Table of content

```r
# Load the required packages
require(ggplot2)
require(tidyverse)
require(MASS)
require(mgcv)
require(dplyr)
require(magrittr)
require(factoextra)
require(reshape2)
require(knitr)

require(mnormt)
require(readr)
require(sf)
require(tmap)
require(geoR)
require(maptools)
require(gstat)
```

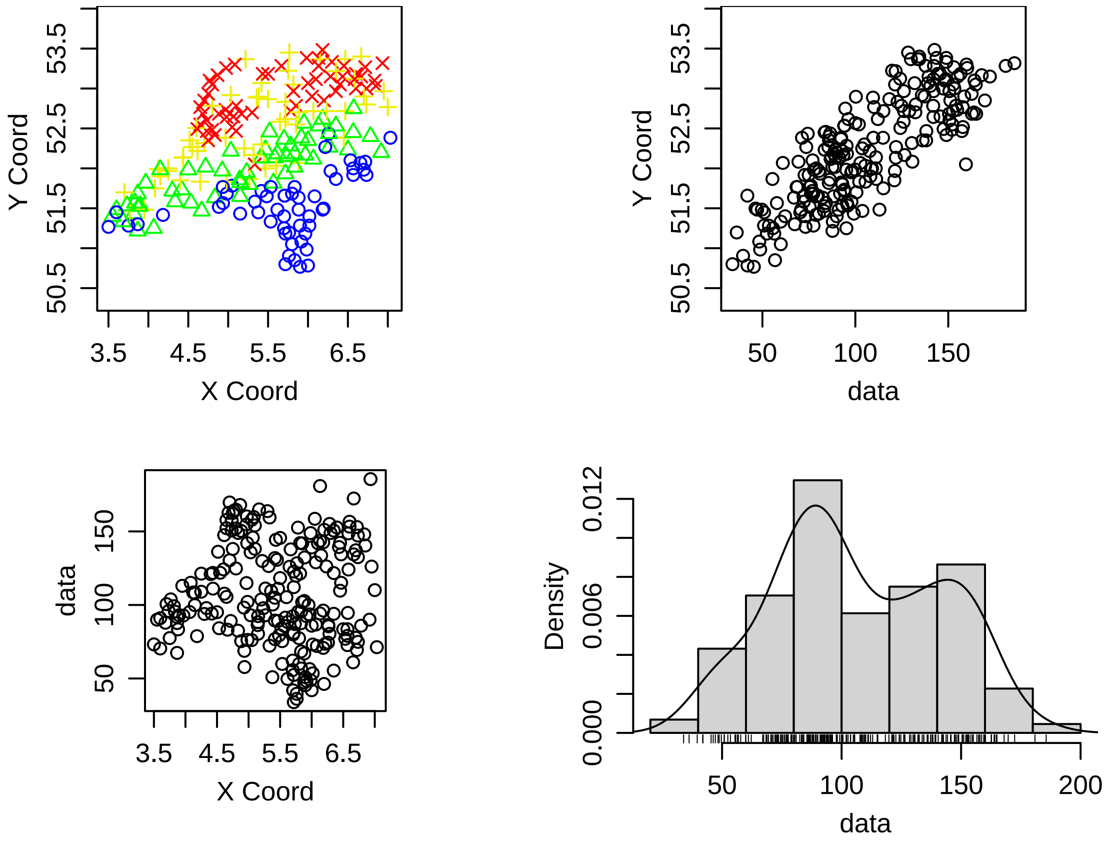
Question 1

```r
ntl <- read_csv("netherlands.csv")

# Convert to geodata object
geo_ntl <- as.geodata(ntl, coords.col = 2:3, data.col = 4)
```

1a

Plot the precipitation of Netherlands in September 2019 by geodata

```r
plot(geo_ntl)
```



Produce numerical summaries

```r
a <- summary(ntl$precip)
at <- t(a)
kable(at)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
33.9	80.85	100.15	106.5705	137.35	185.6

The top right plots the precipitation against latitude. It can be seen that higher latitude recorded higher precipitation. This reflects that northern Netherlands recorded higher precipitation than the southern.

The bottom left plots the precipitation against longitude. Rainfall volume seems to widely vary at right longitude, while at left-most longitude they center around 100. This can be due to the fact that there are fewer observations recorded there.

The top left plots the precipitation all over Netherlands, with blue, green, yellow and red respectively represent the lowest, second, third and highest quartile in the precipitation range. As mentioned, the southern region is covered in blue, while moving up to the north, it gradually changes to green, yellow and then red in the north of Netherlands. There is clearly spatial correlation. Regions of the same latitude seem to have similar or nearly similar precipitation, while precipitation of regions of the same longitude can vary (shown in the mixture of blue, green and yellows points in the same longitude)

From the numerical summary and density plot, precipitation ranges from nearly 34 to over 185, with mean = 100 and median = 106.

1b

Select 3 observed locations

```
# Random choose 3 locations
set.seed(1234)
obs_rows <- sample(1:nrow(ntl), size = 3)

# Separate training set and the 3 chosen locations for
# testing
tst <- ntl[obs_rows, ]
trn <- ntl[-obs_rows, ]

# Label test locations as A, B, C
tst$label <- c("A", "B", "C")
tst
```

```
## # A tibble: 3 x 5
##   station_name  longitude latitude precip label
##   <chr>             <dbl>    <dbl>  <dbl> <chr>
## 1 MARKEN             5.1      52.5  138.   A
## 2 BRIELLE            4.15     51.9  108.   B
## 3 OOST MAARLAND      5.72     50.8   33.9 C
```

1c

Calculate and plot the sample variogram of the data, first with assumption that mean is constant. Then we will test variogram with different adjustments of mean (transformation or linear/ quadratic trend)

```
# Convert training set to geodata
geo_trn <- as.geodata(trn, coords.col = 2:3, data.col = 4)

# Calculate the sample variogram with assumption that mean
# is constant
sample_vario_full <- variog(geo_trn, option = "bin")
```

```
## variog: computing omnidirectional variogram
```

```
# Test different adjustment of mean assumption
sample_vario_lmda2 <- variog(geo_trn, option = "bin", lambda = 2)
```

```
## variog: computing omnidirectional variogram
```

```
sample_vario_lmda0 <- variog(geo_trn, option = "bin", lambda = 0)
```

```
## variog: computing omnidirectional variogram
```

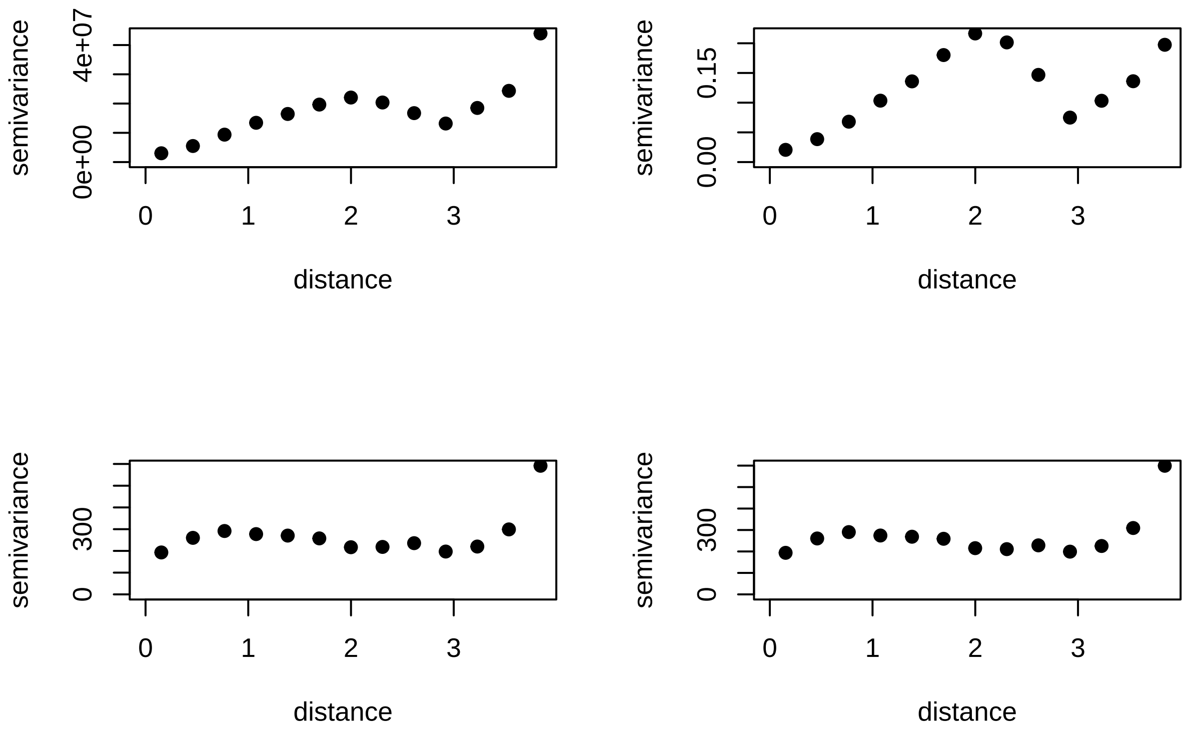
```
sample_vario_trnd1 <- variog(geo_trn, option = "bin", trend = "1st")
```

```
## variog: computing omnidirectional variogram
```

```
sample_vario_trnd2 <- variog(geo_trn, option = "bin", trend = "2nd")
```

```
## variog: computing omnidirectional variogram
```

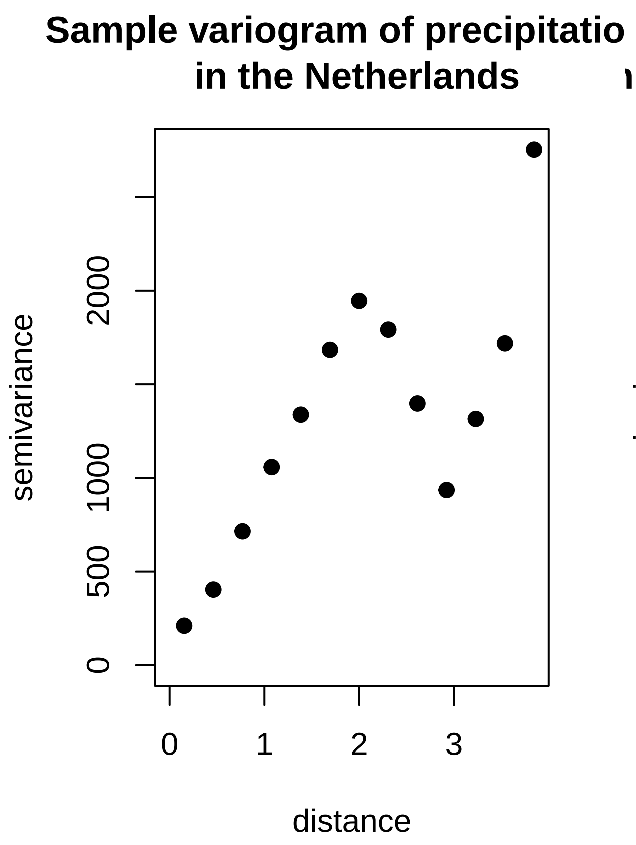
```
par(mfrow = c(2, 2))
plot(sample_vario_lmda2, pch = 19)
plot(sample_vario_lmda0, pch = 19)
plot(sample_vario_trnd1, pch = 19)
plot(sample_vario_trnd2, pch = 19)
```

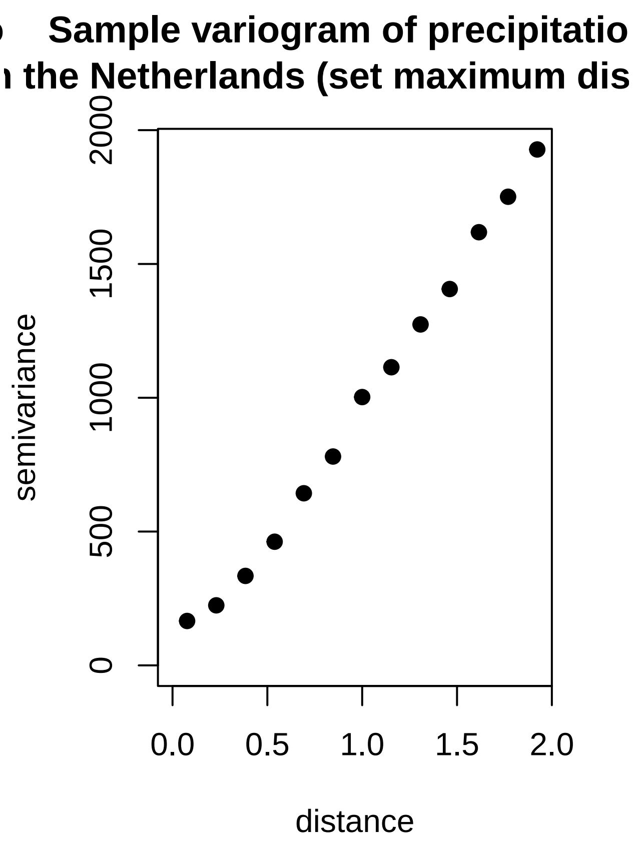


```
# variogram with assumption that mean is constant, cut max
# distance = 2
sample_vario <- variog(geo_trn, option = "bin", max.dist = 2)
```

```
## variog: computing omnidirectional variogram
```

```
# Plot the sample variogram
par(mar = c(4, 4, 4, 2), mfrow = c(1, 2))
plot(sample_vario_full, pch = 19, main = "Sample variogram of precipitation \n in the Netherlands")
plot(sample_vario, pch = 19, main = "Sample variogram of precipitation \n in the Netherlands (set maxim
```





The bottom variograms when we assume the mean has a first or second order polynomial on the coordinates are not stable and continuous. The variance fluctuates over the range of distances and does not reach a sensible sill. The top variograms when we transform the mean share the same trend with the original variogram (with assumption of constant mean by default).

We use the original variogram. We need to set a maximum distance = 2 before fitting a model because when the distance is larger than 2, the variance starts to drop. This decrease does not describe the variance of points which are far apart, but due to the fact that there are not as many data points of this distance as of closer distance. Thus, from distance = 2, variance starts to drop before going up at 3.

We need to include a nugget. As can be seen from the right plot, the variance at distance = 0 is larger than 0 (could be around 100), therefore a nugget is needed to reflect this variance.

1d

Fit the variogram

```r
# fit matern 3/2
vari.mat1.5 <- variofit(sample_vario, kappa = 1.5)
```

```
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(sample_vario, kappa = 1.5): initial values not provided -
## running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq    kappa
## initial.value "1927.8" "0.62" "192.78" "1.5"
## status        "est"    "est"  "est"    "fix"
## loss value: 308377280.663826
```

```
# fit matern 5/2
vari.mat2.5 <- variofit(sample_vario, kappa = 2.5)
```

```
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, kappa = 2.5): initial values not provided -
## running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq    kappa
## initial.value "1927.8" "0.62" "481.95" "2.5"
## status        "est"    "est"  "est"    "fix"
## loss value: 433478946.58925
```

```
# fit matern estimate kappa
vari.matest <- variofit(sample_vario, kappa = 1.5, fix.kappa = FALSE,
    cov.model = "matern")
```

```
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, kappa = 1.5, fix.kappa = FALSE, cov.model =
## "matern"): initial values not provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq    kappa
## initial.value "1927.8" "0.62" "192.78" "2"
## status        "est"    "est"  "est"    "est"
## loss value: 273775698.510249
```

```
# try different model
model_list <- c("exponential", "gaussian", "spherical", "circular",
    "cubic", "wave", "power", "powered.exponential", "cauchy",
    "gneiting")
min_sum_sq <- c()

for (i in 1:length(model_list)) {
    vari_i <- variofit(sample_vario, cov.model = model_list[i])
    # get the minimized weighted sum of squares
    sos_i <- vari_i$value
    pars_i <- vari_i$cov.pars
    nugg_i <- vari_i$nugget
    # print the result
    min_sum_sq <- c(min_sum_sq, sos_i)
}
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
```

```
##               sigmasq  phi    tausq kappa
## initial.value "1927.8" "1.23" "0"   "0.5"
## status        "est"    "est"  "est" "fix"
## loss value: 666872354.120309
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq kappa
## initial.value "1927.8" "1.23" "0"   "0.5"
## status        "est"    "est"  "est" "fix"
## loss value: 167720624.522887
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq   phi    tausq kappa
## initial.value "1445.85" "1.54" "0"   "0.5"
## status        "est"     "est"  "est" "fix"
## loss value: 1122405317.69195
## variofit: covariance model used is circular
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq   phi    tausq kappa
## initial.value "1445.85" "1.54" "0"   "0.5"
## status        "est"     "est"  "est" "fix"
## loss value: 725038008.941232
## variofit: covariance model used is cubic
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq   phi    tausq kappa
## initial.value "1445.85" "1.54" "0"   "0.5"
## status        "est"     "est"  "est" "fix"
## loss value: 1554441090.49136
## variofit: covariance model used is wave
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search
## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq    kappa
## initial.value "1927.8" "0.62" "192.78" "0.5"
## status        "est"    "est"  "est"    "fix"
## loss value: 207325770.680383
## variofit: covariance model used is power
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq    kappa
## initial.value "963.9" "1.54" "192.78" "0.5"
## status        "est"   "est"  "est"    "fix"
## loss value: 6086381185.28682
## variofit: covariance model used is powered.exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq kappa
## initial.value "1927.8" "1.54" "0"   "0.5"
## status        "est"    "est"  "est" "fix"
## loss value: 2045040682.53139
## variofit: covariance model used is cauchy
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq    kappa
## initial.value "1927.8" "0.62" "192.78" "0.5"
## status        "est"    "est"  "est"    "fix"
## loss value: 649859898.493717
## variofit: covariance model used is gneiting
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(sample_vario, cov.model = model_list[i]): initial values not
## provided - running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq  phi    tausq kappa
## initial.value "1927.8" "1.23" "0"   "0.5"
## status        "est"    "est"  "est" "fix"
## loss value: 170150938.08685
```

```
model_list <- c(model_list, "mattern 3/2", "mattern 5/2", "mattern est")
min_sum_sq <- c(min_sum_sq, vari.mat1.5$value, vari.mat2.5$value,
    vari.matest$value)
```

```
# List of model and minimized weighted sum of square of
# residuals
variog_table <- data.frame(model = model_list, min_sum_sq = min_sum_sq)
variog_table <- variog_table[order(variog_table$min_sum_sq),
    ]
kable(variog_table)
```

	model	min_sum_sq
7	power	11181319
11	mattern 3/2	11853782
9	cauchy	14806997
12	mattern 5/2	15767716
5	cubic	18264745
10	gneiting	22376235
2	gaussian	23263283
3	spherical	24095602
4	circular	24096674
1	exponential	24174761
6	wave	28534445
13	mattern est	30819663
8	powered.exponential	1061231062

```
# refit the power variogram
vari_pow <- variofit(sample_vario, cov.model = "power")
```

```
## variofit: covariance model used is power
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = "power"): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##                 sigmasq phi    tausq    kappa
## initial.value "963.9" "1.54" "192.78" "0.5"
## status        "est"   "est"  "est"    "fix"
## loss value: 6086381185.28682
```

```
# refit the cauchy variogram
vari_cau <- variofit(sample_vario, cov.model = "cauchy")
```

```
## variofit: covariance model used is cauchy
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(sample_vario, cov.model = "cauchy"): initial values not
## provided - running the default search

## variofit: searching for best initial value ... selected values:
##                 sigmasq  phi    tausq    kappa
## initial.value "1927.8" "0.62" "192.78" "0.5"
## status        "est"    "est"  "est"    "fix"
## loss value: 649859898.493717
```

```r
vari_pow$cov.pars
```

```
## [1] 918.106342   1.109156
```

```r
vari_pow$nugget
```

```
## [1] 34.49368
```

```r
vari.mat1.5$cov.pars
```

```
## [1] 3153.554158    1.050256
```

```r
vari.mat1.5$nugget
```

```
## [1] 174.9499
```

```r
vari_cau$cov.pars
```

```
## [1] 4208.379467    1.433507
```

```r
vari_cau$nugget
```

```
## [1] 196.3193
```

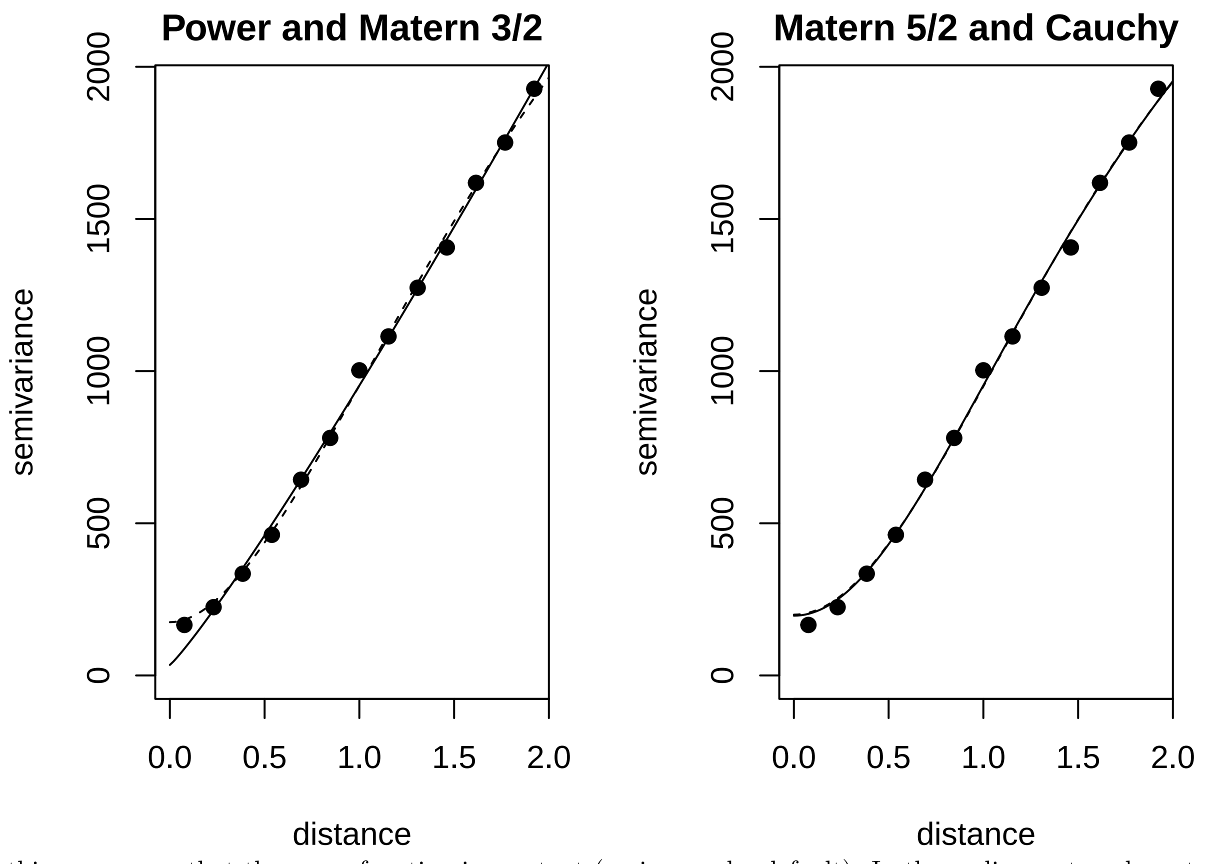
```r
vari.mat2.5$cov.pars
```

```
## [1] 2626.8968422    0.6499085
```

```r
vari.mat2.5$nugget
```

```
## [1] 199.7438
```

```r
par(mar = c(4, 4, 2, 2), mfrow = c(1, 2))
plot(sample_vario, pch = 19, main = "Power and Matern 3/2")
lines(vari_pow)
lines(vari.mat1.5, lty = 2)
plot(sample_vario, pch = 19, main = "Matern 5/2 and Cauchy")
lines(vari_cau)
lines(vari.mat2.5, lty = 2)
```



By this we assume that the mean function is constant (variogram by default). In the earlier part we have tested with other assumptions of mean, but ending up with constant mean assumption over the region.

We iterate the variogram with different assumptions about covariance functions, with fitted nugget and let it estimate the paramteters using weighted least squares. With Matern model, we test some assumptions of smoothness parameter kappa, including 3/2 and 5/2. With other models, we do not force the nugget to be zero, and fit the variogram by assuming different covariance models. Comparing different fitted models by the minimised weighted sum of squares, we can see that the Covariance Model = Power, Matern 3/2, Matern 5/2 and Cauchy result in the least residuals. We plot these fitted models to our sample variogram.

The matern 3/2 captures better the variance at close distance, while the power underestimate the nugget but looks better at far distance (when distance >1). The Matern 5/2 and Cauchy are quite identical, and both slightly overestimate the nugget. Except for the Power which estimates the nugget at around 34, The matern 3/2 estimates the nugget at over 174 and the other two models estimate the nugget at more than 190. The correlation length of Power and Matern 3/2 models are quite similar, 1.1 (Power) and Matern 3/2 (1.05). Meanwhile, Cauchy returns high correlation length (1.4) and Matern 5/2 returns much lower correlation length (0.6).

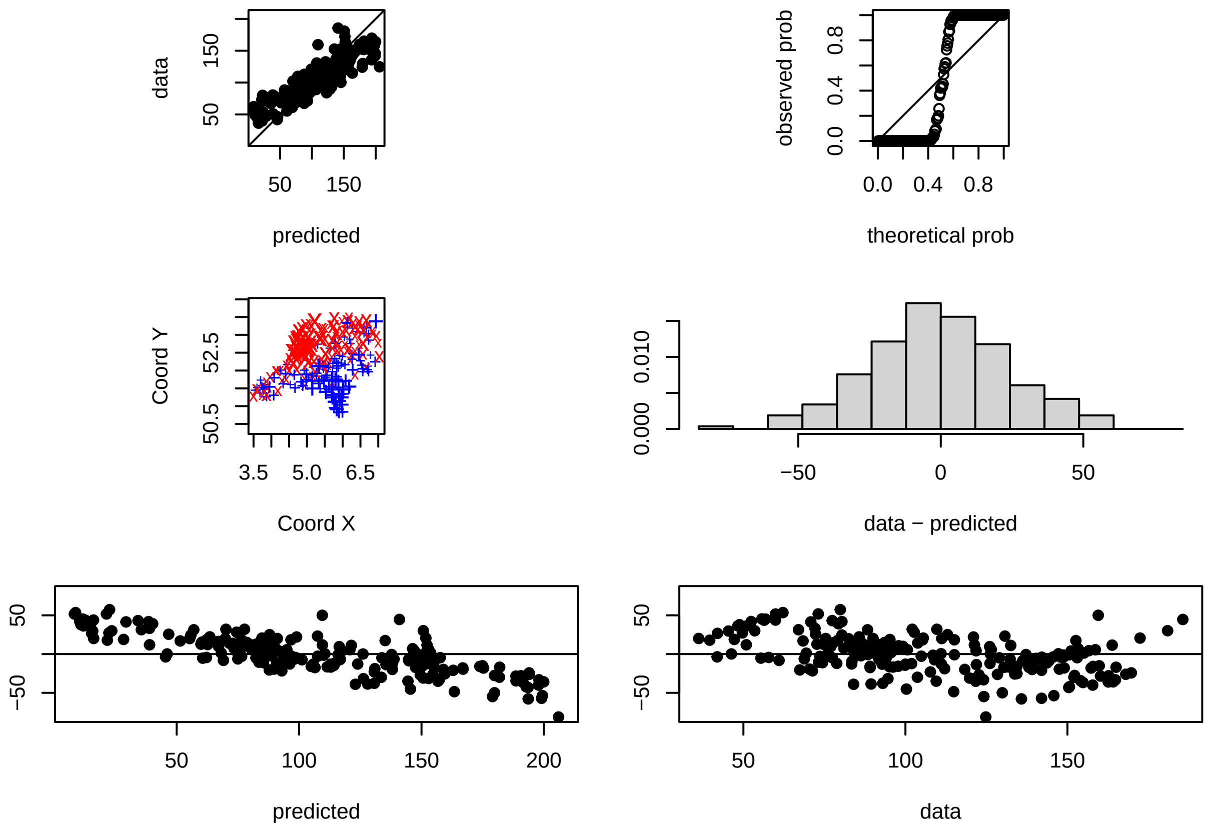
Out of the 4 models, Power does not have a sensible partial sill (918, much lower than 2000). The other three models all have the partial sill larger than 2500, which resonates in the variogram plot.

Validate variogram model: Power, Matern 3/2, Matern 5/2 and Cauchy

```
# Validate Variogram model Power
xv.ml1 <- xvalid(geo_trn, model = vari_pow)
```

```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```

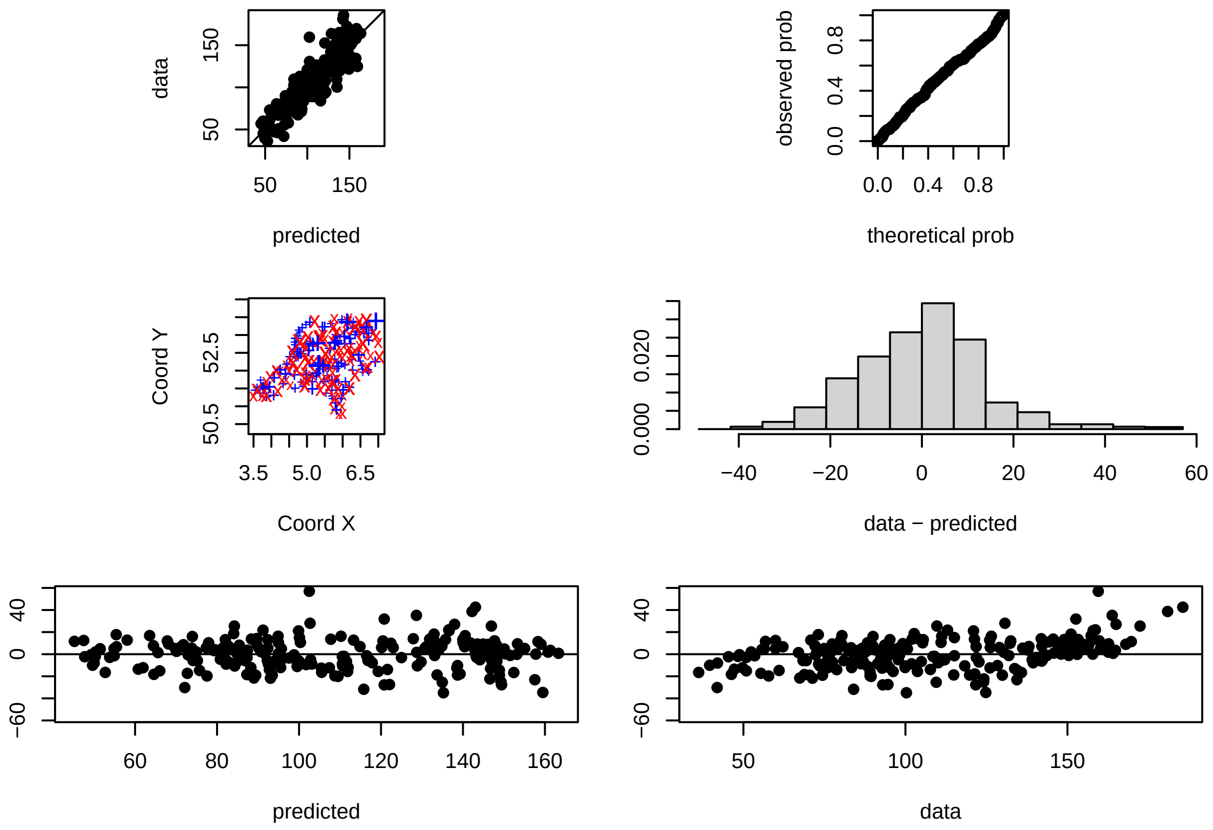
```r
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.ml1, error = TRUE, std.error = FALSE, pch = 19, main = "Validate Power")
```



```r
# Validate Variogram model Matern 3/2
xv.ml2 <- xvalid(geo_trn, model = vari.mat1.5)
```

```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```

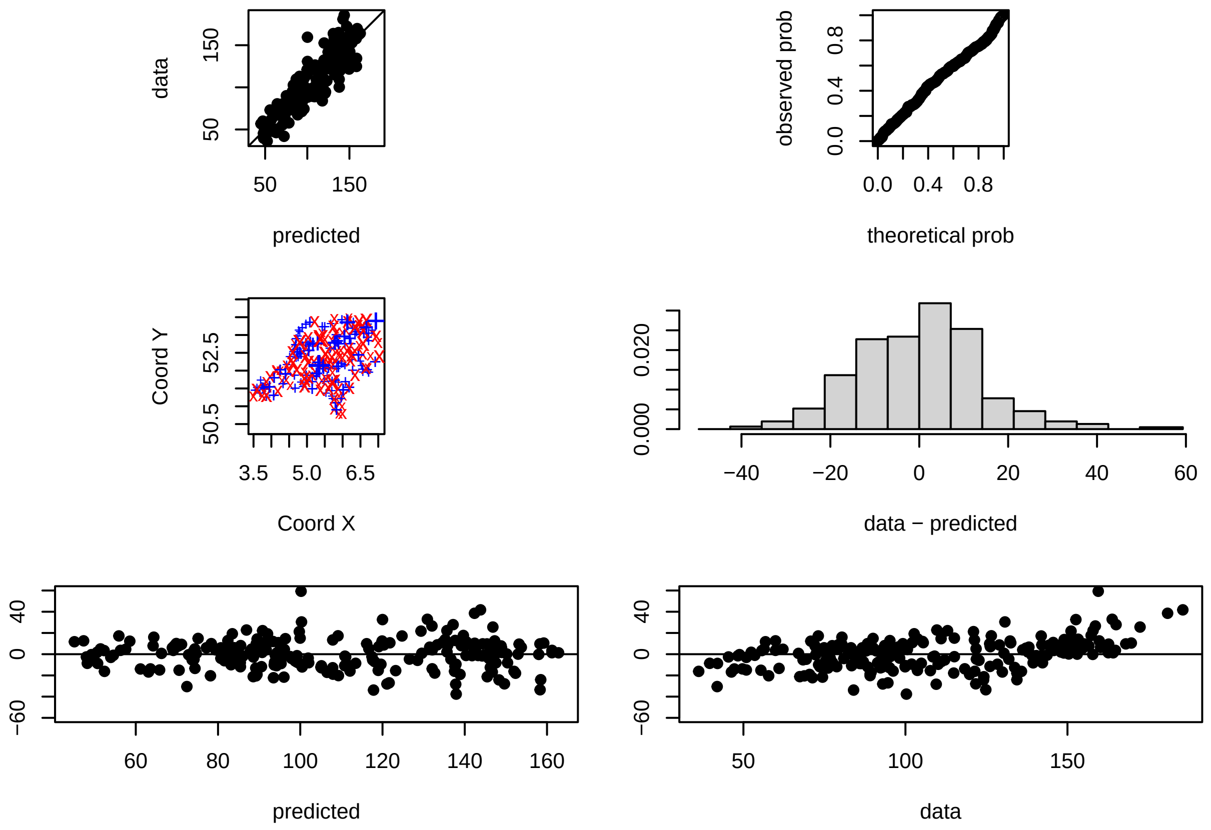
```r
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.ml2, error = TRUE, std.error = FALSE, pch = 19, main = "Validate Matern 3/2")
```



```r
# Validate Variogram model Matern 5/2
xv.ml3 <- xvalid(geo_trn, model = vari.mat2.5)
```

```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```

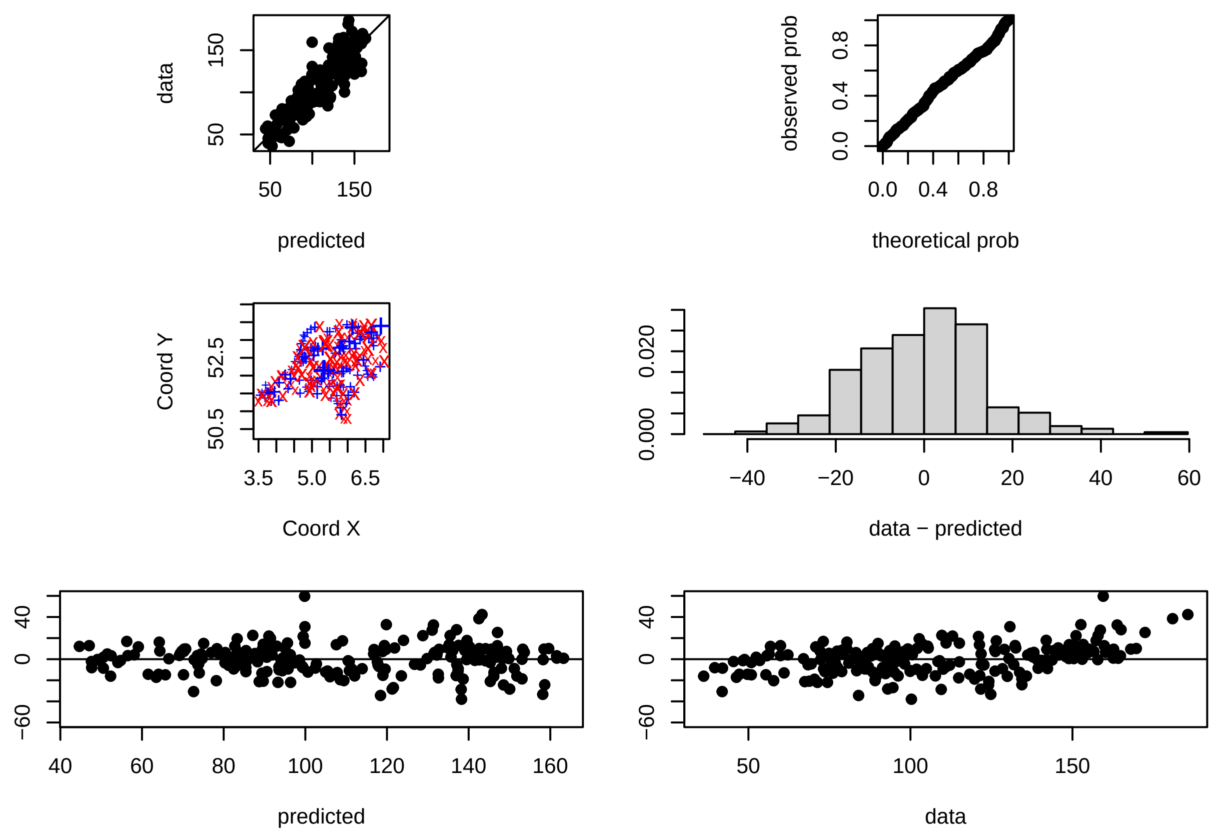
```r
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.ml3, error = TRUE, std.error = FALSE, pch = 19, main = "Validate Matern 5/2")
```



```r
# Validate Variogram model Cauchy
xv.ml4 <- xvalid(geo_trn, model = vari_cau)
```

```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```

```r
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.ml4, error = TRUE, std.error = FALSE, pch = 19, main = "Validate Cauchy")
```



```
# Summary statistics for the errors and standard errors
summary(xv.ml2)
```

```
##                   Min.   1st Qu.    Median          Mean   3rd Qu.      Max.
## errors      -34.928225 -8.6999126 0.5227679 -0.030534243 8.4489395 56.996404
## std.errors   -2.486348 -0.6244408 0.0376028 -0.001103788 0.5955349  4.096564
##                     sd
## errors      13.9291599
## std.errors   0.9765343
```

```
summary(xv.ml3)
```

```
##                   Min.   1st Qu.     Median           Mean   3rd Qu.      Max.
## errors      -37.616017 -9.3526088 0.83989772 -0.0198696993 8.6582696 59.303850
## std.errors   -2.545357 -0.6005366 0.05752065 -0.0006847676 0.5760121  4.071131
##                     sd
## errors      14.1788598
## std.errors   0.9555508
```

```
summary(xv.ml4)
```

```
##                   Min.   1st Qu.     Median           Mean   3rd Qu.      Max.
## errors      -37.960171 -9.2509677 0.57617508 -0.0143638472 8.8631443 59.667751
## std.errors   -2.601369 -0.6239712 0.03992289 -0.0005097304 0.5964391  4.145316
##                     sd
## errors      14.2519891
## std.errors   0.9721234
```

From the validation plot, Power is a poor model. Its residuals do not follow normal distribution, and residuals are allocated by regions: higher residuals lie in higher latitude, while lower residuals lie in lower latitude. We can clearly see the red and blue regions in the error map.

The other three models are quite a good fit. The Leave-on-out residuals are quite Normal, and there is no strong patterns or systematic biases in the residuals. In all the three models, there is a relatively strong relationship between the fitted and true values (top left plot), although we do slightly underestimate the data values that are over 150. The residuals are reasonably Normal - closely following the QQ line (top right). There's no clear pattern in the spatial residuals, with blue and red locations (corresponding to the sign of the residual) mostly randomly scattered (middle left). Histogram of errors show a reasonably normal distribution (middle right). In the bottom two plots, we can see that the errors of data (bottom right) above 150 tend to be positive - which confirms that the models underestimate data above 150 while errors of data under 50 tend to be negative - the models overestimate lower values. Errors by prediction scatter randomly.

Statistical summary of the errors and standard errors of the three models shows that, out of the three models, errors of model Matern 3/2 are closer together (ranging from -35 to 56 with sd = 13.85) while those of the other two are more distant.

For these reasons, Matern 3/2 is the best fit model out of these models.

1e

Fit the maximum likelihood model

```r
# Loop through models and trends, compare by loglikelihood
# and AIC
ml_model <- c("exponential", "spherical", "circular", "cubic",
    "matern")
ml_trend <- c("cte", "1st", "2nd")

model <- c()
trend <- c()
llh_list <- c()
AIC_list <- c()

for (i in 1:length(ml_model)) {
    for (j in 1:length(ml_trend)) {
        model_ml <- likfit(geo_trn, ini.cov.pars = c(2000, 1),
            cov.model = ml_model[i], trend = ml_trend[j], fix.kappa = FALSE,
            fix.lambda = FALSE)
        loglike <- model_ml$loglik
        AIC <- model_ml$AIC
        model <- c(model, ml_model[i])
        trend <- c(trend, ml_trend[j])
        llh_list <- c(llh_list, loglike)
        AIC_list <- c(AIC_list, AIC)
    }
}
```

```
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
# List of model results
mlh_table <- data.frame(model = model, trend = trend, loglikehood = llh_list,
    AIC = AIC_list)
mlh_table <- mlh_table[order(mlh_table$AIC), ]
kable(mlh_table)
```

	model	trend	loglikehood	AIC
11	cubic	1st	-877.9548	1771.910
5	spherical	1st	-878.7807	1773.561
2	exponential	1st	-880.3243	1776.649
14	matern	1st	-880.3243	1776.649
12	cubic	2nd	-877.3556	1776.711
6	spherical	2nd	-878.1445	1778.289
8	circular	1st	-881.3759	1778.752
3	exponential	2nd	-879.8930	1781.786
15	matern	2nd	-879.8930	1781.786
9	circular	2nd	-881.0283	1784.057
4	spherical	cte	-891.0904	1794.181
1	exponential	cte	-891.9485	1795.897
13	matern	cte	-891.9485	1795.897
7	circular	cte	-892.9145	1797.829

	model	trend	loglikehood	AIC
10	cubic	cte	-893.0123	1798.025

From the table results, it can be seen that models with the lowest scores of AIC and comparatively high maximised log-likelihood all have the mean be the first order polynomial on the coordinates. We will look into these four models: cubic, spherical, exponential and matern.

```
ml_cubic <- likfit(geo_trn, ini.cov.pars = c(2000, 1), trend = "1st",
    cov.model = "cubic", fix.lambda = FALSE)
```

```
## kappa not used for the cubic correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
ml_mat15 <- likfit(geo_trn, ini.cov.pars = c(2000, 1), trend = "1st",
    cov.model = "matern", kappa = 3/2, fix.lambda = FALSE)
```

```
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
ml_mat25 <- likfit(geo_trn, ini.cov.pars = c(2000, 1), trend = "1st",
    cov.model = "matern", kappa = 5/2, fix.lambda = FALSE)
```

```
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
ml_spher <- likfit(geo_trn, ini.cov.pars = c(2000, 1), trend = "1st",
    cov.model = "spherical", fix.lambda = FALSE)
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
ml_expnt <- likfit(geo_trn, ini.cov.pars = c(2000, 1), trend = "1st",
    cov.model = "exp", fix.lambda = FALSE)
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```
ml_cubic
```

```
## likfit: estimated model parameters:
##      beta0      beta1      beta2      tausq    sigmasq        phi     lambda
## "-226.894" "  -1.439" "   4.850" "   1.393" "   1.013" "   0.639" "   0.506"
## Practical Range with cor=0.05 for asymptotic range: 0.639027
##
## likfit: maximised log-likelihood = -878
```

```
ml_mat15
```

```
## likfit: estimated model parameters:
##       beta0       beta1       beta2       tausq     sigmasq         phi
## "-216.4910" "  -1.3463" "   4.6326" "   1.2775" "   1.0162" "   0.1392"
##      lambda
## "   0.4984"
## Practical Range with cor=0.05 for asymptotic range: 0.660435
##
## likfit: maximised log-likelihood = -878.8
```

```
ml_mat25
```

```
## likfit: estimated model parameters:
##       beta0       beta1       beta2       tausq     sigmasq         phi
## "-216.8702" "  -1.3563" "   4.6406" "   1.2993" "   0.9797" "   0.1019"
##      lambda
## "   0.4982"
## Practical Range with cor=0.05 for asymptotic range: 0.6031952
##
## likfit: maximised log-likelihood = -878.5
```

```
ml_spher
```

```
## likfit: estimated model parameters:
```

```
##       beta0       beta1       beta2       tausq     sigmasq         phi
## "-233.5359" "  -1.4849" "   4.9883" "   1.3076" "   1.1792" "   0.5180"
##      lambda
## "   0.5114"
## Practical Range with cor=0.05 for asymptotic range: 0.5179631
##
## likfit: maximised log-likelihood = -878.8
```

```
ml_expnt
```

```
## likfit: estimated model parameters:
##       beta0       beta1       beta2       tausq     sigmasq         phi
## "-215.1091" "  -1.3119" "   4.6026" "   1.1486" "   1.1561" "   0.2834"
##      lambda
## "   0.4982"
## Practical Range with cor=0.05 for asymptotic range: 0.8491253
##
## likfit: maximised log-likelihood = -880.3
```

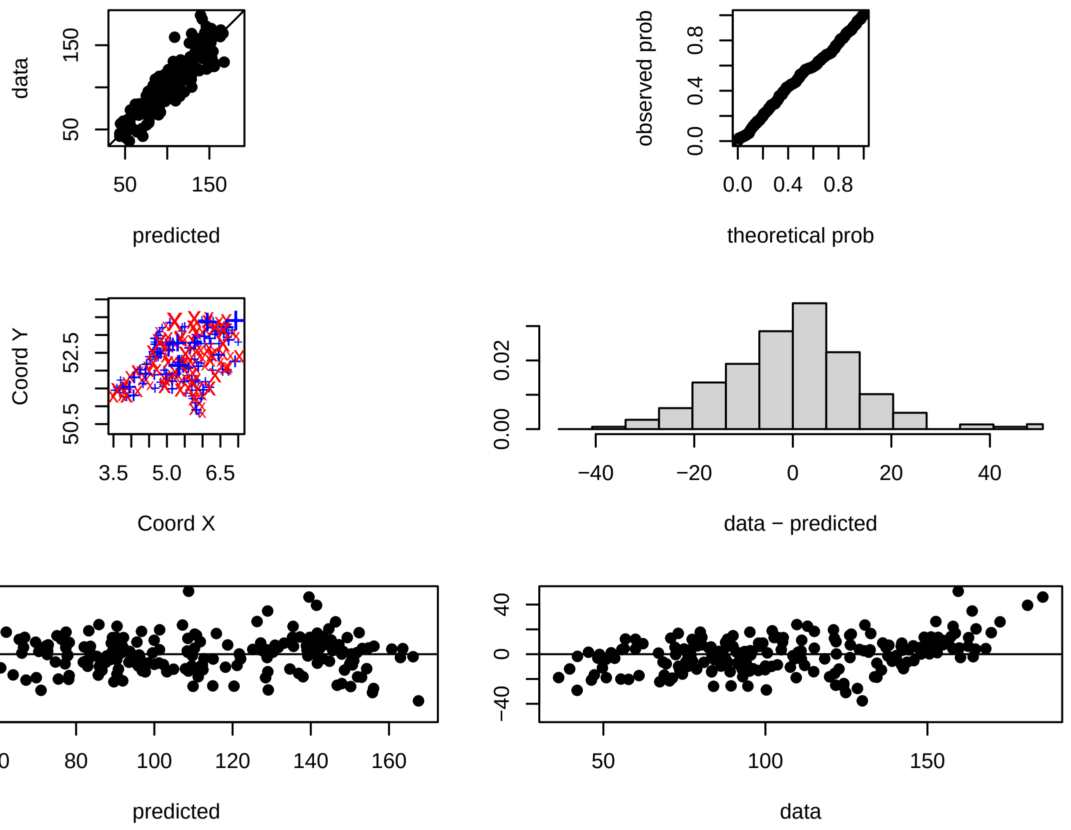
All these models estimate the estimate of lambda = 0.5, meaning that the transformation is the square root of the mean, while the nugget is estimated more than 1. Estimates of correlation length vary from 0.1 to 0.5.

Betas indicate the coefficients of the mean function, with beta1 and beta2 respectively coefficients of the spatial coordinates. All models result in high beta2 (~ from 4 to 5), confirming the positively linear relationship between data values and latitude mentioned in part a. Beta1 are estimated to be around -1, showing a slightly negative relationship with longitude.

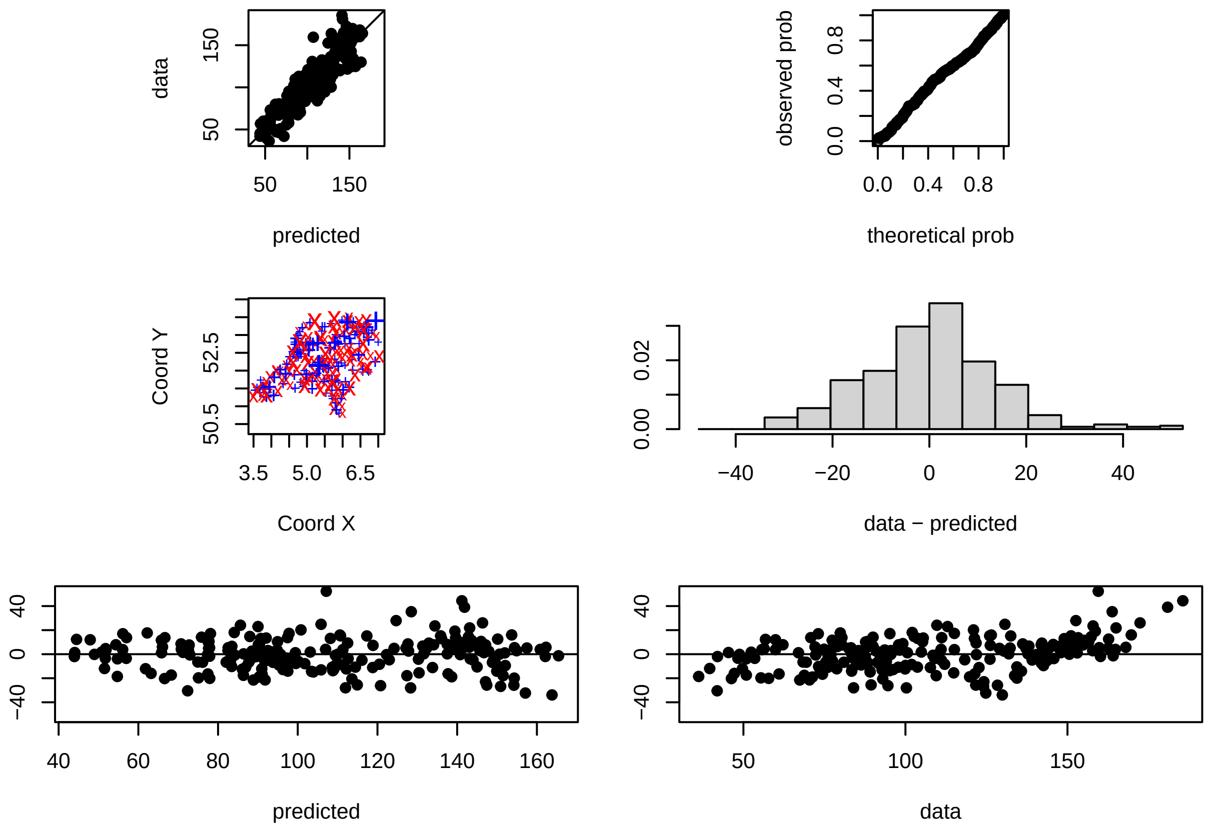
We validate these 5 models

```
# Validate models
ml_list <- list(ml_cubic, ml_mat15, ml_mat25, ml_spher, ml_expnt)
ml_list_name <- c("ml_cubic", "ml_mat15", "ml_mat25", "ml_spher",
    "ml_expnt")
err_vec <- vector("list", length(ml_list))

# Validate Maximum likelihood models
for (i in 1:length(ml_list)) {
    xv.mlh <- xvalid(geo_trn, model = ml_list[[i]])
    par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
    plot(xv.mlh, error = TRUE, std.error = FALSE, pch = 19)
    err_vec[[i]] <- xv.mlh$error
}
```

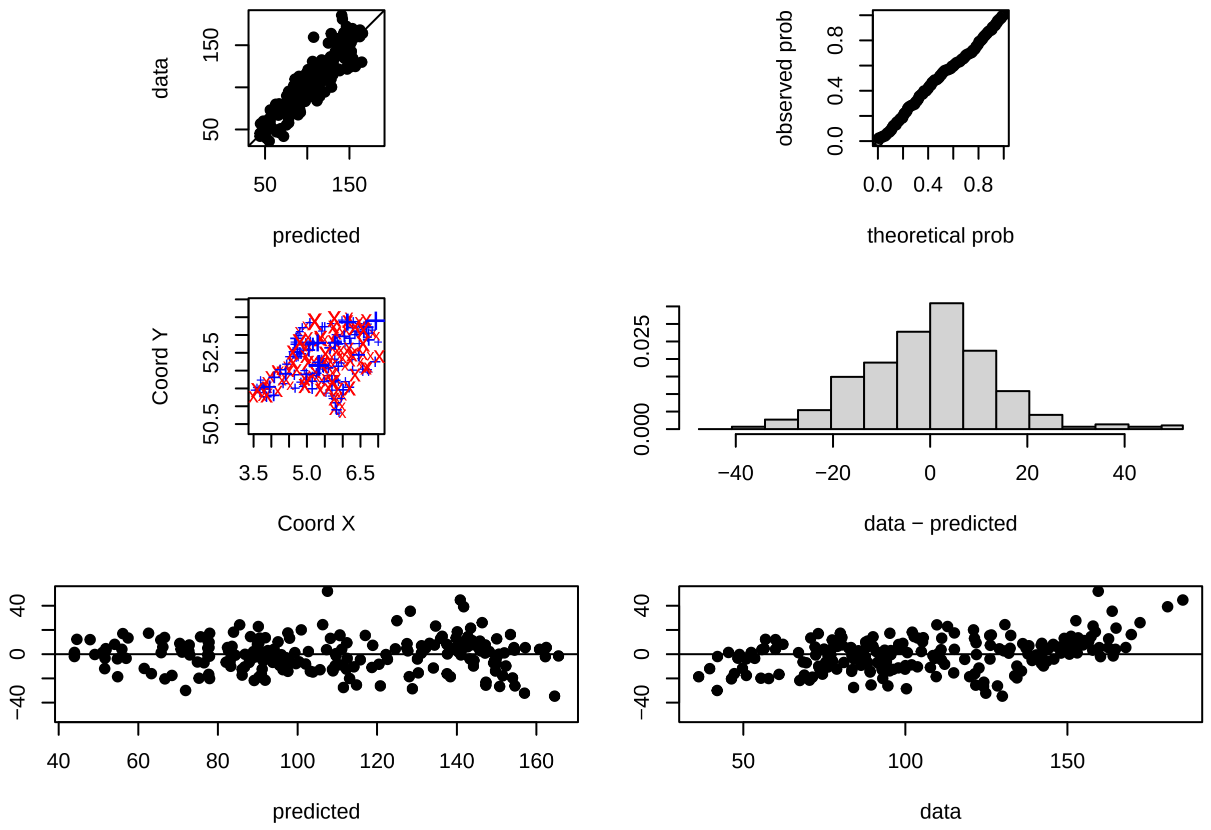
```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```



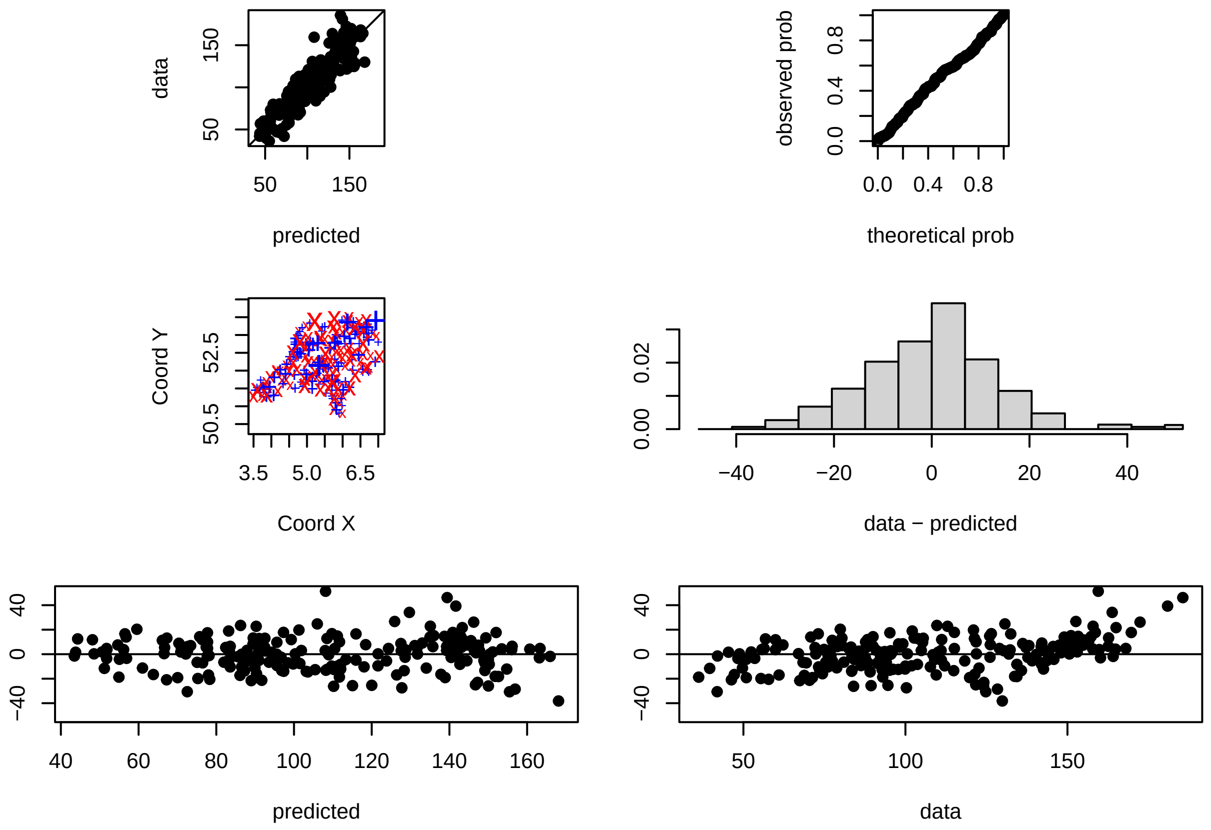
```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```



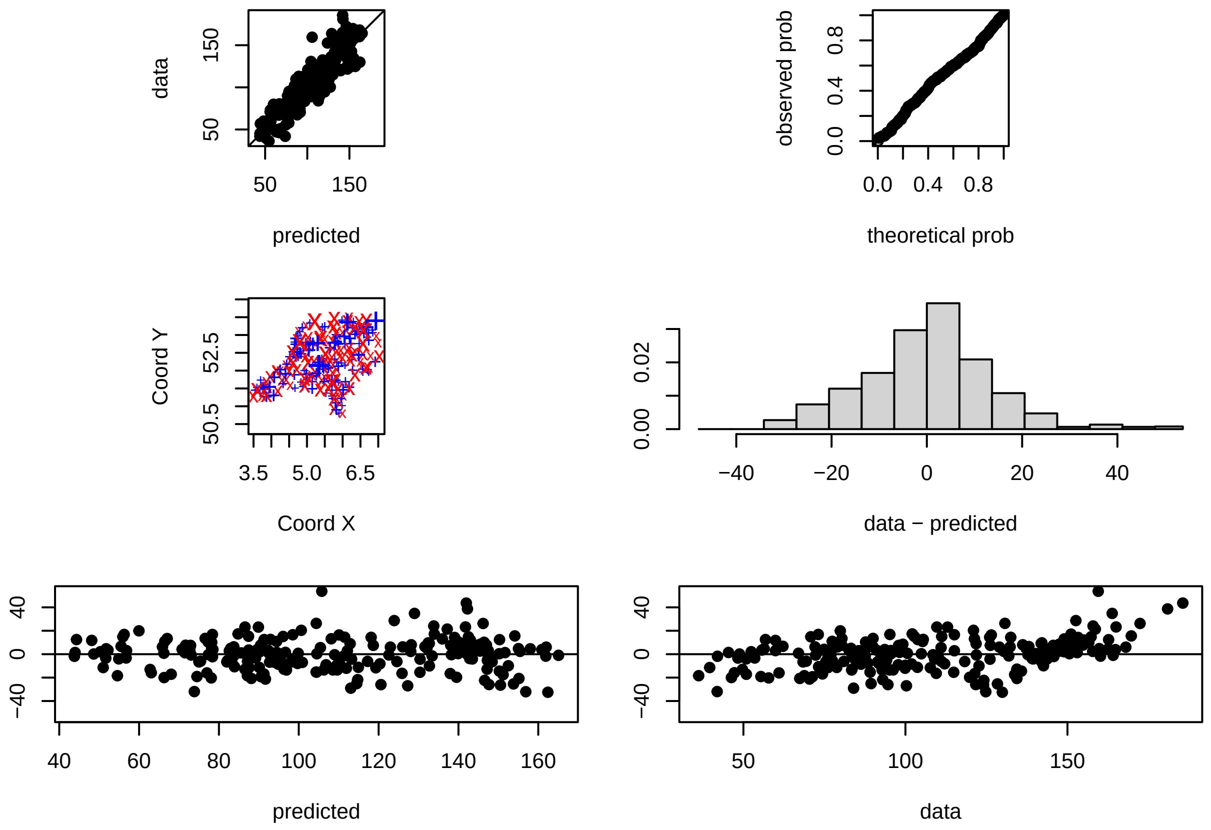
```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```



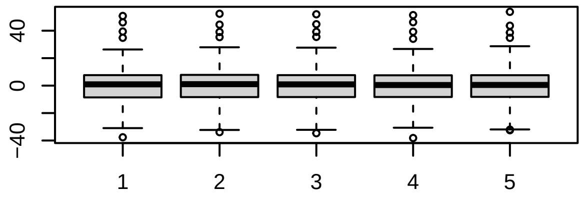
```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```



```
## xvalid: number of data locations       = 217
## xvalid: number of validation locations = 217
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1
## xvalid: end of cross-validation
```



```
# List of model validate results
par(mar = c(4, 2, 2, 2))
boxplot(err_vec)
```



From the plots, all look a good fit to the data. All 6 plots of 5 models are comparatively identical. The predicted values scatter on both sides of the “data line” (top left) and the probability follow strictly the QQ line, showing a good Normal distribution (top right). Middle left map show a mixture of blue and red points over the region, and middle right shows histogram of error which looks normal. The bottom plots illustrate errors against data and predicted values, with no strong patterns to be found. However, as in the Variogram model, all models seems to overestimate data under 50 and underestimate data above 150. This result is acceptable, and as we have plugged in most optional arguments, we will try improving models by other methods in later parts.

The fourth model, which is spherical, has the shortest interquartile range of errors, thus, could be considered the best models in maximum likelihood.

1f

Predict precipitation at A, B, C

```
# Predict A,B,C
preds_vr <- krige.conv(geo_trn, loc = tst[2:3], krige = krige.control(obj.model = vari.mat1.5))
```

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

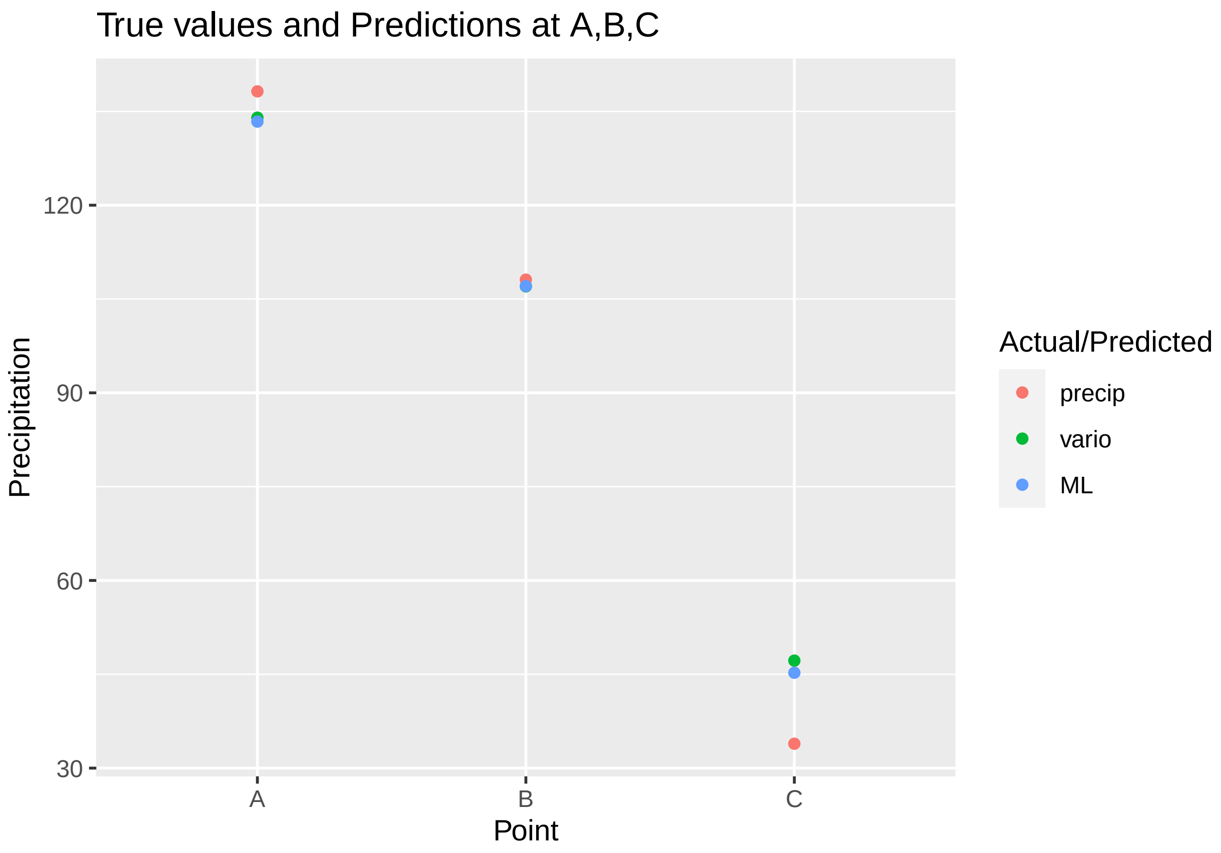
```r
preds_ml <- krige.conv(geo_trn, loc = tst[2:3], krige = krige.control(obj.model = ml_spher))
```

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: performing the Box-Cox data transformation
## krige.conv: back-transforming the predicted mean and variance
## krige.conv: back-transforming by simulating from the predictive.
##             (run the function a few times and check stability of the results.
## krige.conv: Kriging performed using global neighbourhood
```

```r
pred_table <- data.frame(true = tst[4], vario = preds_vr$predict,
    ML = preds_ml$predict, point = tst[5])

# Plot predictions and true values
pred_long <- reshape2::melt(pred_table)
```

```
## Using label as id variables
```

```r
ggplot(pred_long, aes(x = label, y = value, color = variable)) +
    geom_point() + labs(x = "Point", y = "Precipitation", color = "Actual/Predicted") +
    ggtitle("True values and Predictions at A,B,C")
```



Compare predictions and true values The actual values (red points) and predicted values (green by variogram and blue by ML) are shown in the above plot. The actual values and predictions are quite close at A and B, however, both models overestimated precipitation at C. Both predicted C to have around 45, however, observed figure at C was just above 30. At all three locations, both estimation methods produce closely similar prediction (in terms of the mean prediction).

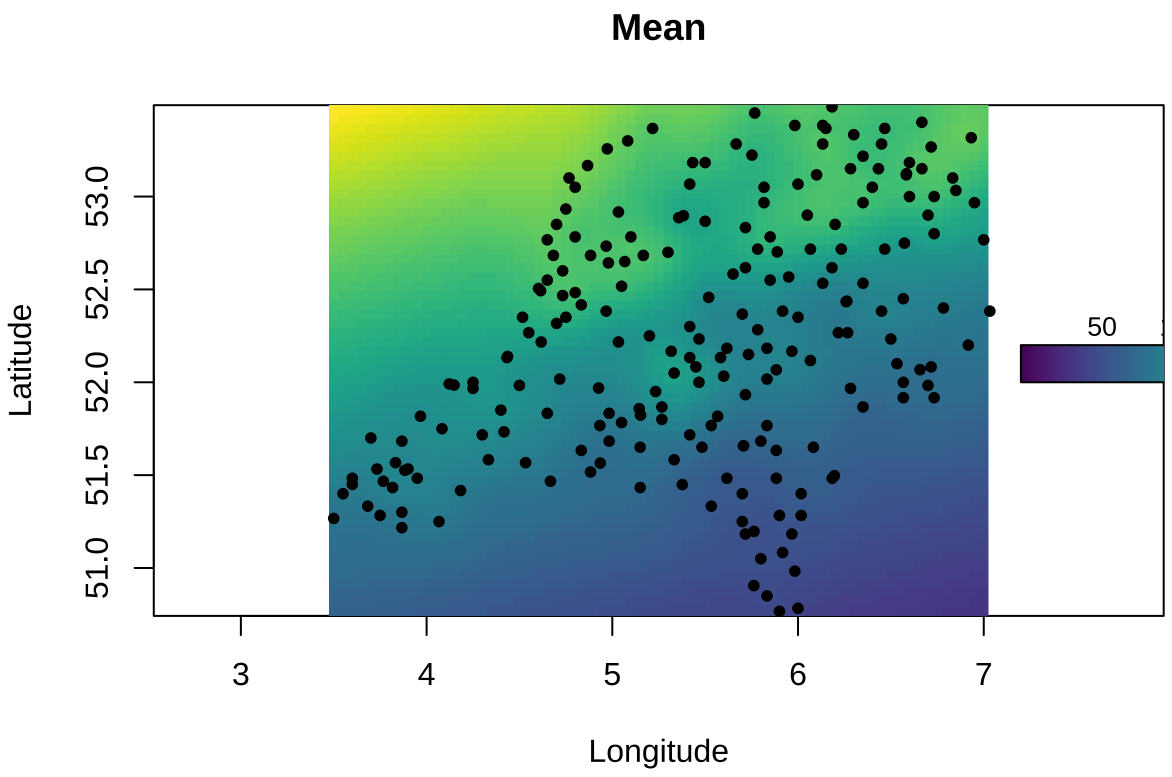
1g

Plot the mean and variance from maximum likelihood

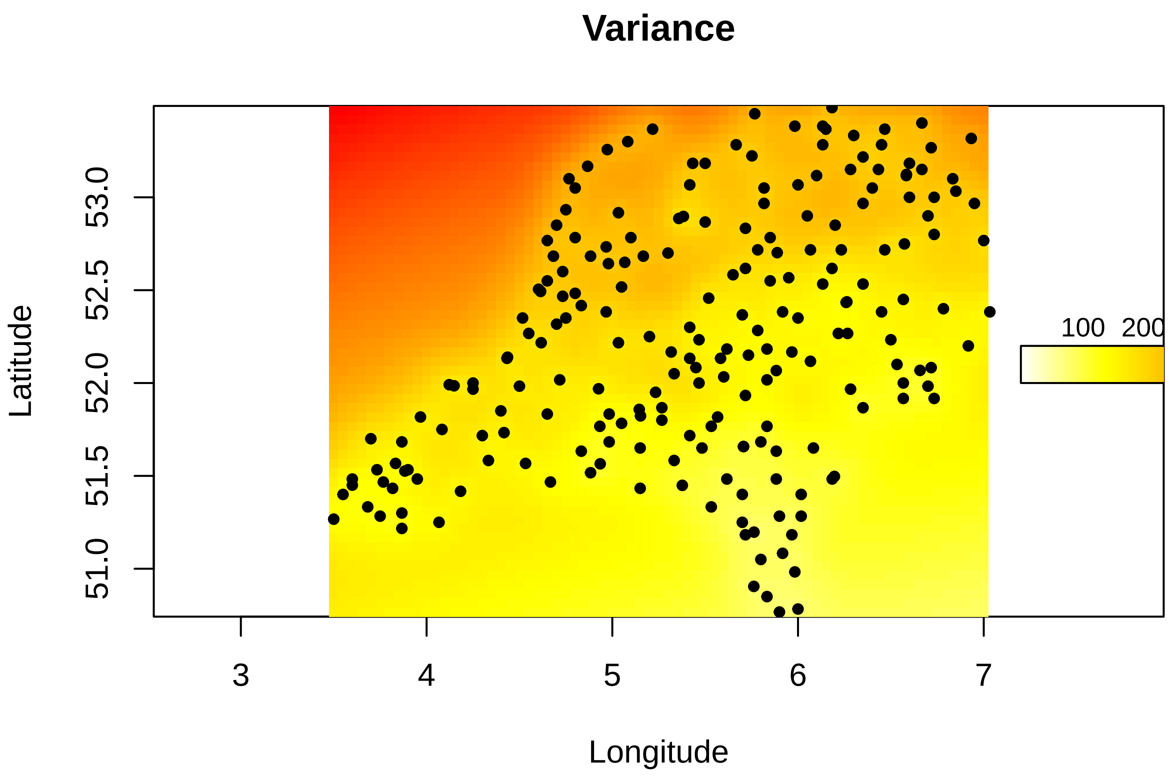
```r
# Create grid of prediction points
grd <- expand.grid(longitude = seq(min(geo_ntl$coords[, 1]),
    max(geo_ntl$coords[, 1]), by = 0.05), latitude = seq(min(geo_ntl$coords[,
    2]), max(geo_ntl$coords[, 2]), by = 0.05))

# Predict mean and variance at grid points using kriging
pred_grd <- krige.conv(geo_trn, loc = grd, krige = krige.control(obj.model = ml_spher))
```

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: performing the Box-Cox data transformation
## krige.conv: back-transforming the predicted mean and variance
## krige.conv: back-transforming by simulating from the predictive.
##             (run the function a few times and check stability of the results.
## krige.conv: Kriging performed using global neighbourhood
```

```r
# Plot mean and variance
image(pred_grd, col = viridis::viridis(100), zlim = c(0, max(c(pred_grd$predict))),
    coords.data = geo_trn[1]$coords, main = "Mean", xlab = "Longitude",
    ylab = "Latitude", x.leg = c(7.2, 9), y.leg = c(52, 52.2))
```



```r
image(pred_grd, values = pred_grd$krige.var, col = heat.colors(100)[100:1],
    zlim = c(0, max(c(pred_grd$krige.var))), coords.data = geo_trn[1]$coords,
    main = "Variance", xlab = "Longitude", ylab = "Latitude",
    x.leg = c(7.2, 9), y.leg = c(52, 52.2))
```



###
1h

Fit a Bayesian model using discrete priors.

From the estimates of multiple maximum likelihood models, we recall that the lambda is around 0.5, the nugget is more than 1 and the phi ranges from 0.1 to 0.5. Besides, from the previous analysis, we can have a reasonable assumption that the mean is the first order polynomial of coordinations and the covariance matrix is spherical.

We can use estimates of parameters from maximum likelihood spherical as reference.

```
ml_spher$parameters.summary
```

```
##            status    values
## beta0   estimated -233.5359
## beta1   estimated   -1.4849
## beta2   estimated    4.9883
## tausq   estimated    1.3076
## sigmasq estimated    1.1792
## phi     estimated    0.5180
## kappa       fixed    0.5000
## psiA        fixed    0.0000
## psiR        fixed    1.0000
## lambda  estimated    0.5114
```

Build Bayes GP with and without nugget

```
# Recreate a grid with degree = 0.1 to reduce burden on
# computer
grd2 <- expand.grid(longitude = seq(min(geo_ntl$coords[, 1]),
    max(geo_ntl$coords[, 1]), by = 0.1), latitude = seq(min(geo_ntl$coords[,
    2]), max(geo_ntl$coords[, 2]), by = 0.1))
```

```r
ex.grid <- as.matrix(grd2)

# Bayes GP without nugget
ex.bayes_1 <- krige.bayes(geodata = geo_trn, loc = ex.grid, model = model.control(trend.d = "1st",
    trend.l = "1st", cov.m = "spherical", lambda = 0.5), prior = prior.control(phi.discrete = seq(0,
    1, l = 21), phi.prior = "reciprocal"))
```

```
## krige.bayes: Box-Cox's transformation performed for lambda = 0.5
## krige.bayes: model with mean given by a 1st order polynomial on the coordinates
## krige.bayes: computing the discrete posterior of phi/tausq.rel
## krige.bayes: computing the posterior probabilities.
##              Number of parameter sets:  21
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
##
## krige.bayes: sampling from posterior distribution
## krige.bayes: sample from the (joint) posterior of phi and tausq.rel
##             [,1]
## phi       5e-02
## tausq.rel 0e+00
## frequency 1e+03
##
## krige.bayes: starting prediction at the provided locations
## krige.bayes: phi/tausq.rel samples for the predictive are same as for the posterior
## krige.bayes: computing moments of the predictive distribution
## krige.bayes: sampling from the predictive
##              Number of parameter sets:  1
## 1,
## krige.bayes: Box-Cox data transformation performed.
##              Simulations back-transformed to the original scale
## krige.bayes: preparing summaries of the predictive distribution
```

```r
# Bayes GP with nugget
ex.bayes_2 <- krige.bayes(geodata = geo_trn, loc = ex.grid, model = model.control(trend.d = "1st",
    trend.l = "1st", cov.m = "spherical", lambda = 0.5), prior = prior.control(phi.discrete = seq(0,
    1, l = 21), phi.prior = "reciprocal", tausq.rel.discrete = seq(1,
    2, l = 11), tausq.rel.prior = "reciprocal"))
```

```
## krige.bayes: Box-Cox's transformation performed for lambda = 0.5
## krige.bayes: model with mean given by a 1st order polynomial on the coordinates
## krige.bayes: computing the discrete posterior of phi/tausq.rel
## krige.bayes: computing the posterior probabilities.
##              Number of parameter sets:  231
## 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 201, 211, 2
##
## krige.bayes: sampling from posterior distribution
## krige.bayes: sample from the (joint) posterior of phi and tausq.rel
##           [,1]  [,2] [,3]  [,4] [,5]  [,6] [,7]  [,8] [,9] [,10] [,11] [,12]
## phi        0.4  0.45  0.5  0.55  0.6  0.65  0.7  0.75  0.8  0.85   0.9  0.95
## tausq.rel  1.0  1.00  1.0  1.00  1.0  1.00  1.0  1.00  1.0  1.00   1.0  1.00
## frequency  4.0 12.00 30.0 34.00 25.0 13.00  7.0 10.00 10.0 10.00   3.0  6.00
##           [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23]
## phi           1  0.35   0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8
## tausq.rel     1  1.10   1.1  1.10   1.1  1.10   1.1  1.10   1.1  1.10   1.1
## frequency     9  1.00   5.0 20.00  28.0 27.00  19.0 16.00   8.0  3.00   9.0
##           [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34]
## phi        0.85   0.9  0.95   1.0   0.4  0.45   0.5  0.55   0.6  0.65   0.7
## tausq.rel  1.10   1.1  1.10   1.1   1.2  1.20   1.2  1.20   1.2  1.20   1.2
## frequency 13.00   8.0 13.00   8.0   2.0 20.00  20.0 21.00  11.0 19.00   6.0
##           [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45]
## phi        0.75   0.8  0.85   0.9  0.95   1.0  0.35   0.4  0.45   0.5  0.55
## tausq.rel  1.20   1.2  1.20   1.2  1.20   1.2  1.30   1.3  1.30   1.3  1.30
## frequency  4.00   4.0 14.00   5.0  4.00   4.0  2.00   1.0 13.00  16.0 24.00
##           [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56]
## phi         0.6  0.65   0.7  0.75   0.8  0.85   0.9  0.95   1.0  0.35   0.4
## tausq.rel   1.3  1.30   1.3  1.30   1.3  1.30   1.3  1.30   1.3  1.40   1.4
## frequency  22.0  8.00   5.0 10.00   3.0  8.00   6.0  9.00   5.0  1.00   2.0
##           [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67]
## phi        0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8  0.85   0.9  0.95
## tausq.rel  1.40   1.4  1.40   1.4  1.40   1.4  1.40   1.4  1.40   1.4  1.40
## frequency 10.00  10.0 15.00  20.0  5.00   6.0  4.00   5.0  3.00   7.0  2.00
##           [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78]
## phi         1.0   0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8  0.85
## tausq.rel   1.4   1.5  1.50   1.5  1.50   1.5  1.50   1.5  1.50   1.5  1.50
## frequency   3.0   1.0  4.00   8.0  9.00  15.0  7.00   5.0  6.00   8.0  3.00
##           [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89]
## phi         0.9  0.95   1.0   0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75
## tausq.rel   1.5  1.50   1.5   1.6  1.60   1.6  1.60   1.6  1.60   1.6  1.60
## frequency   6.0  4.00   1.0   1.0  3.00  13.0  9.00   8.0  6.00   4.0  4.00
##           [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## phi         0.8  0.85   0.9  0.95   1.0   0.4  0.45   0.5  0.55   0.6   0.65
## tausq.rel   1.6  1.60   1.6  1.60   1.6   1.7  1.70   1.7  1.70   1.7   1.70
## frequency   2.0  2.00   3.0  5.00   4.0   1.0  2.00   7.0  6.00  10.0   6.00
##           [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108] [,109] [,110]
## phi          0.7    0.8   0.85    0.9   0.95    0.3   0.45    0.5   0.55    0.6
## tausq.rel    1.7    1.7   1.70    1.7   1.70    1.8   1.80    1.8   1.80    1.8
## frequency    1.0    4.0   3.00    4.0   2.00    1.0   1.00    1.0   7.00    6.0
##           [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118] [,119] [,120]
## phi         0.65    0.7   0.75    0.8   0.85    0.9   0.95    1.0   0.45    0.5
## tausq.rel   1.80    1.8   1.80    1.8   1.80    1.8   1.80    1.8   1.90    1.9
## frequency   7.00    2.0   4.00    3.0   4.00    3.0   2.00    2.0   1.00    4.0
##           [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128] [,129] [,130]
## phi         0.55    0.6   0.65    0.7   0.75    0.8   0.85    0.9   0.95   0.45
## tausq.rel   1.90    1.9   1.90    1.9   1.90    1.9   1.90    1.9   1.90   2.00
## frequency   3.00    4.0   5.00    2.0   4.00    5.0   2.00    1.0   1.00   2.00
##           [,131] [,132] [,133] [,134] [,135] [,136] [,137] [,138] [,139]
## phi         0.55   0.65    0.7   0.75    0.8   0.85    0.9   0.95      1
## tausq.rel   2.00   2.00    2.0   2.00    2.0   2.00    2.0   2.00      2
## frequency   8.00   4.00    4.0   2.00    1.0   2.00    3.0   1.00      1
##
## krige.bayes: starting prediction at the provided locations
## krige.bayes: phi/tausq.rel samples for the predictive are same as for the posterior
## krige.bayes: computing moments of the predictive distribution
## krige.bayes: sampling from the predictive
##              Number of parameter sets:  139
## 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131,
## krige.bayes: Box-Cox data transformation performed.
##              Simulations back-transformed to the original scale
## krige.bayes: preparing summaries of the predictive distribution
```

We will summarise our posterior distributions, compare them against each other and compare them with earlier estimates.

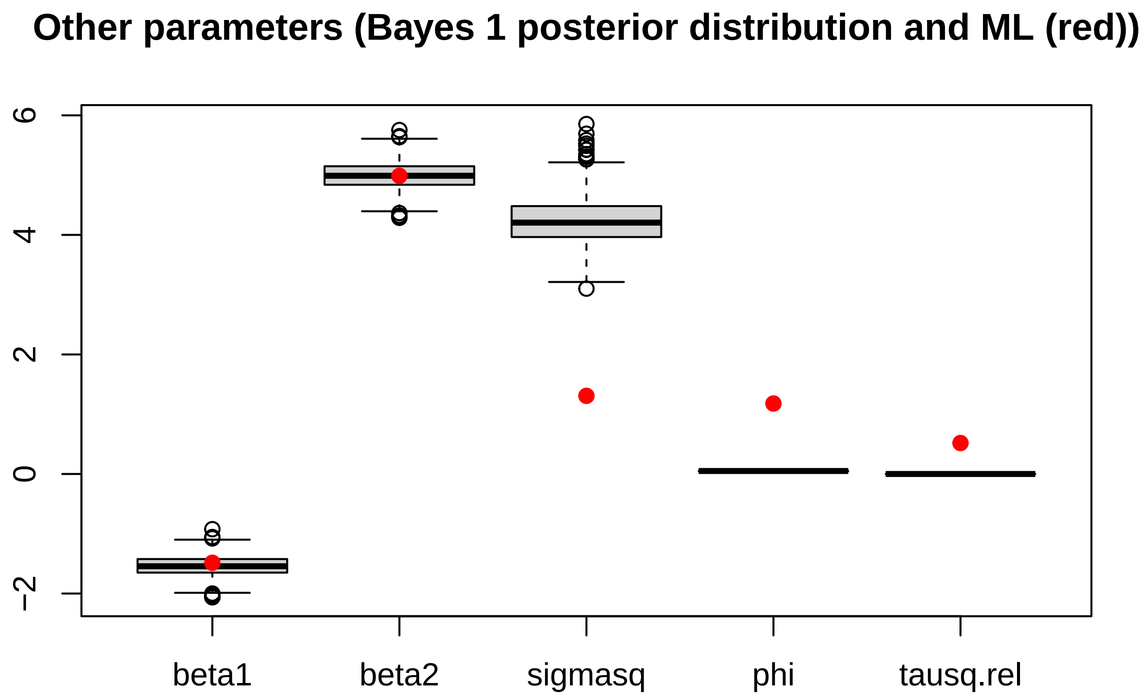
From the four plots below, the posterior of the model 2 (with nugget) is quite close with the parameter estimates from the maximum likelihood model, especially betas parameters and variance. For phi and nugget, maximum likelihood estimates are out of interquartile range of distributions from Bayes (with nugget), however, they are not too far away.

Meanwhile, the posterior distribution of the model 1 (without nugget) only closely matches with the maximum likelihood estimates in beta parameters. Its variance and correlation length are far from the estimates by ML.

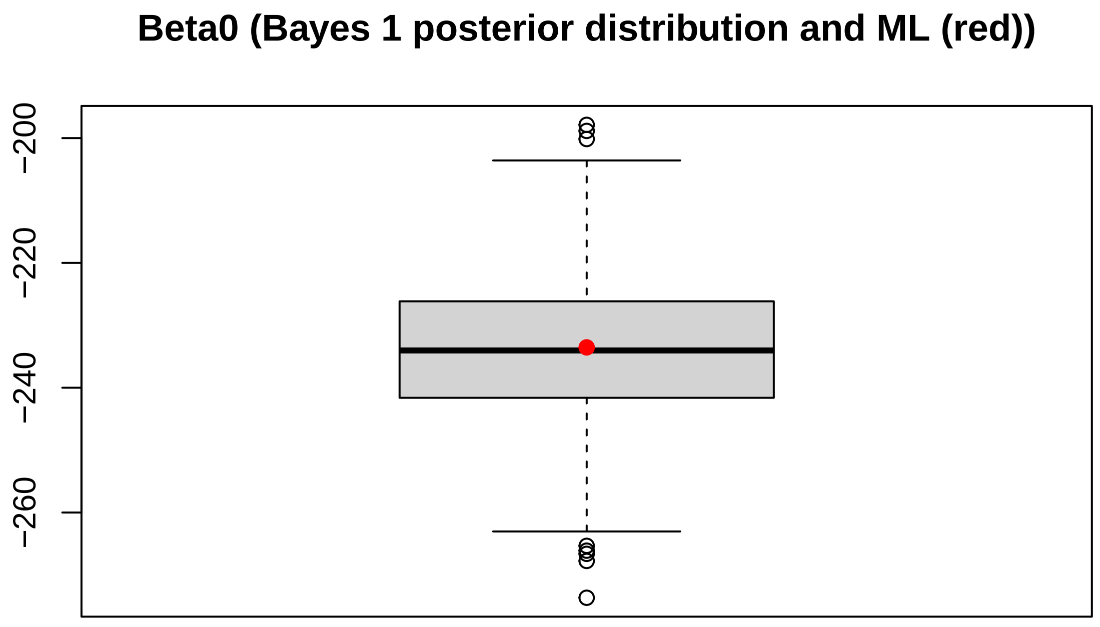
Comparing Bayesian with and without nugget, we can see that the Bayesian 1 without nugget produce much higher estimates of variance (mean around 4) than the Bayesian 2 with nugget (mean around 1). Distribution of correlation length and nugget by Bayesian 2 are sensible, quite close with our earlier estimates. Both Bayesian models estimate quite similar betas parameters.

We will use cross-validation to validate both models

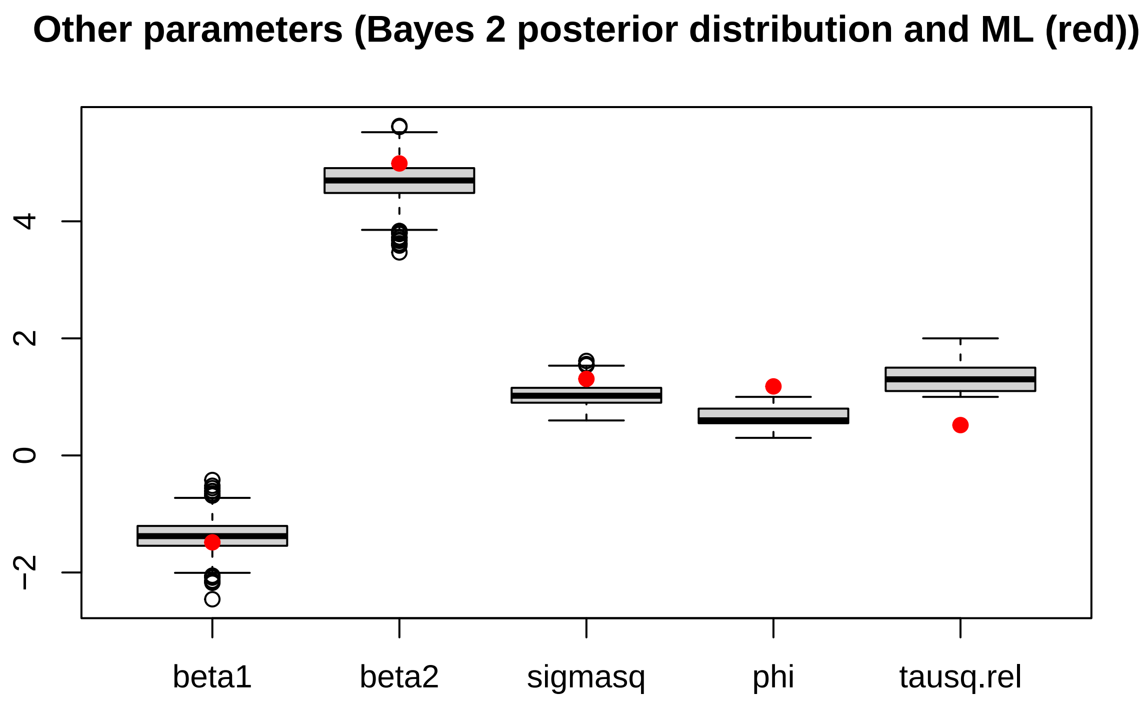
```
boxplot(ex.bayes_1$posterior$sample[2:6], main = "Other parameters (Bayes 1 posterior distribution and 
points(ml_spher$parameters.summary[2:6, 2], col = "red", pch = 19)
```



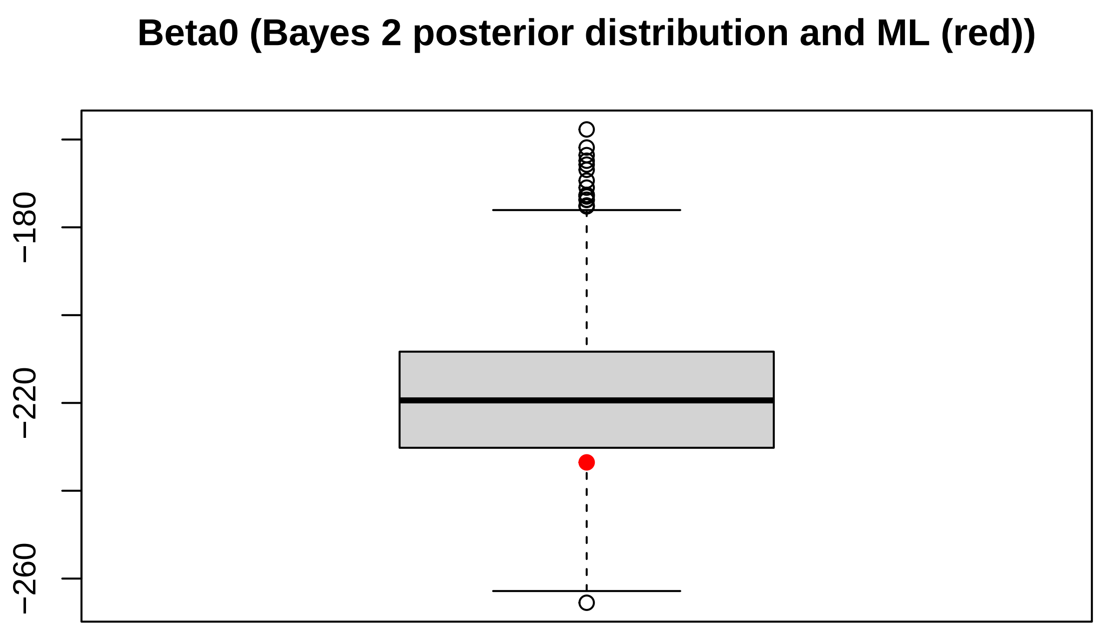
```
boxplot(ex.bayes_1$posterior$sample[1], main = "Beta0 (Bayes 1 posterior distribution and ML (red))")
points(ml_spher$parameters.summary[1, 2], col = "red", pch = 19)
```



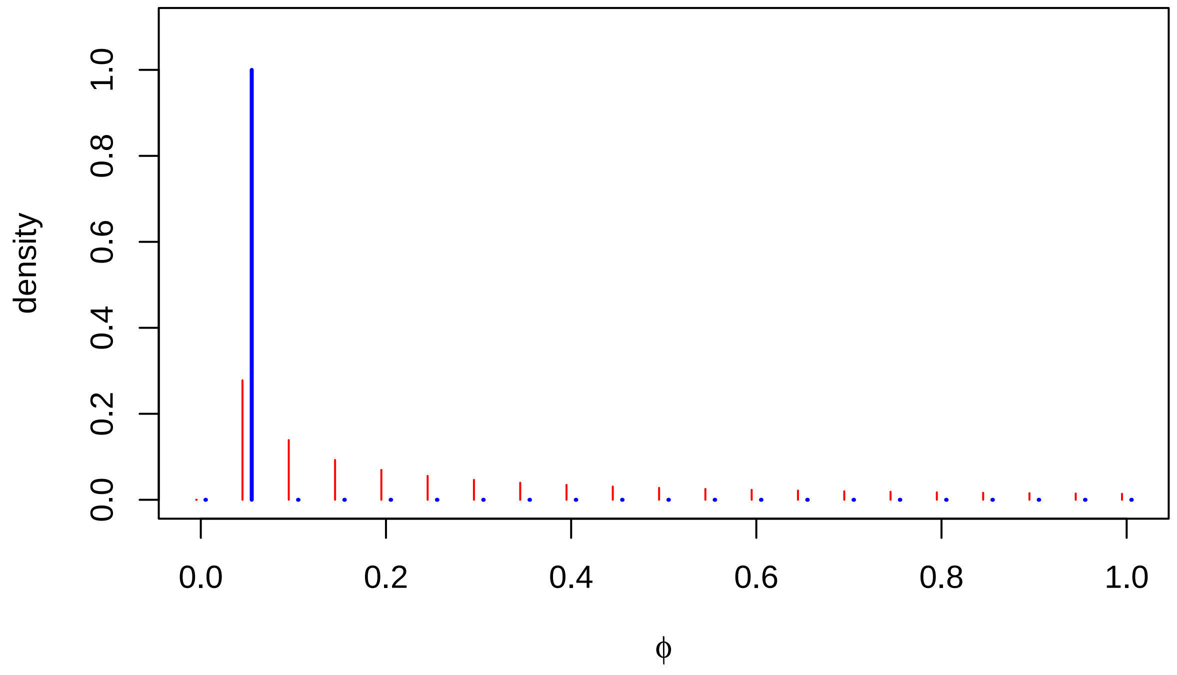
```
boxplot(ex.bayes_2$posterior$sample[2:6], main = "Other parameters (Bayes 2 posterior distribution and 
points(ml_spher$parameters.summary[2:6, 2], col = "red", pch = 19)
```



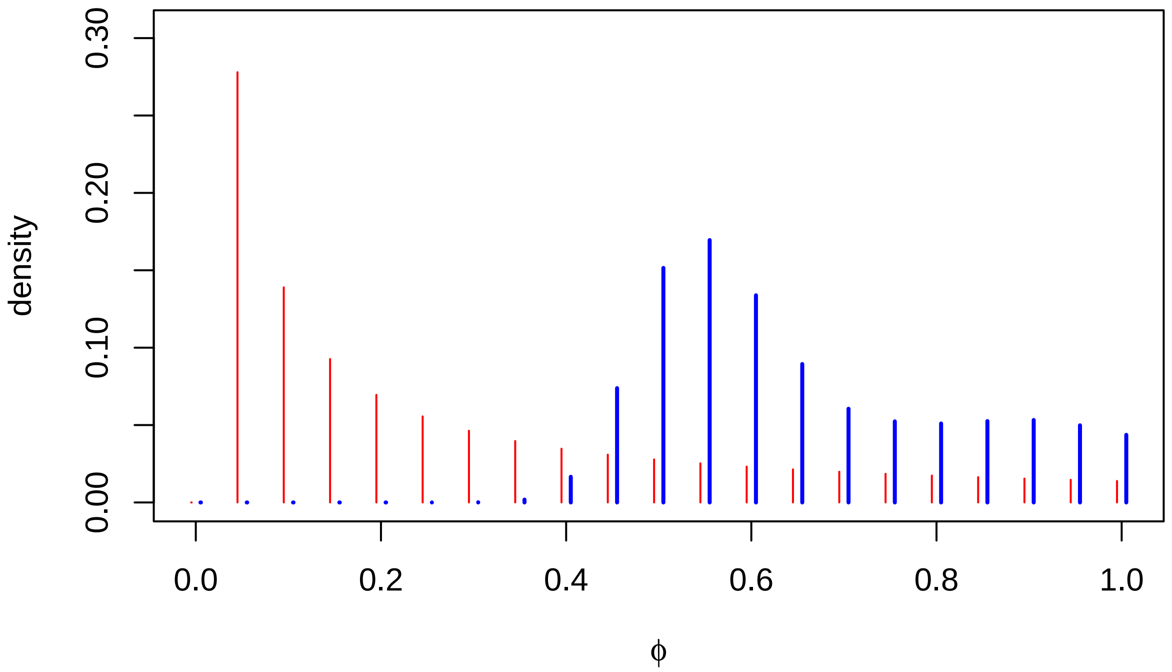
```
boxplot(ex.bayes_2$posterior$sample[1], main = "Beta0 (Bayes 2 posterior distribution and ML (red))")
points(ml_spher$parameters.summary[1, 2], col = "red", pch = 19)
```

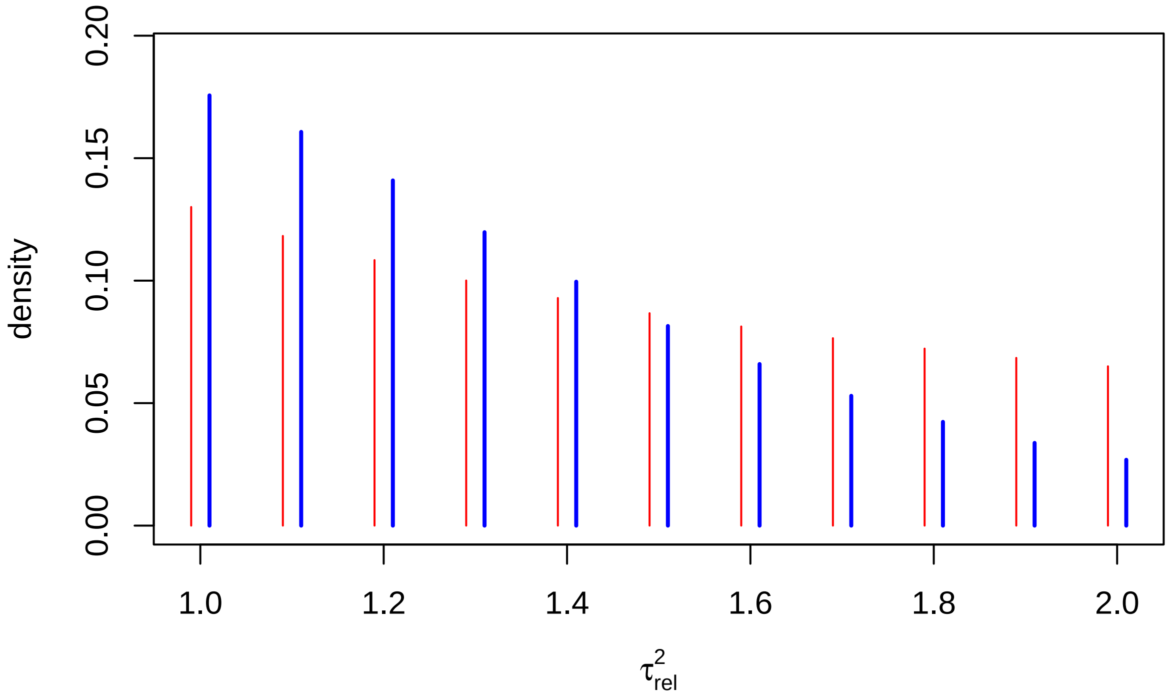


```
# Prior and posterior for the parameter phi
plot(ex.bayes_1, type = "h", tausq.rel = FALSE, col = c("red",
    "blue"))
```

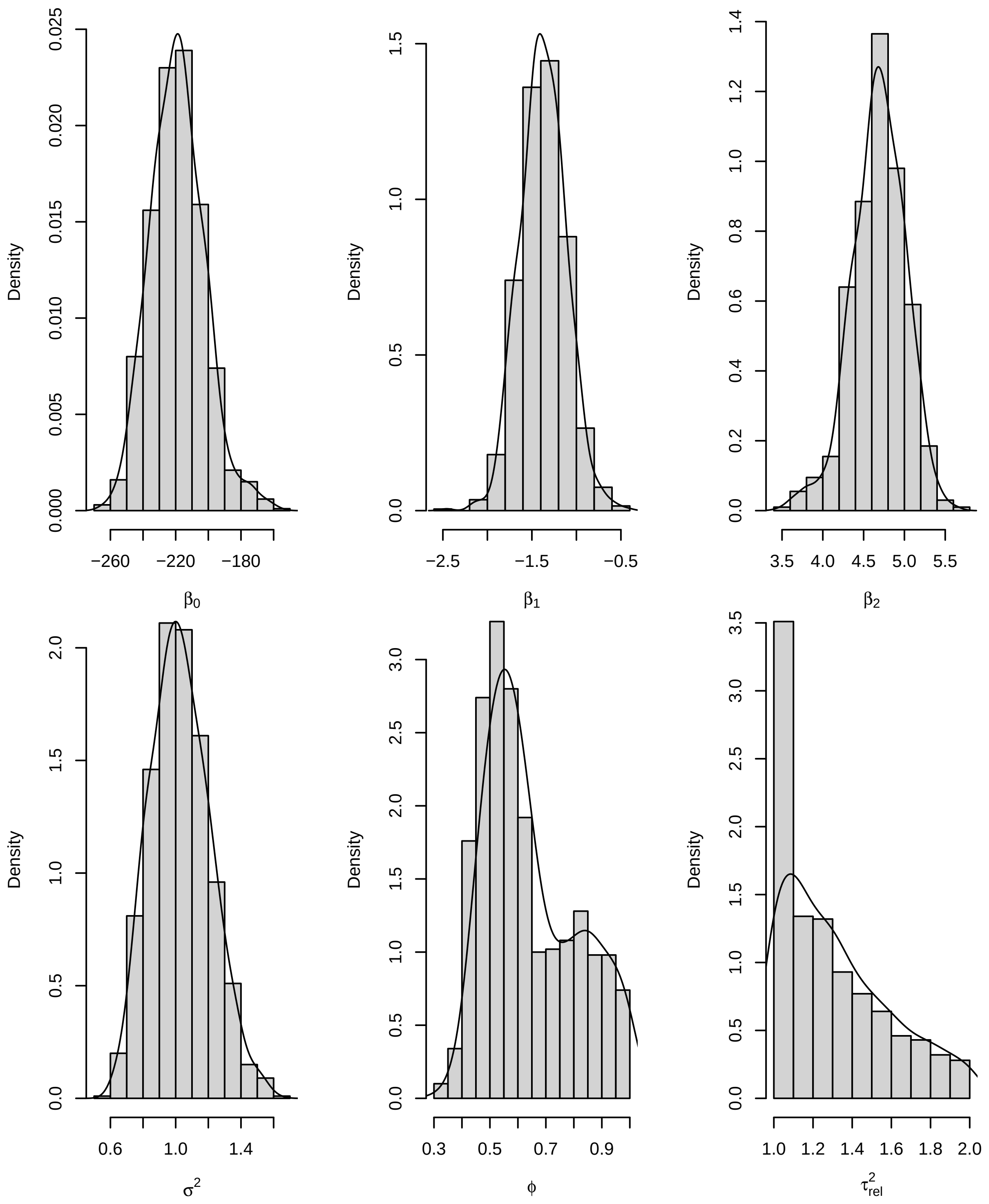


```
# Prior and posterior for the parameter phi
plot(ex.bayes_2, type = "h", tausq.rel = TRUE, col = c("red",
    "blue"))
```





```
# Plot histograms with samples from the posterior
par(mfrow = c(1, 3))
hist(ex.bayes_2)
```



For each model, produce predictions for locations A, B and C.

```
# Predict A,B,C with Bayes
ex.bayes_1_pred <- krige.bayes(geodata = geo_trn, loc = tst[2:3],
    model = model.control(trend.d = "1st", trend.l = "1st", cov.m = "spherical",
        lambda = 0.5), prior = prior.control(phi.discrete = seq(0,
        1, l = 21), phi.prior = "reciprocal"))
```

```
## krige.bayes: Box-Cox's transformation performed for lambda = 0.5
## krige.bayes: model with mean given by a 1st order polynomial on the coordinates
## krige.bayes: computing the discrete posterior of phi/tausq.rel
## krige.bayes: computing the posterior probabilities.
##              Number of parameter sets:  21
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
##
## krige.bayes: sampling from posterior distribution
## krige.bayes: sample from the (joint) posterior of phi and tausq.rel
##             [,1]
## phi       5e-02
## tausq.rel 0e+00
## frequency 1e+03
##
## krige.bayes: starting prediction at the provided locations
## krige.bayes: phi/tausq.rel samples for the predictive are same as for the posterior
## krige.bayes: computing moments of the predictive distribution
## krige.bayes: sampling from the predictive
##              Number of parameter sets:  1
## 1,
## krige.bayes: Box-Cox data transformation performed.
##              Simulations back-transformed to the original scale
## krige.bayes: preparing summaries of the predictive distribution
```

```
ex.bayes_2_pred <- krige.bayes(geodata = geo_trn, loc = tst[2:3],
    model = model.control(trend.d = "1st", trend.l = "1st", cov.m = "spherical",
        lambda = 0.5), prior = prior.control(phi.discrete = seq(0,
        1, l = 21), phi.prior = "reciprocal", tausq.rel.discrete = seq(1,
        2, l = 11), tausq.rel.prior = "reciprocal"))
```

```
## krige.bayes: Box-Cox's transformation performed for lambda = 0.5
## krige.bayes: model with mean given by a 1st order polynomial on the coordinates
## krige.bayes: computing the discrete posterior of phi/tausq.rel
## krige.bayes: computing the posterior probabilities.
##              Number of parameter sets:  231
## 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 201, 211, 2
##
## krige.bayes: sampling from posterior distribution
## krige.bayes: sample from the (joint) posterior of phi and tausq.rel
##            [,1]  [,2] [,3]  [,4] [,5]  [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## phi         0.4  0.45  0.5  0.55  0.6  0.65  0.7 0.75  0.8  0.85   0.9  0.95
## tausq.rel   1.0  1.00  1.0  1.00  1.0  1.00  1.0 1.00  1.0  1.00   1.0  1.00
## frequency   4.0 16.00 34.0 45.00 20.0 10.00  4.0 4.00  8.0 12.00   8.0 15.00
##            [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23]
## phi            1   0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8  0.85
## tausq.rel      1   1.1  1.10   1.1  1.10   1.1  1.10   1.1  1.10   1.1  1.10
## frequency      7   2.0 15.00  28.0 35.00  18.0  7.00  13.0  9.00   9.0  8.00
##            [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34]
## phi          0.9  0.95   1.0   0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75
## tausq.rel    1.1  1.10   1.1   1.2  1.20   1.2  1.20   1.2  1.20   1.2  1.20
## frequency    7.0  5.00   8.0   1.0  9.00  29.0 21.00  24.0 21.00   6.0  7.00
##            [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45]
## phi          0.8  0.85   0.9  0.95   1.0   0.4  0.45   0.5  0.55   0.6  0.65
## tausq.rel    1.2  1.20   1.2  1.20   1.2   1.3  1.30   1.3  1.30   1.3  1.30
## frequency    9.0  6.00   5.0  3.00   2.0   3.0  8.00  22.0 21.00  18.0 13.00
##            [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56]
## phi         0.7  0.75   0.8  0.85   0.9  0.95   1.0  0.35   0.4  0.45   0.5
## tausq.rel   1.3  1.30   1.3  1.30   1.3  1.30   1.3  1.40   1.4  1.40   1.4
## frequency   7.0  2.00   3.0  6.00   3.0  2.00   7.0  1.00   2.0  7.00  14.0
##           [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67]
## phi        0.55   0.6  0.65   0.7  0.75   0.8  0.85   0.9  0.95   1.0  0.45
## tausq.rel  1.40   1.4  1.40   1.4  1.40   1.4  1.40   1.4  1.40   1.4  1.50
## frequency 17.00  19.0  5.00  10.0  7.00   6.0  8.00   6.0  6.00   3.0  3.00
##           [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78]
## phi         0.5  0.55   0.6  0.65   0.7  0.75   0.8  0.85   0.9  0.95   1.0
## tausq.rel   1.5  1.50   1.5  1.50   1.5  1.50   1.5  1.50   1.5  1.50   1.5
## frequency  10.0 16.00   6.0 11.00   8.0  4.00   3.0  3.00   5.0  6.00   5.0
##           [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89]
## phi         0.4  0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8   0.9   1.0
## tausq.rel   1.6  1.60   1.6  1.60   1.6  1.60   1.6  1.60   1.6   1.6   1.6
## frequency   3.0  2.00   4.0 10.00  10.0  4.00   3.0  7.00   1.0   3.0   1.0
##           [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## phi        0.45   0.5  0.55   0.6  0.65   0.7  0.75   0.8  0.85   0.9   0.95
## tausq.rel  1.70   1.7  1.70   1.7  1.70   1.7  1.70   1.7  1.70   1.7   1.70
## frequency  1.00   3.0 10.00   6.0  7.00   2.0  3.00   1.0  2.00   4.0   4.00
##           [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108] [,109] [,110]
## phi          1.0   0.45    0.5   0.55    0.6   0.65    0.7   0.75    0.8   0.85
## tausq.rel    1.7   1.80    1.8   1.80    1.8   1.80    1.8   1.80    1.8   1.80
## frequency    1.0   1.00    4.0  10.00    9.0   1.00    3.0   2.00    5.0   5.00
##           [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118] [,119] [,120]
## phi          0.9   0.95    1.0   0.45    0.5   0.55    0.6   0.65    0.7   0.75
## tausq.rel    1.8   1.80    1.8   1.90    1.9   1.90    1.9   1.90    1.9   1.90
## frequency    3.0   1.00    1.0   2.00    3.0   5.00    8.0   5.00    4.0   3.00
##           [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128] [,129] [,130]
## phi          0.8   0.85   0.95    1.0   0.45    0.5   0.55    0.6   0.65    0.7
## tausq.rel    1.9   1.90   1.90    1.9   2.00    2.0   2.00    2.0   2.00    2.0
## frequency    4.0   3.00   2.00    1.0   1.00    2.0   4.00    2.0   1.00    4.0
##           [,131] [,132] [,133] [,134] [,135]
## phi         0.75    0.8   0.85    0.9      1
## tausq.rel   2.00    2.0   2.00    2.0      2
## frequency   1.00    1.0   1.00    2.0      4
##
## krige.bayes: starting prediction at the provided locations
## krige.bayes: phi/tausq.rel samples for the predictive are same as for the posterior
## krige.bayes: computing moments of the predictive distribution
## krige.bayes: sampling from the predictive
##              Number of parameter sets:  135
## 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131,
## krige.bayes: Box-Cox data transformation performed.
##              Simulations back-transformed to the original scale
## krige.bayes: preparing summaries of the predictive distribution
```

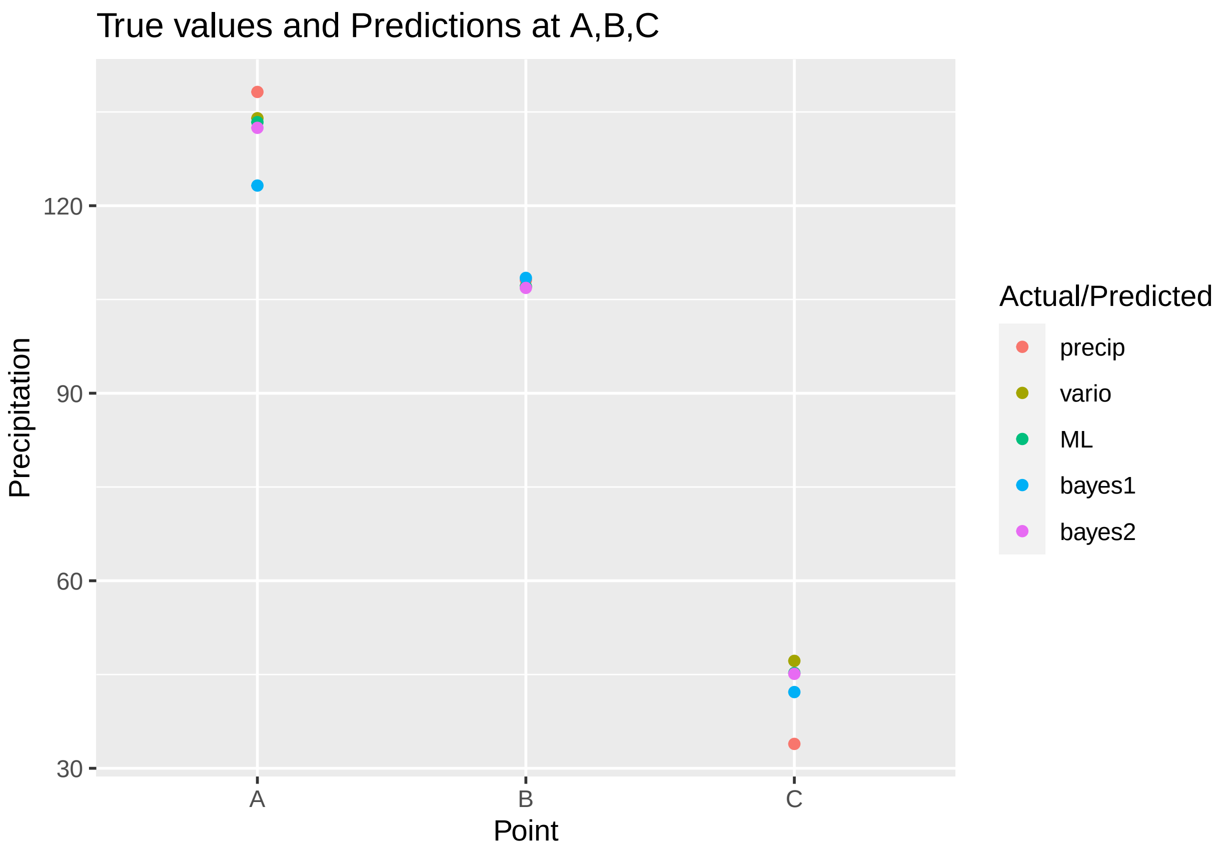
```
# Plot prediction
pred_table$bayes1 <- ex.bayes_1_pred$predictive$mean
pred_table$bayes2 <- ex.bayes_2_pred$predictive$mean

# Prediction results from all models
pred_table
```

```
##   precip     vario        ML label    bayes1    bayes2
## 1  138.2 133.99475 133.36295     A 123.22181 132.46042
## 2  108.1 107.01734 107.06540     B 108.44782 106.85327
## 3   33.9  47.18596  45.24743     C  42.19173  45.11035
```

```
# Plot predictions and true values
pred_long <- reshape2::melt(pred_table)
```

```
## Using label as id variables
```

```
ggplot(pred_long, aes(x = label, y = value, color = variable)) +
    geom_point() + labs(x = "Point", y = "Precipitation", color = "Actual/Predicted") +
    ggtitle("True values and Predictions at A,B,C")
```



As can be seen from the plot, Bayesian without nugget poorly predicts precipitation at A, but performs similarly with other models at B and C, especially at C, its estimate is closest to the actual value. Meanwhile, Bayesian 2 performs similarly with the other two models at predicting values at A, B and C.