length = nsm)
for (i in 1:nsm) {
    S.sibesP <- as.data.frame(cbind(simF2P$geodata$coords, simF2P$simulations[,
        i], simF2P$geodata$covariate))
    names(S.sibesP) <- c("x", "y", "S", names(simF2P$geodata$covariate))
    xvalP2[[i]] <- krige(formula = S ~ mgs + mgs2 + silt + silt2 +
        depth + depth2 + oost + noord, data = S.sibesP, locations = ~x +
        y, newdata = xy.sp.pr2, model = vgmP, beta = mcmlEstimation2P$beta,
        nmax = 100, nsim = 0, debug.level = 1)
}
saveRDS(object = xvalP2, file = "./output/xvalP2.rds")
# reshape to matrix
mu.pr2 <- matrix(nrow = nrow(xvalP2[[1]]), ncol = nsm)
for (i in 1:nsm) {
    mu.pr2[, i] <- transf.predP(xvalP2[[i]]$var1.pred, xvalP2[[i]]$var1.var)
}
# add coordinates
xy.mu.pr2 <- cbind(xvalP2[[1]]$x, xvalP2[[1]]$y, mu.pr2)
xy.mu.pr1 <- cbind(xvalP[[1]]$x, xvalP[[1]]$y, mu.pr)
xy.pi.pr <- cbind(xvalB[[1]]$x, xvalB[[1]]$y, pi.pr)
# split predicted pi (xy.pi.pr) in 2 subsets 1633 poisson loc
xy.pi.pr1 <- subset(xy.pi.pr, (xy.pi.pr[, 1] %in% xy.mu.pr1[,
    1] & xy.pi.pr[, 2] %in% xy.mu.pr1[, 2]))
# 2393 bernoulli 0
xy.pi.pr2 <- subset(xy.pi.pr, (xy.pi.pr[, 1] %in% xy.mu.pr2[,
    1] & xy.pi.pr[, 2] %in% xy.mu.pr2[, 2]))
# split observed sibes
obs1 <- subset(dat.obs, (dat.obs$x %in% xvalP[[1]]$x & dat.obs$y %in%
    xvalP[[1]]$y), select = macoma)
obs2 <- subset(dat.obs, (!dat.obs$x %in% xvalP[[1]]$x & !dat.obs$y %in%
    xvalP[[1]]$y), select = macoma)
# averages of 100 products
pimu.pr1 <- rowMeans(xy.pi.pr1[, -c(1:2)] * xy.mu.pr1[, -c(1:2)])
pimu.pr2 <- rowMeans(xy.pi.pr2[, -c(1:2)] * xy.mu.pr2[, -c(1:2)])
me.pimu <- mean(rbind((pimu.pr1 - as.matrix(obs1)), (pimu.pr2 -
    as.matrix(obs2))))
me.pimu
## -0.18
mse.pimu <- mean(rbind(((pimu.pr1 - obs1)^2), ((pimu.pr2 - obs2)^2)))
mse.pimu
## 17.53
rpd <- sqrt(var(dat.obs$macoma))/sqrt(mse.pimu)
rpd
```

10 Effect of grid spacing

10.1 Select one simulated field

```r
# select X on 100 m that best resembles sibes function to
# define criteria to select simulated field that resembles
# sibes criteria 1: fraction of zeros
f1 <- function(x) {
    table(x > 1)[1]/length(x)
}
# criteria 2: mean of non zero
f2 <- function(x) {
    mean(subset(x, x != 0))
}
# criteria 3: var of non zero
f3 <- function(x) {
    var(subset(x, x != 0))
}
# load scaled sibes data
dat <- read.csv(file = "./input/datsc.csv")
r1 <- apply(dat[2], 2, f1)
# 0.80
r2 <- apply(dat[2], 2, f2)
# 4.13
r3 <- apply(dat[2], 2, f3)
# 59.24 load simulated signals on 100 m grid from
# paper1/R4/output/predB or predP
predB <- readRDS("./input/predB.rds")
predP <- readRDS("./input/predP.rds")
varB <- readRDS("./input/varB.rds")
varP <- readRDS("./input/varP.rds")
# transform kriging prediction
pred.pi <- mapply(transf.predB, var1.pred = predB[, -c(1:2)],
    var1.var = varB[, -c(1:2)])
var.pi <- mapply(transf.varB, var1.pred = predB[, -c(1:2)], var1.var = varB[,
    -c(1:2)])  #dont use?
pred.mu <- mapply(transf.predP, var1.pred = predP[, -c(1:2)],
    var1.var = varP[, -c(1:2)])
var.mu <- mapply(transf.varP, var1.pred = predP[, -c(1:2)], var1.var = varP[,
    -c(1:2)])  #dont use?
# obtain 100 field with counts
pred.count <- myfun.count(pred.pi, pred.mu)
# apply criteria
r1X <- apply(pred.count, 2, f1)
r2X <- apply(pred.count, 2, f2)
r3X <- apply(pred.count, 2, f3)
# divide abs difference by standard deviation
r1Xa <- abs(r1 - r1X)/(sd(abs(r1 - r1X)))
r2Xa <- abs(r2 - r2X)/(sd(abs(r2 - r2X)))
r3Xa <- abs(r3 - r3X)/(sd(abs(r3 - r3X)))
# find sum of absolute differences assign weights 0.5, 0.25,
# 0.25 and select min
rX <- 0.5 * r1Xa + 0.25 * r2Xa + 0.25 * r3Xa
which.min(rX)
# 38
```

10.2 Select validation points through stratified sampling

```r
## prediction grid with covariates
sgrid <- readRDS("input/sgrid.small.rds")
## set aside 1000 points based on tidal basins load tidal
## basins
tidalbasins.shp <- readOGR("./input/tidalbasins", "tidalbasins")
## reproject tidal basin
tidalbasins.shp <- spTransform(tidalbasins.shp, RD)
tb <- over(sgrid, tidalbasins.shp)
sgriddf <- as.data.frame(sgrid)
sgriddf$tb <- tb$TB
## compute size (number of points) of strata
Ntb <- table(tb$TB)
N <- sum(Ntb)
## compute sample sizes (proportional allocation)
n <- 1000
# ntb<-round(Ntb/N*n)
sum(ntb)
## in stratum 1 only 1 point, change this to 2.
ids <- which(ntb < 2)
ntb[ids] <- 2
sumn <- sum(ntb)
ids <- which(ntb == max(ntb))
ntb[ids] <- ntb[ids] - (sum(ntb) - n)
sum(ntb)
stratumid <- sort(unique(tb$TB))
val <- NULL
## set aside 1000 points using tidal basins as strata
for (i in stratumid) {
    sam <- sgriddf[myfun.sample(which(sgriddf$tb == i), ntb[[as.character(i)]],
        replace = FALSE), ]
    val <- rbind(val, sam)
}
## val.id
sgriddf$val.id <- row.names(sgriddf) %in% row.names(val)
# saveRDS(sgriddf, 'output/sgriddf.rds')
sgriddf <- readRDS("output/sgriddf.rds")
val.xy <- subset(sgriddf, val.id == T, select = c(x1, x2))
# use avg of 100 realistaion instead of S38
predBdf <- data.frame(S = rowMeans(predB[, -c(1:2)]), sgriddf)
# add covariates
predBdf <- data.frame(S = predB$S38, sgriddf)
varBdf <- data.frame(S = varB$S38, sgriddf)
predPdf <- data.frame(S = predP$S38, sgriddf)
```

```r
varPdf <- data.frame(S = varP$S38, sgriddf)
## prediction dataset
grdB <- predBdf[which(predBdf$val.id == F), ]
grdB.var <- varBdf[which(varBdf$val.id == F), ]
grdP <- predPdf[which(predPdf$val.id == F), ]
grdP.var <- varPdf[which(varPdf$val.id == F), ]
# validation dataset
valB <- predBdf[which(predBdf$val.id == T), ]
valB.var <- varBdf[which(varBdf$val.id == T), ]
valP <- predPdf[which(predPdf$val.id == T), ]
valP.var <- varPdf[which(varPdf$val.id == T), ]
# assign coordinates
coordinates(valB) <- ~x1 + x2
proj4string(valB) <- RD
coordinates(valP) <- ~x1 + x2
proj4string(valP) <- RD
coordinates(grdB) <- ~x1 + x2
proj4string(grdB) <- RD
coordinates(grdP) <- ~x1 + x2
proj4string(grdP) <- RD
gridded(grdB) <- TRUE
gridded(grdP) <- TRUE
saveRDS(grdB, file = "output/grdB.rds")
saveRDS(grdB.var, file = "output/grdB.var.rds")
saveRDS(grdP, file = "output/grdP.rds")
saveRDS(grdP.var, file = "output/grdP.var.rds")
saveRDS(valB, file = "output/valB.rds")
saveRDS(valB.var, file = "output/valB.var.rds")
saveRDS(valP, file = "output/valP.rds")
saveRDS(valP.var, file = "output/valP.var.rds")
```

```r
val.xy <- readRDS("output/val.xy.rds")
gg.intertidal + geom_point(data = val.xy, mapping = aes(x = x1/1000,
    y = x2/1000), size = 0.8, colour = "red", shape = 3)
```

10.3 Prediction with fixed model parameters

10.3.1 Model setup

```r
# MCML parameters evaluated on 100m grid
mcmlEstimation2B <- readRDS("input/bin.mcml2.rds")
mcmlEstimation2P <- readRDS("input/pois.mcml3.rds")
# define trend and model
trendB <- trendP <- S ~ silt + silt2 + depth
vgmB <- vgm(model = "Sph", psill = mcmlEstimation2B$cov.pars[1],
    range = mcmlEstimation2B$cov.pars[2], nugget = mcmlEstimation2B$nugget.rel *
        mcmlEstimation2B$cov.pars[1])
vgmP <- vgm(model = "Sph", psill = mcmlEstimation2P$cov.pars[1],
    range = mcmlEstimation2P$cov.pars[2], nugget = mcmlEstimation2P$nugget.rel *
        mcmlEstimation2P$cov.pars[1])
```

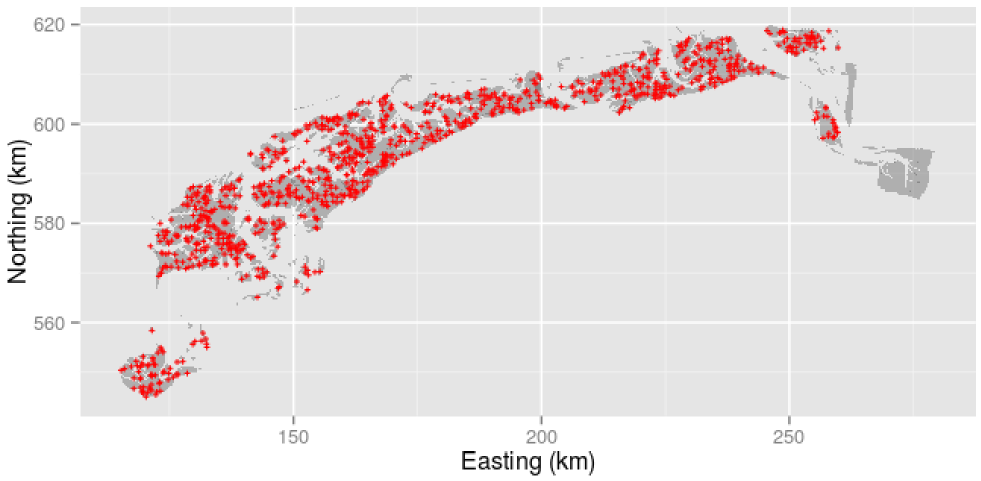


Figure 14: 1000 validation points allocated through stratified sampling using tidal basins as strata

10.3.2 Prediction Bernoulli

```
set.seed(22800)
#grid spacing
spacing<-2^seq(1,7,1)*100
# steekrproef number
nsteek<-100
seB<-seP<-se.pi<-se.mu<-prB<-prP<-array(dim=c(length(spacing), nsteek, 1000))
pr.pi<-pr.mu<-var.pi<-var.mu<-array(dim=c(length(spacing), nsteek, 1000))
npointsB<-npointsP<-array(dim=c(length(spacing),nsteek))
#binomial
for(k in 1:length(spacing)){
for (j in 1:nsteek){
s<- spsample(grdB, cellsize=spacing[k],type="regular")
npt<-length(s)
o<-over(s, grdB)
sam<-data.frame(s,o)
names(sam)[1:2]<-c("x1","x2")
sam<-na.omit(sam)
coordinates(sam)<-~x1+x2
proj4string(sam)<-RD
dataB<-sam
tmpB <- krige(
  formula = trendB,
  locations = dataB,
  newdata=valB,
  model = vgmB,
  beta = mcmlEstimation2B$beta,
  nmax=100,
  debug.level=1
)
#signal
prB[k,j,]<-tmpB@data$var1.pred
seB[k,j,]<-(tmpB@data$var1.pred-valB$S)^2
```

```r
#pi
pr.pi[k,j,]<-transf.predB(tmpB@data$var1.pred,tmpB@data$var1.var)
#kriging variance
var.pi[k,j,]<-transf.varB(tmpB@data$var1.pred,tmpB@data$var1.var)
se.pi[k,j,]<-(transf.predB(tmpB@data$var1.pred, tmpB@data$var1.var)-
              transf.predB(valB$S,valB.var$S))^2
npointsB[k,j]<-npt
}
}
```

10.3.3 Predictions Poisson

```r
# poisson
for (k in 1:length(spacing)) {
    for (j in 1:nsteek) {
        s <- spsample(grdP, cellsize = spacing[k], type = "regular")
        npt <- length(s)
        o <- over(s, grdP)
        sam <- data.frame(s, o)
        names(sam)[1:2] <- c("x1", "x2")
        sam <- na.omit(sam)
        coordinates(sam) <- ~x1 + x2
        proj4string(sam) <- RD
        dataP <- sam
        tmpP <- krige(formula = trendP, locations = dataP, newdata = valP,
            model = vgmP, beta = mcmlEstimation2P$beta, nmax = 100,
            debug.level = 1)
        # signal
        prP[k, j, ] <- tmpP@data$var1.pred
        seP[k, j, ] <- (tmpP@data$var1.pred - valP$S)^2
        # mu
        pr.mu[k, j, ] <- transf.predP(tmpP@data$var1.pred, tmpP@data$var1.var)
        # kriging variance
        var.mu[k, j, ] <- transf.varP(tmpP@data$var1.pred, tmpP@data$var1.var)
        se.mu[k, j, ] <- (transf.predP(tmpP@data$var1.pred, tmpP@data$var1.var) -
            transf.predP(valP$S, valP.var$S))^2
        npointsP[k, j] <- npt
    }
}
```

10.3.4 Quality measures

```r
# calculate mse for signals B/P and for pi/mu
mseB <- apply(seB, c(1:2), mean)
mse.pi <- apply(se.pi, c(1:2), mean)
mseP <- apply(seP, c(1:2), mean)
mse.mu <- apply(se.mu, c(1:2), mean)
# 1000 val points averaged over nsteek per grid spacing
```

```
mean.pr.pi <- apply(pr.pi, c(1, 3), mean)
mean.pr.mu <- apply(pr.mu, c(1, 3), mean)
val.pi <- transf.predB(valB$S, valB.var$S)
save(seB, seP, prB, prP, pr.pi, pr.mu, se.pi, se.mu, var.pi,
    var.mu, mse.pi, mse.mu, npointsB, npointsP, val.xy, val.pi,
    file = "./output/results-mcml.RData")
```

```
# measure of quality for pi
load(file = "./output/results-mcml.RData")
bin.pr <- ifelse(apply(pr.pi, c(1, 3), mean) > 0.5, 1, 0)
valpi <- ifelse(val.pi > 0.5, 1, 0)
spacing <- 2^seq(2, 7, 1) * 100
conttb <- vector(mode = "list", length = length(spacing))
for (i in 1:length(spacing)) {
    x00 <- table(valpi, bin.pr[i, ])[[1]]
    x01 <- table(valpi, bin.pr[i, ])[[3]]
    x10 <- table(valpi, bin.pr[i, ])[[2]]
    x11 <- table(valpi, bin.pr[i, ])[[4]]
    conttb[[i]] <- list(overall = (x11 + x00)/(x00 + x01 + x10 +
        x11), users1 = x11/(x11 + x01), users0 = x00/(x00 + x10),
        prod1 = x11/(x10 + x11), prod0 = x00/(x00 + x01))
}
df <- matrix(unlist(conttb), nrow = 5, ncol = 6)
colnames(df) <- spacing[1:6]
row.names(df) <- c("overall", "user1", "users0", "producers1",
    "producers0")
stargazer(df, title = "Estimates of overall accuracy, users accuracy and producers accuracy f
```

Table 4: Estimates of overall accuracy, users accuracy and producers accuracy for predicted prevalence calculated for different grid spacings (m) with fixed model parameters. Average of 100 realisations.

	400	800	1600	3200	6400	12800
overall	0.997	0.994	0.984	0.968	0.949	0.927
user1	0.991	0.981	0.971	0.937	0.900	0.879
users0	0.999	0.997	0.987	0.976	0.961	0.938
producers1	0.995	0.991	0.953	0.910	0.853	0.758
producers0	0.997	0.995	0.992	0.984	0.975	0.972

10.4 Prediction with variable model parameters

10.4.1 Model setup

```
# MCML parameters evaluated on 100m grid
mcmlEstimation2B <- readRDS("../R1/input/bin.mcml2.rds")
mcmlEstimation2P <- readRDS("../R1/input/pois.mcml3.rds")
```

10.4.2 Prediction Bernoulli and Poisson

```
#grid spacing
#start with 400m
spacing<-2^seq(2,7,1)*100
k<-1
seB<-seP<-se.pi<-se.mu<-prB<-prP<-array(dim=c(length(spacing), 1000))
    pr.pi<-pr.mu<-var.pi<-var.mu<-array(dim=c(length(spacing), 1000))
remlB.conv<-remlP.conv<-remlB.sigmasq<-array(dim=c(length(spacing)))
remlP.sigmasq<-remlB.nugget<-remlP.nugget<- array(dim=c(length(spacing)))
remlB.phi<-remlP.phi<-npointsB<-npointsP<-array(dim=c(length(spacing)))
remlB.beta<-remlP.beta<-array(dim=c(length(spacing),(length(trendB)+1)))
for(k in 1:length(spacing)){
#binomial
s<- spsample(grdB, cellsize=spacing[k],type="regular")
nptB<-length(s)
o<-over(s, grdB)
sam<-data.frame(s,o)
names(sam)[1:2]<-c("x1","x2")
sam<-na.omit(sam)
coordinates(sam)<-~x1+x2
proj4string(sam)<-RD
dataB<-sam
#poisson
s<- spsample(grdP, cellsize=spacing[k],type="regular")
nptP<-length(s)
o<-over(s, grdP)
sam<-data.frame(s,o)
names(sam)[1:2]<-c("x1","x2")
sam<-na.omit(sam)
coordinates(sam)<-~x1+x2
proj4string(sam)<-RD
dataP<-sam
trendB<-trendP<- ~ silt + silt2 + depth
#estimate parameters with reml
remlB <- likfit(
    geodata = as.geodata(sam,data.col="S",covar.col=c("silt", "silt2", "depth")),
    trend=trend.spatial(trend=trendB),
    cov.model="spherical",
    ini.cov.pars=c(mcmlEstimation2B$cov.pars[1], mcmlEstimation2B$cov.pars[2]),
    nugget=mcmlEstimation2B$nugget.rel*mcmlEstimation2B$cov.pars[1],
    lik.method="REML"
)
remlP <- likfit(
    geodata = as.geodata(sam,data.col="S",covar.col=c("silt", "silt2", "depth")),
    trend=trend.spatial(trend=trendP),
    cov.model="spherical",
    ini.cov.pars=c(mcmlEstimation2P$cov.pars[1], mcmlEstimation2P$cov.pars[2]),
    nugget=mcmlEstimation2P$nugget.rel*mcmlEstimation2P$cov.pars[1],
    lik.method="REML"
)
```

```
#check convergence
if(remlB$info.minimisation.function$convergence==0&remlP$info.minimisation.function$convergen
if(remlB$phi==0) remlB$phi<-0.001
if(remlP$phi==0) remlP$phi<-0.001
#redefine trend
trendB<-trendP<- S ~ silt + silt2 + depth
#kriging S Bernoulli
tmpB <- krige(
  formula = trendB,
  locations = dataB,
  newdata=valB,
  model = vgm(model="Sph",psill=remlB$sigmasq,range=remlB$phi,nugget=remlB$nugget),
  beta = remlP$beta,
  nmax=100,
  debug.level=1
)
#signal
prB[k,]<-tmpB@data$var1.pred
seB[k,]<-(tmpB@data$var1.pred-valB$S)^2
#pi
pr.pi[k,]<-transf.predB(tmpB@data$var1.pred,tmpB@data$var1.var)
#kriging variance
var.pi[k,]<-transf.varB(tmpB@data$var1.pred,tmpB@data$var1.var)
se.pi[k,]<-(transf.predB(tmpB@data$var1.pred, tmpB@data$var1.var)-transf.predB(valB$S,valB.va
npointsB[k]<-nptB
#kriging S Poisson
tmpP <- krige(
  formula = trendP,
  locations = dataP,
  newdata=valP,
  model = vgm(model="Sph",psill=remlP$sigmasq,range=remlP$phi,nugget=remlP$nugget),
  beta = remlP$beta,
  nmax=100,
  debug.level=1
)
#signal
prP[k,]<-tmpP@data$var1.pred
seP[k,]<-(tmpP@data$var1.pred-valP$S)^2
#pi
pr.mu[k,]<-transf.predP(tmpP@data$var1.pred,tmpP@data$var1.var)
#kriging variance
var.mu[k,]<-transf.varP(tmpP@data$var1.pred,tmpP@data$var1.var)
se.mu[k,]<-(transf.predP(tmpP@data$var1.pred, tmpP@data$var1.var)-transf.predP(valP$S,valP.va
npointsP[k]<-nptP
#accumulate reml convergence
remlB.conv[k]<-remlB$info.minimisation.function$convergence
remlB.phi[k]<-remlB$phi
remlB.sigmasq[k]<-remlB$sigmasq
remlB.nugget[k]<-remlB$nugget
remlP.conv[k]<-remlP$info.minimisation.function$convergence
remlP.phi[k]<-remlP$phi
remlP.sigmasq[k]<-remlP$sigmasq
remlP.nugget[k]<-remlP$nugget
remlB.beta[k,]<-remlB$beta
remlP.beta[k,]<-remlP$beta
}
}
```

10.4.3 Quality measures

```
mse.pi <- apply(se.pi, 1, mean)
mse.mu <- apply(se.mu, 1, mean)
mse.pi
mse.mu
val.pi <- transf.predB(valB$S, valB.var$S)
val.mu <- transf.predP(valP$S, valP.var$S)
# add results400 save all results
save(mse.pi, mse.mu, seB, seP, prB, prP, pr.pi, pr.mu, se.pi,
    se.mu, var.pi, var.mu, npointsB, npointsP, val.xy, val.pi,
    val.mu, file = "./output/results-reml1.RData")
# save reml parameters
save(remlB.conv, remlP.conv, remlB.sigmasq, remlP.sigmasq, remlB.nugget,
    remlP.nugget, npointsB, npointsP, remlB.beta, remlP.beta,
    file = "./output/reml1.RData")
```

```
load(file = "./output/results-reml1.RData")
pr.pi.reml1 <- pr.pi
my.list <- list(pr.pi.reml1)
df <- Reduce("+", my.list)/length(my.list)
bin.pr <- ifelse(df > 0.5, 1, 0)
valpi <- ifelse(val.pi > 0.5, 1, 0)
conttb <- vector(mode = "list", length = length(spacing[-1]))
for (i in 1:length(spacing[-1])) {
    x00 <- table(valpi, bin.pr[i, ])[[1]]
    x01 <- table(valpi, bin.pr[i, ])[[3]]
    x10 <- table(valpi, bin.pr[i, ])[[2]]
    x11 <- table(valpi, bin.pr[i, ])[[4]]
    conttb[[i]] <- list(overall = (x11 + x00)/(x00 + x01 + x10 +
        x11), users1 = x11/(x11 + x01), users0 = x00/(x00 + x10),
        prod1 = x11/(x10 + x11), prod0 = x00/(x00 + x01))
}
df <- matrix(unlist(conttb), nrow = 5, ncol = 6)
colnames(df) <- spacing[1:6]
row.names(df) <- c("overall", "user1", "users0", "producers1",
    "producers0")
stargazer(df, title = "Estimates of overall accuracy, users accuracy and producers accuracy f
```

Table 5: Estimates of overall accuracy, users accuracy and producers accuracy for predicted prevalence calculated for different grid spacings (m) with variable model parameters

	400	800	1600	3200	6400	12800
overall	0.974	0.970	0.943	0.840	0.835	0.974
user1	0.965	0.959	0.897	0.573	0.566	0.965
users0	0.976	0.973	0.954	0.985	0.979	0.976
producers1	0.910	0.896	0.825	0.953	0.934	0.910
producers0	0.991	0.990	0.975	0.810	0.809	0.991

```r
load(file = "./output/results-reml1.RData")
pr.pi.reml1 <- pr.pi
pr.mu.reml1 <- pr.mu
spacing <- 2^seq(1, 7, 1) * 100
# combine pi mu and take avg
my.list <- list(pr.pi.reml1 * pr.mu.reml1)
df <- Reduce("+", my.list)/length(my.list)
row.names(df) <- spacing[-1]
# use the same splits as for 100 m for comparison
b <- c(-Inf, 0.3, 0.7, 1.5, Inf)
l <- c("[0-0.3)", "[0.3-0.7)", "[0.7-1.5)", "[1.5-Inf)")
colo <- c("#FFFF00", "#AAD800", "#55B100", "#008B00")
tmp <- cbind(val.xy, t(df))
tmp <- melt(tmp, id = c("x1", "x2"))
tmp$group <- cut(tmp$value, breaks = b, labels = l, right = F)
for (i in spacing[-1]) {
    print(gg.intertidal + geom_point(data = subset(tmp, variable ==
        i), aes(x = x1 * 0.001, y = x2 * 0.001, colour = group),
        size = 0.7) + scale_colour_manual(name = "", values = colo) +
        coord_equal(ratio = 1) + theme(legend.position = "bottom",
        panel.background = element_rect(fill = "white", color = NA),
        panel.border = element_rect(colour = "grey50", fill = NA)) +
        guides(colour = guide_legend(override.aes = list(size = 3))))
}
```

```r
load(file="output/results-reml1.RData")
mse.mu.reml1<-apply(se.mu,1,mean)
mse.pi.reml1<-apply(se.pi,1,mean)
load(file="./output/results-mcml.RData")
mse.mu.mcml<-mse.mu
mse.pi.mcml<-mse.pi
spacing<-2^seq(1,7,1)*100
row.names(mse.mu.mcml)<-spacing
df1<-melt(mse.mu.mcml)
df2<-data.frame(spacing, mse.mu.reml=c(NA, mse.mu.reml1))
ggplot(df1, aes(x=as.factor(Var1), y=value))+
geom_boxplot(outlier.size=2, notch=F, fill="skyblue")+
geom_point(data=df2,
```

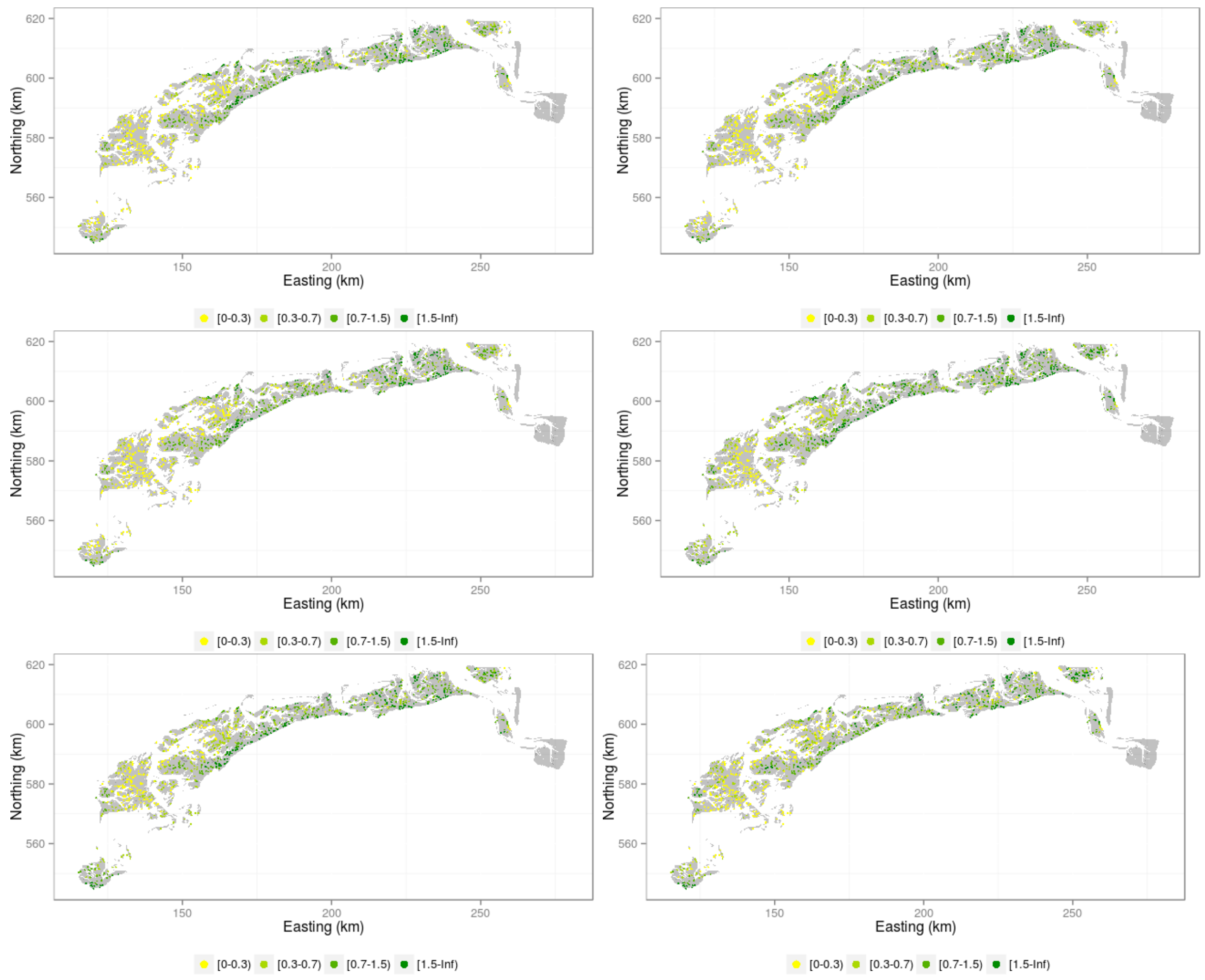


Figure 15: Unconditional intensity predicted at 1000 validation points by simple kriging with an external drift with variable model parameters from 400 (a), 800 (b), 1600 (c), 3200 (d), 6400 (e), 12800 (f) m grid.

```
           aes(x=as.factor(spacing), y=mse.mu.reml),
           size=3,
           shape=8,
           col="red")+
scale_x_discrete(name = "Spacing (m)") +
scale_y_continuous(name = "MSE (intensity)")+
theme_bw()
row.names(mse.pi.mcml)<-spacing
df1<-melt(mse.pi.mcml*mse.mu.mcml)
df2<-data.frame(spacing, mse.pimu.reml=c(NA, mse.pi.reml1*mse.mu.reml1))
ggplot(df1, aes(x=as.factor(Var1), y=value))+
geom_boxplot(outlier.size=2, notch=F, fill="skyblue")+
geom_point(data=df2, aes(x=as.factor(spacing), y=mse.pimu.reml),
           size=3,
           shape=8,
           col="red")+
scale_x_discrete(name = "Spacing (m)") +
scale_y_continuous(limits=c(0,0.1),
```

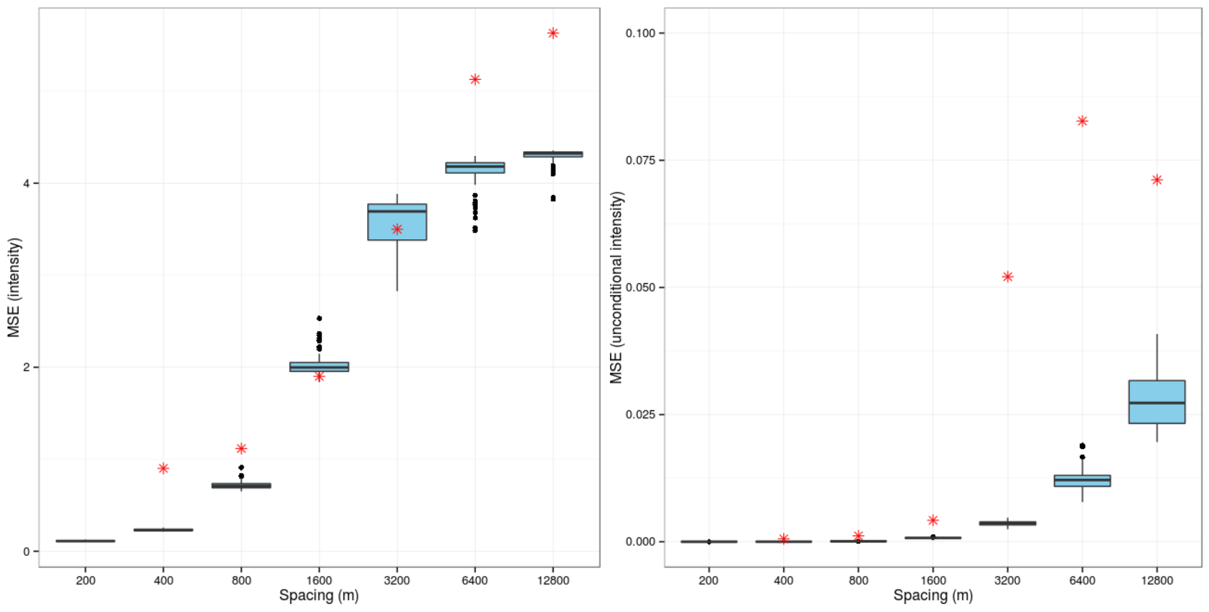


Figure 16: The MSE for predicted intensity (a) and predicted unconditional intensity (b) as a function of grid spacing for 200, 400, 800, 1600, 3200, 6400, and 12800 m. Predictions were obtained by simple kriging with an external drift with fixed model parameters (blue) and variable model parameters (red).

```
                              name = "MSE (unconditional intensity)")+
theme_bw()
```