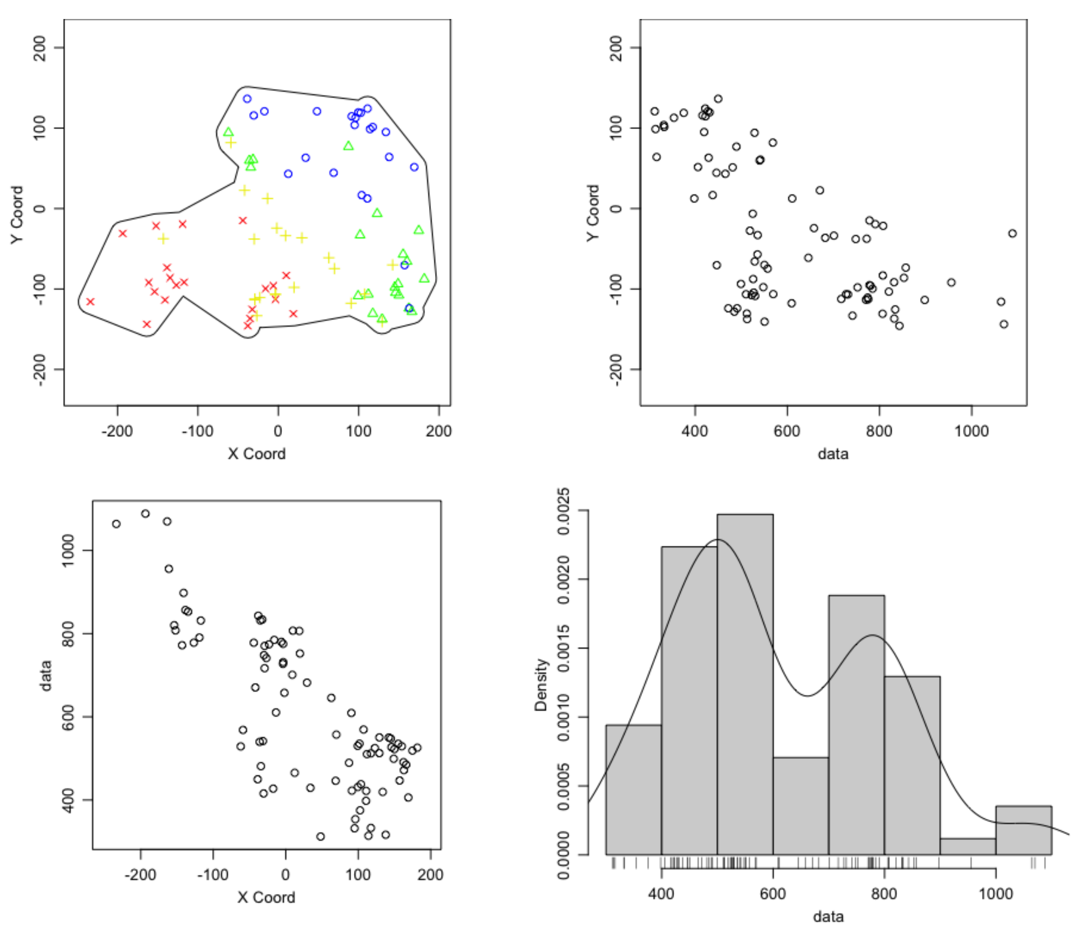
Exercise 1: Load the data into memory, data(wolfcamp); add a border to the data set, plot it, discuss briefly. What are these data?

Solution:
Loading the data and computing the border with my get.borders() function we get the following plot of the data,

Figure 1: Plot of Wolfcamp data with border.



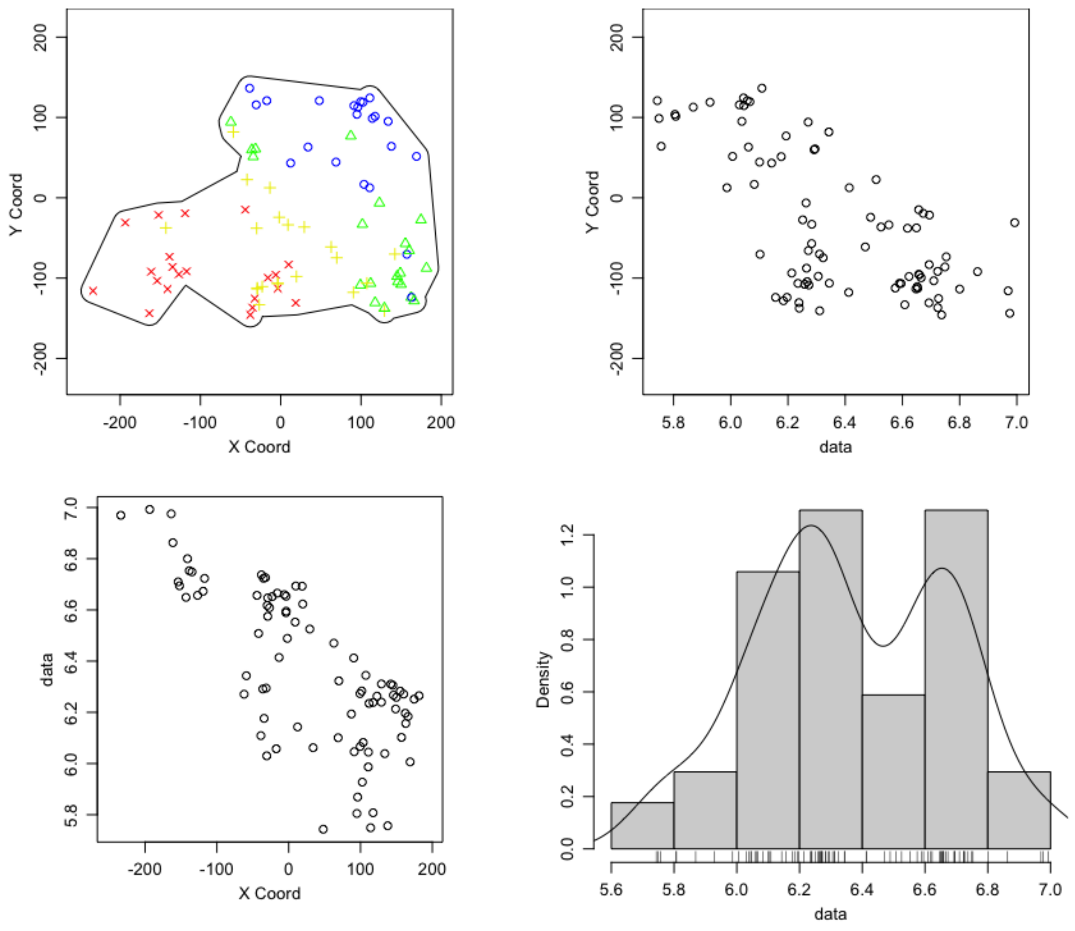
Code:

```
> wolfcamp$borders <- get.borders(
                      cbind(wolfcamp$coords[,1],wolfcamp$coords[,2]),
                      concavity_param = 3, frac = 15)
> plot(wolfcamp)
> ? wolfcamp
```

From the description provided by the '? wolfcamp' command we find that the data are Piezometric head measurements at various locations in the Wolfcamp aquifer located in Texas. This data is a measure of pressure in an aquifer computed from the height of water in a tube.

Recall that this same dataset was discussed in question question #4 of homework #2. In that exercise we concluded that there is a trend of increasing pressure as we move in the south west direction. We stated that the partial dependence plots corroborated that assertion. We also found the data to be slightly right skewed, and log transforming it produced a more centered, albeit still bimodal density histogram (We won't be using this transformation).

Figure 2: Plot of log transformed Wolfcamp data with border.



Code:

```
> wolfcampLog <- wolfcamp
> wolfcampLog$data <- log(wolfcampLog$data)
> plot(wolfcampLog)
```

Exercise 2: Act as a frequentist; fit a spatial model to these data, and obtain a smoothed map. Did you include a trend term or terms? Did you need to log-transform the data? Explain your reasoning. It might be helpful to show result from fitting one or two multiple linear regression models.
Be sure to discuss how you selected a type of semivariogram and how you estimated it's parameters. Be sure to state your final fitted model. Discuss briefly.

Solution:
To elaborate on the discussion on whether or not we should transform the data, I decided to perform a box-cox power analysis to see if we might find a worthwhile transformation. In doing so I found that we might want to consider the root transform, as well as the log transform. A Shapiro-Wilks test of normality on all three candidate transforms concludes that they are sufficiently normal and looking at the qq norm plot, I don't see a worthwhile difference between the transformations, so leaving the data alone seems to be the most parsimonious and interpretable option.
Code:

```
############## Box-Cox Power Analysis
> WolfData <- wolfcamp$data
> lon <- wolfcamp$coords[,1]
> lat <- wolfcamp$coords[,2]

## This tell's that we should consider a root transform on the data.
> summary(powerTransform(lm(WolfData ~ lat + lon )))
bcPower Transformation to Normality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1    0.5064         0.5       0.1132       0.8996

## The second order model says leave it alone.
> summary(powerTransform(lm(WolfData ~ lat + lon +
                            I(lat^2) + I(lon^2) +
                            I(lon*lat))))
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1    0.5932           1       0.1296       1.0569


## Shapiro-Wilks test of normality.
> shapiro.test(wolfcamp$data)

        Shapiro-Wilk normality test

data:  wolfcamp$data
W = 0.94696, p-value = 0.001586

> shapiro.test(sqrt(wolfcamp$data))

        Shapiro-Wilk normality test

data:  sqrt(wolfcamp$data)
W = 0.96266, p-value = 0.01468
```

```
> shapiro.test(log(wolfcamp$data))

        Shapiro-Wilk normality test

data:  log(wolfcamp$data)
W = 0.96761, p-value = 0.03101


## Generating qq-norm plots to compare.
> qqPlot(wolfcamp$data)
> qqPlot(sqrt(wolfcamp$data))
> qqPlot(log(wolfcamp$data))
```

Figure 3: QQ normPlot of Wolfcamp data.

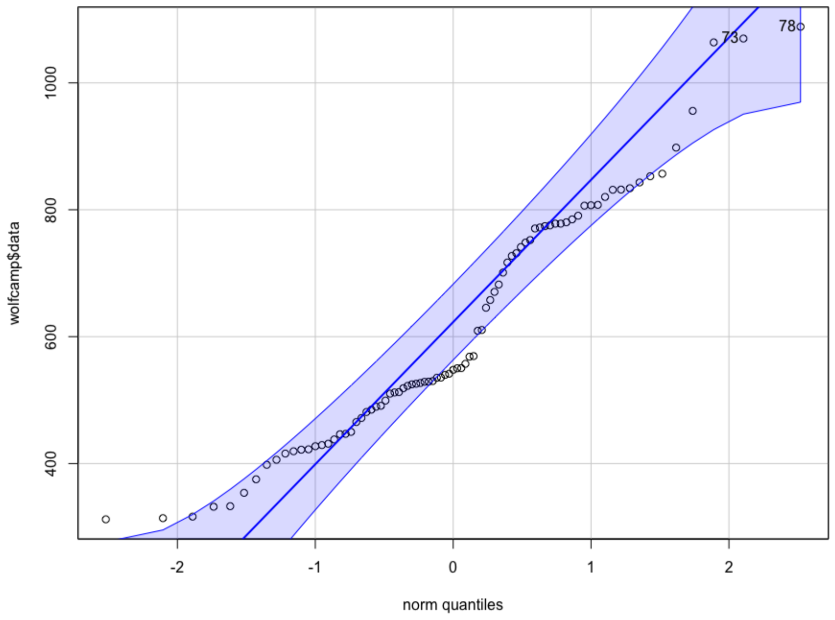


Figure 4: QQ normPlot of root transformed Wolfcamp data.

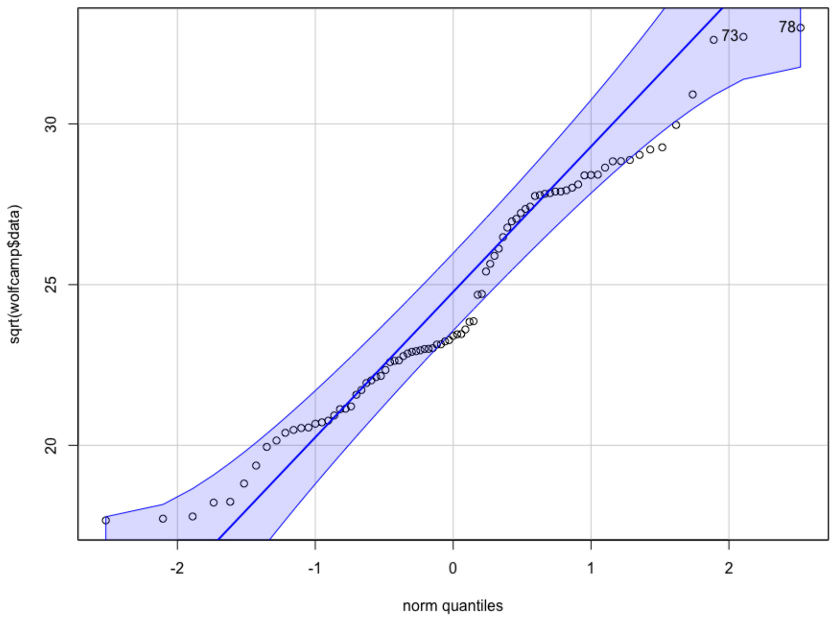
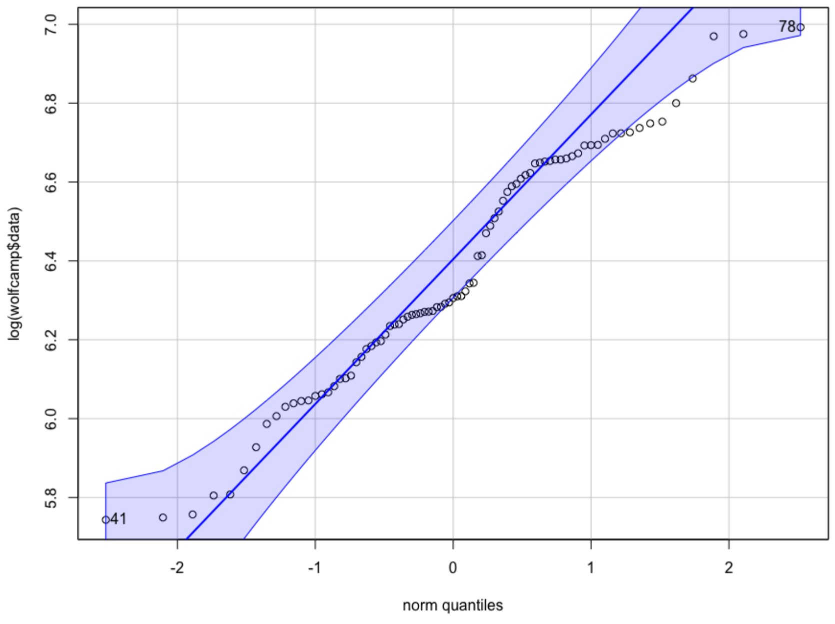


Figure 5: QQ normPlot of log transformed Wolfcamp data..



To determine if any trend term might be appropriate, a partial F test using the anova()

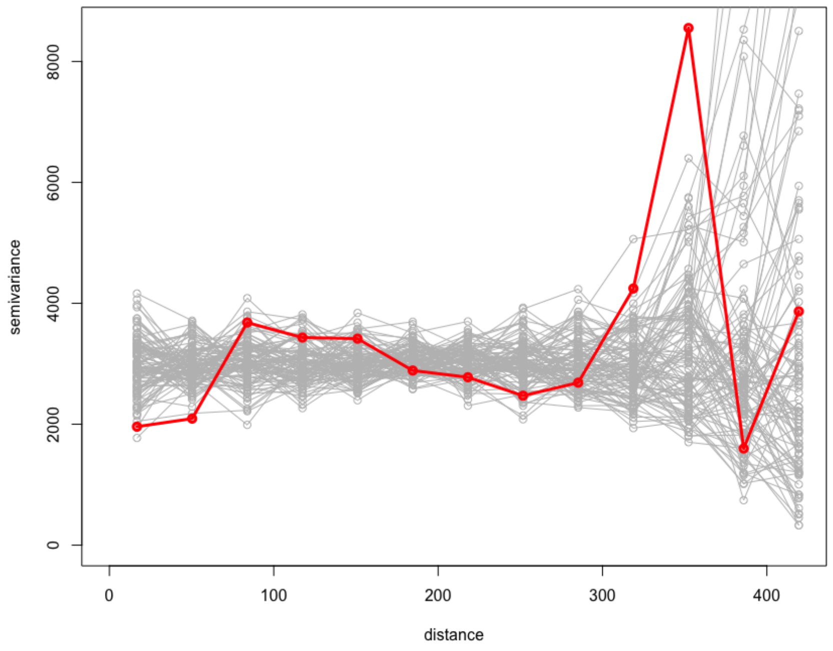
function states that the second order model explains a significant amount of the variance. Looking at the summary report we can see which terms contain that significance, and we find that the interaction and second order latitude are the most significant of the second order terms. Moving forward we will consider the second order trend. (Although it's not included I found that centered locations produced almost identical models.)
Code:

```
########### Defining first and second order models
> firstOrder <- lm(WolfData ~ lat + lon)
> secondOrder <- lm(WolfData ~ lat + lon +
                    I(lat^2) + I(lon^2) + I(lon*lat))
###### Partial F-test
> anova(firstOrder, secondOrder)
Analysis of Variance Table

Model 1: WolfData ~ lat + lon
Model 2: WolfData ~ lat + lon + I(lat^2) + I(lon^2) + I(lon * lat)
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     82 318200
2     79 256887  3     61313 6.2852 0.000702 ***



########### Consider the significance of each term.
> summary(secondOrder)

Call:
lm(formula = WolfData ~ lat + lon + I(lat^2) + I(lon^2) + I(lon *
    lat))

Residuals:
     Min       1Q   Median       3Q      Max
-124.405  -43.662   -2.337   39.017  199.198

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.203e+02  1.295e+01  47.902  < 2e-16 ***
lat          -1.330e+00  8.861e-02 -15.008  < 2e-16 ***
lon          -1.075e+00  8.191e-02 -13.128  < 2e-16 ***
I(lat^2)     -2.929e-03  1.101e-03  -2.659 0.009486 **
I(lon^2)      8.994e-05  5.908e-04   0.152 0.879388
I(lon * lat)  3.184e-03  8.790e-04   3.622 0.000515 ***


Residual standard error: 57.02 on 79 degrees of freedom
Multiple R-squared:  0.9119,    Adjusted R-squared:  0.9063
F-statistic: 163.6 on 5 and 79 DF,  p-value: < 2.2e-16
```

Given that we know will consider variograms with second order trend we can now construct an empirical semivariogram. We will be using the robust empirical semivariogram estimator, but before we move on to estimating the semivariogram we must consider a few diagnostic concerns. Do we even need a spatial model? Does the data exhibit anisotropy?

We will address the need for constructing a spatial model first. Testing for spatial autocorrelation using a randomization test we can see that the empirical semivariogram for our original data shows a clear trend amongst the randomized empirical semivariograms, we need a spatial model.

Figure 6: Randomization test for spatial autocorrelation. (red is the original data)

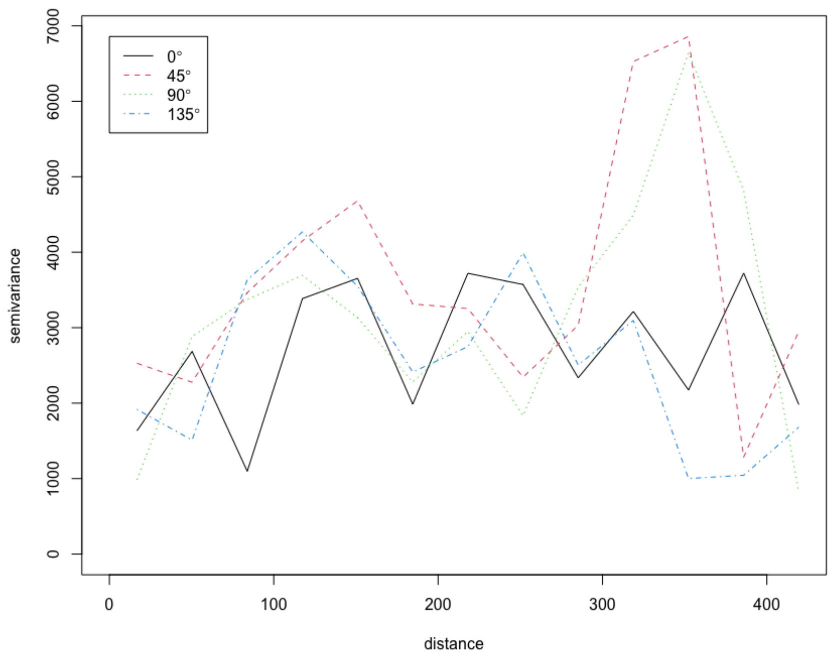


Code:

```
# Generate Second Order Model
secondOrder <- lm(WolfData ~ lat + lon + I(lat^2) + I(lon^2) + I(lon*lat))

# New geodata object with residuals of 2nd order model
WolfcampRes <- as.geodata(cbind(wolfcamp$coords[,1],
                                wolfcamp$coords[,2],
                                secondOrder$residuals))

# Consider empirical semivariogram with original spatial order
orig_vario <- variog(WolfcampRes, trend="cte", estimator.type="modulus" )
lines(orig_vario, col = 'red', lwd=3 )

# Conducting randomization test
n <- length(wolfcamp$data)
for( i in 1:100) {
  # Sample a permutation
  new.order <- sample(1:n, size=n, replace=FALSE )
  # Generate Reordered data
  reordered <- WolfcampRes$data[new.order]
  reordered.geodata <- as.geodata(
    cbind(WolfcampRes$coords, reordered))
  # Generate reordered semivariogram
  reordered_vario <- variog(reordered.geodata,
                           trend="cte", estimator.type="modulus" )
  # Plot reordered semivariograms
  lines(reordered_vario, col="gray")
}
```

To check for anisotropy we will construct 4 directional empirical semivariograms using the variog4() function in geoR. Doing so we find that the variance in each direction seems to stay the same, except for lags in the range of 300 to 350. This makes sense then we look at the spatial trend and distribution in our data, however what is most important is where we have data in the lower range from 0 to 250 we see very similar variances.

Figure 7: Testing for anisotropy with directional semivariograms

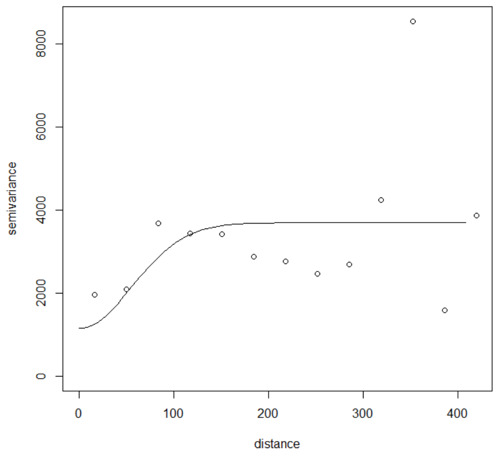


Code:

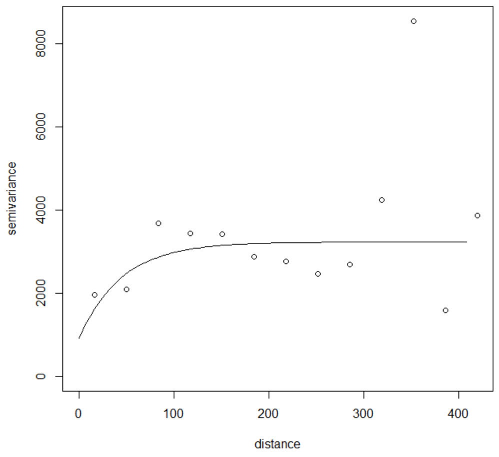
```
directional_vario <- variog4(wolfcamp, trend="2nd",
                              estimator.type="modulus")
plot(directional_vario)
```

Having roughly tested the assumptions of isotropic second order stationarity we can continue by considering what type and technique we are going to use to estimate the variogram. Using eyefit() to find initial values for each type of semivariogram we get the following,

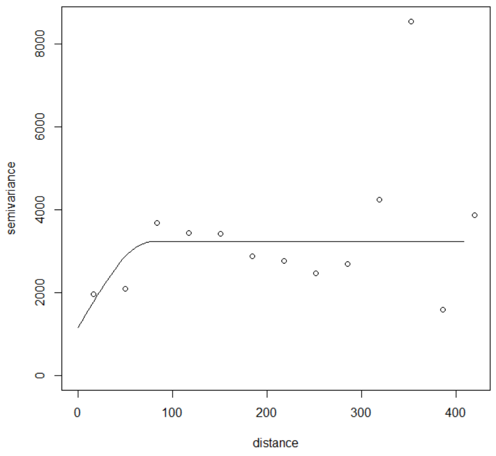
Figure 8: Eyefit() plots.



(a) Gaussian



(b) Exponential



(c) Spherical

Figure 9: Initial values from eyefit()

	σ^2	ϕ	τ^2
Gaussian	2543	79.35	1155.91
Exponential	2311	45.34	924.73
Spherical	2080	79.35	1155.91

Using these initial values we can fit the WLS estimator, then subsequently the ML and

REML estimators. Doing so we get the following.

Figure 10: WLS estimated semivariograms.

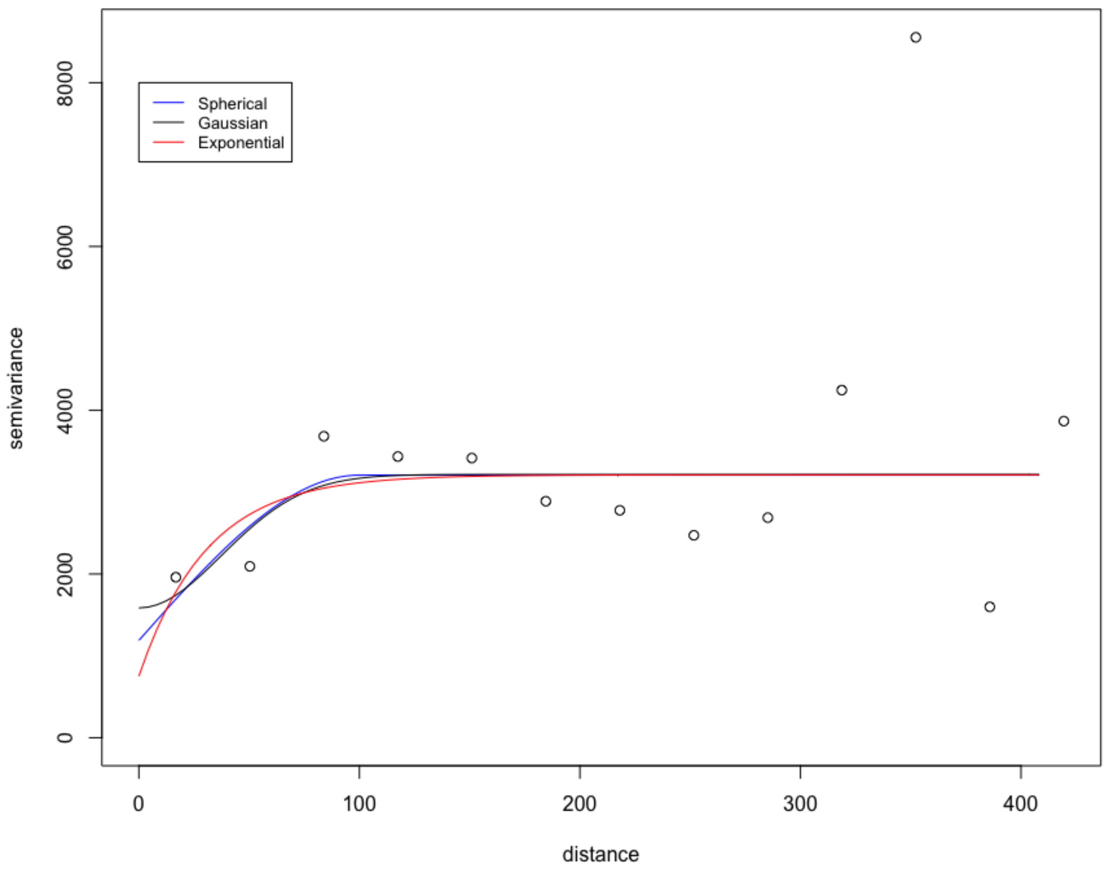


Figure 11: ML estimated semivariograms.

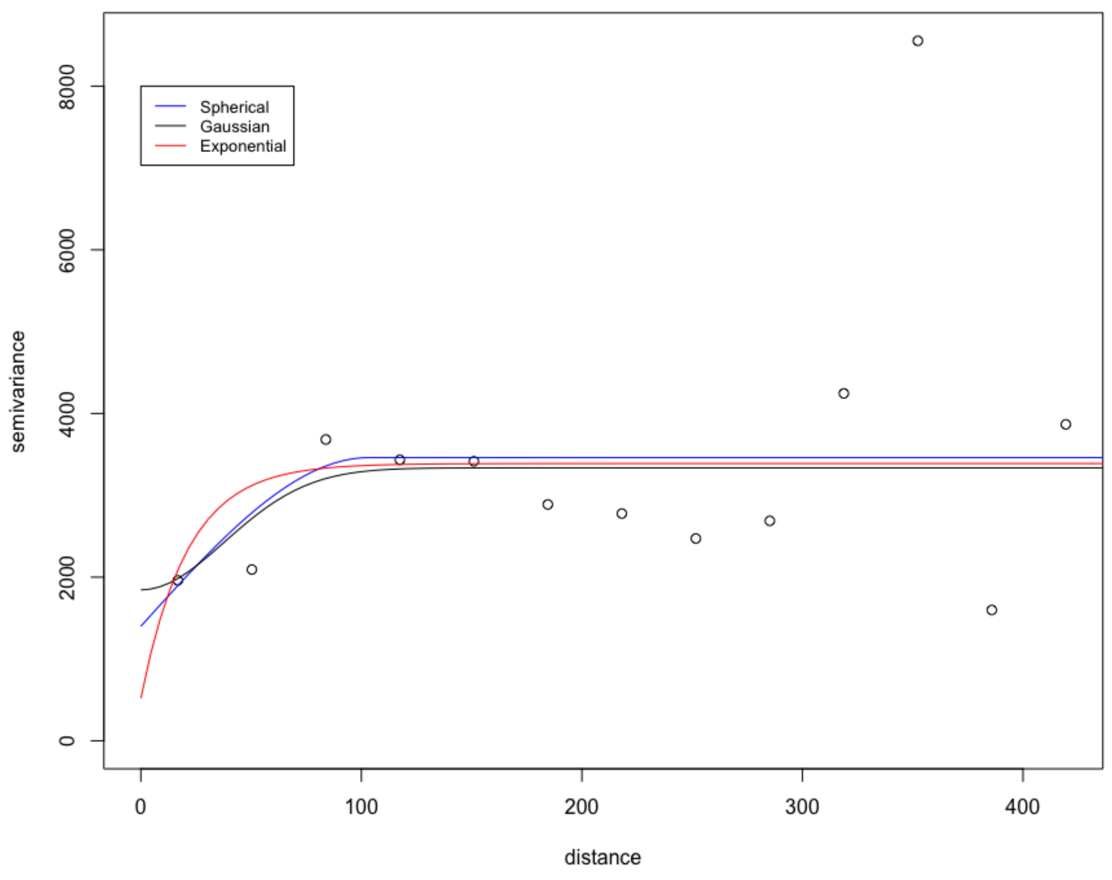
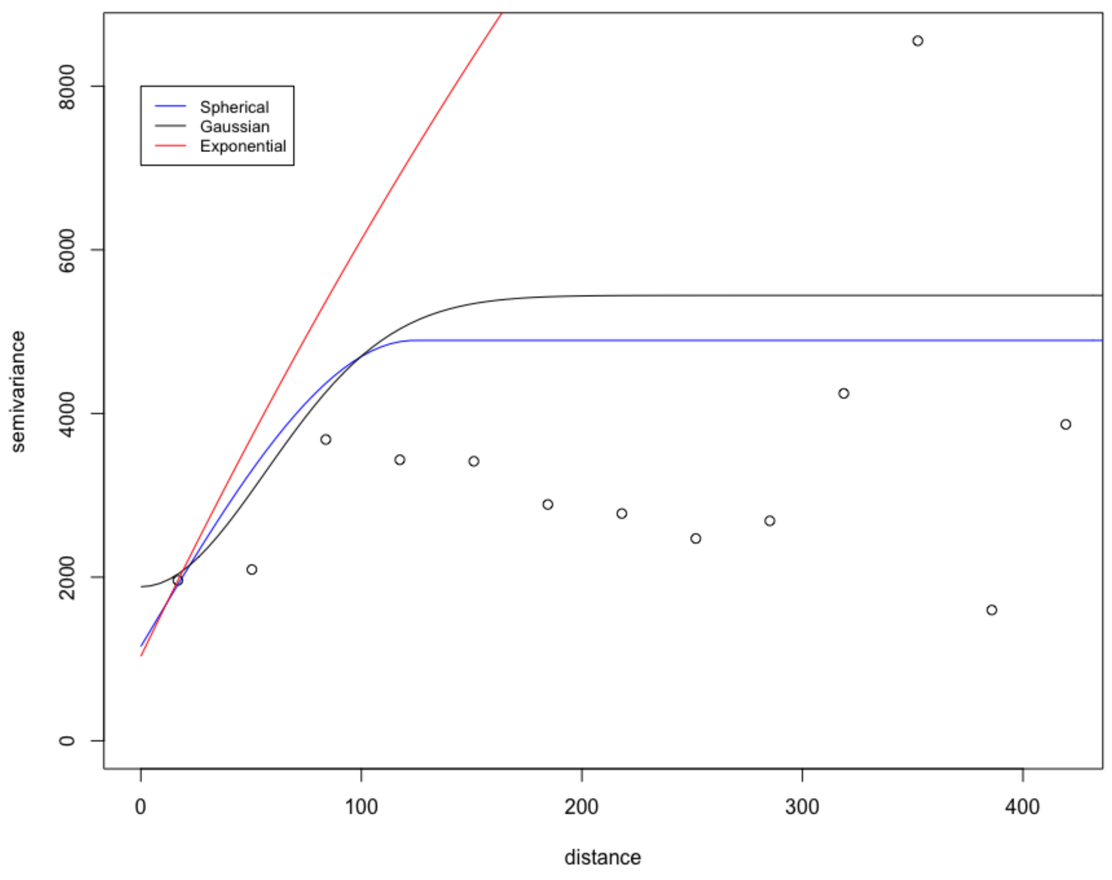


Figure 12: REML estimated semivariograms.



Code:

```
######## Fitting the robust empirical second order trend semivariogram
robust_vario <- variog(wolfcamp, trend="2nd", estimator.type="modulus")


### Fitting the WLS estimator for each type of semivariogram
WLS_exponential <- variofit(robust_vario, cov.model = 'exponential',
                            max.dist = 408,
                            ini.cov.pars = c(2311,45.34),
                            nugget = 924.73)


WLS_gaussian <- variofit(robust_vario, cov.model = 'gaussian',
                         max.dist = 408,
                         ini.cov.pars = c(2311,45.34),
                         nugget = 924.73)
```

```
WLS_spherical <- variofit(robust_vario, cov.model = 'spherical',
                          max.dist = 408,
                          ini.cov.pars = c(2311,45.34),
                          nugget = 924.73)


## Generating plot of WLS estimators
plot(robust_vario)
lines(WLS_spherical, col = 'blue')
lines(WLS_gaussian)
lines(WLS_exponential, col = 'red')
legend(x = 0, y = 8000, legend = c('Spherical', 'Gaussian', 'Exponential'),
       col = c('blue', 'black','red'),
       lty=1, cex=0.8)



## Storing initial values for ML and REML
init_pars_exp <- WLS_exponential$cov.pars
init_pars_gauss <- WLS_gaussian$cov.pars
init_pars_sphere <- WLS_spherical$cov.pars

init_tau_exp <- WLS_exponential$nugget
init_tau_gauss <- WLS_gaussian$nugget
init_tau_sphere <- WLS_spherical$nugget





### Fitting the ML estimator for each type of semivariogram
ML_exponential <- likfit(wolfcamp, cov.model = 'exponential',
                          trend = '2nd',
                          ini.cov.pars = init_pars_exp,
                          nugget = init_tau_exp,
                          lik.method = 'ML',
                          max.dist = 408)



ML_gaussian <- likfit(wolfcamp, cov.model = 'gaussian',
                          trend = '2nd',
                          ini.cov.pars = init_pars_gauss,
                          nugget = init_tau_gauss,
                          lik.method = 'ML',
                      max.dist = 408)



ML_spherical <- likfit(wolfcamp, cov.model = 'spherical',
                          trend = '2nd',
                          ini.cov.pars = init_pars_sphere,
                          nugget = init_tau_sphere,
                          lik.method = 'ML',
                        max.dist = 408)


## Generating plot of ML estimators.
plot(robust_vario)
lines(ML_spherical, col = 'blue')
lines(ML_gaussian)
lines(ML_exponential, col = 'red')
legend(x = 0, y = 8000, legend = c('Spherical', 'Gaussian', 'Exponential'),
       col = c('blue', 'black','red'),
       lty=1, cex=0.8)




## Fitting the REML estimator for each family of semivariograms.
REML_exponential <- likfit(wolfcamp, cov.model = 'exponential',
                          trend = '2nd',
                          ini.cov.pars = init_pars_exp,
                          nugget = init_tau_exp,
                          lik.method = 'REML',
                          max.dist = 408)



REML_gaussian <- likfit(wolfcamp, cov.model = 'gaussian',
                       trend = '2nd',
                       ini.cov.pars = init_pars_gauss,
                       nugget = init_tau_gauss,
                       lik.method = 'REML',
                       max.dist = 408)



REML_spherical <- likfit(wolfcamp, cov.model = 'spherical',
                        trend = '2nd',
                        ini.cov.pars = init_pars_sphere,
                        nugget = init_tau_sphere,
                        lik.method = 'REML',
                        max.dist = 408)


## Generating the plot of each REML estimator.
plot(robust_vario)
lines(REML_spherical, col = 'blue')
lines(REML_gaussian)
lines(REML_exponential, col = 'red')
legend(x = 0, y = 8000, legend = c('Spherical', 'Gaussian', 'Exponential'),
       col = c('blue', 'black','red'),
       lty=1, cex=0.8)
```

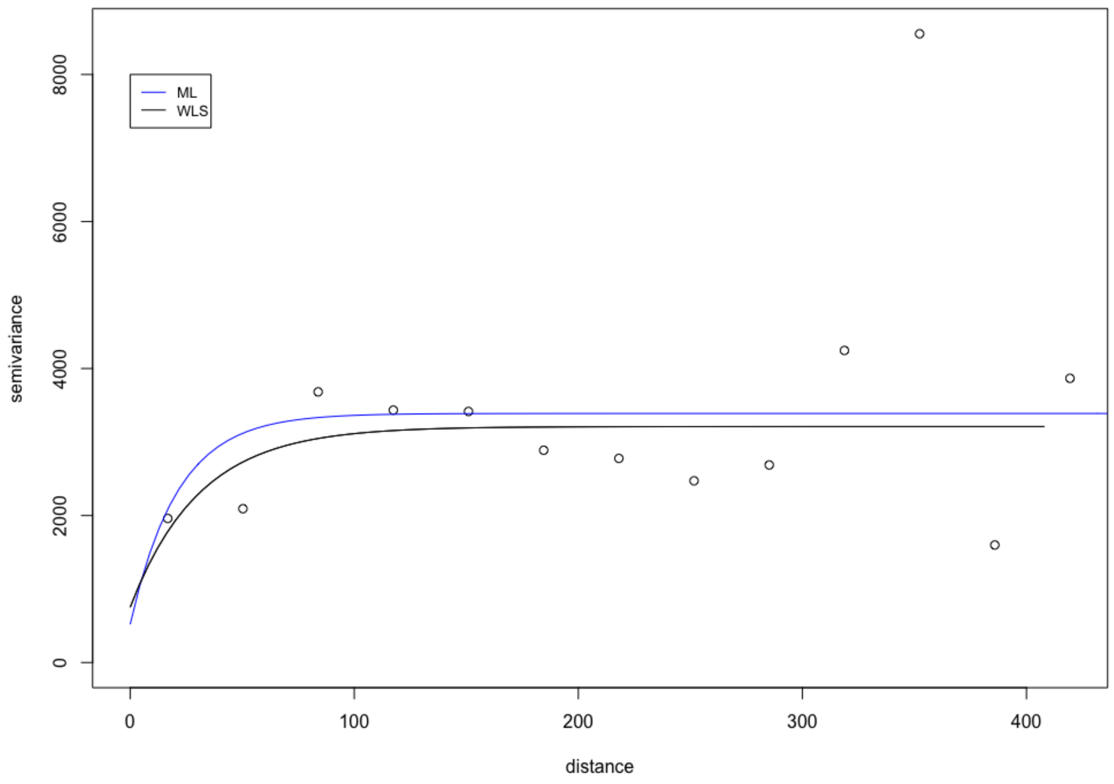
Generally it seems as though the REML estimator does a poor job at estimating the semivariogram. Beyond that it seems that the WLS and ML estimators generally do a good job at estimating the semivariogram. We can see that there is larger variance among the ML estimated variograms. Computing the AIC for each ML estimator we get the following table,

Figure 13: Computed AIC values from ML estimated models.

	AIC
Gaussian	929.7375
Exponential	927.8655
Spherical	928.1551

Here we see that exponential model exhibits the best fit. Comparing the WLS and ML exponential models we see that, the ML model seems to have a larger sill favoring the larger lags in the 300 to 400 region. Moving forward we will be considering the ML exponential model.

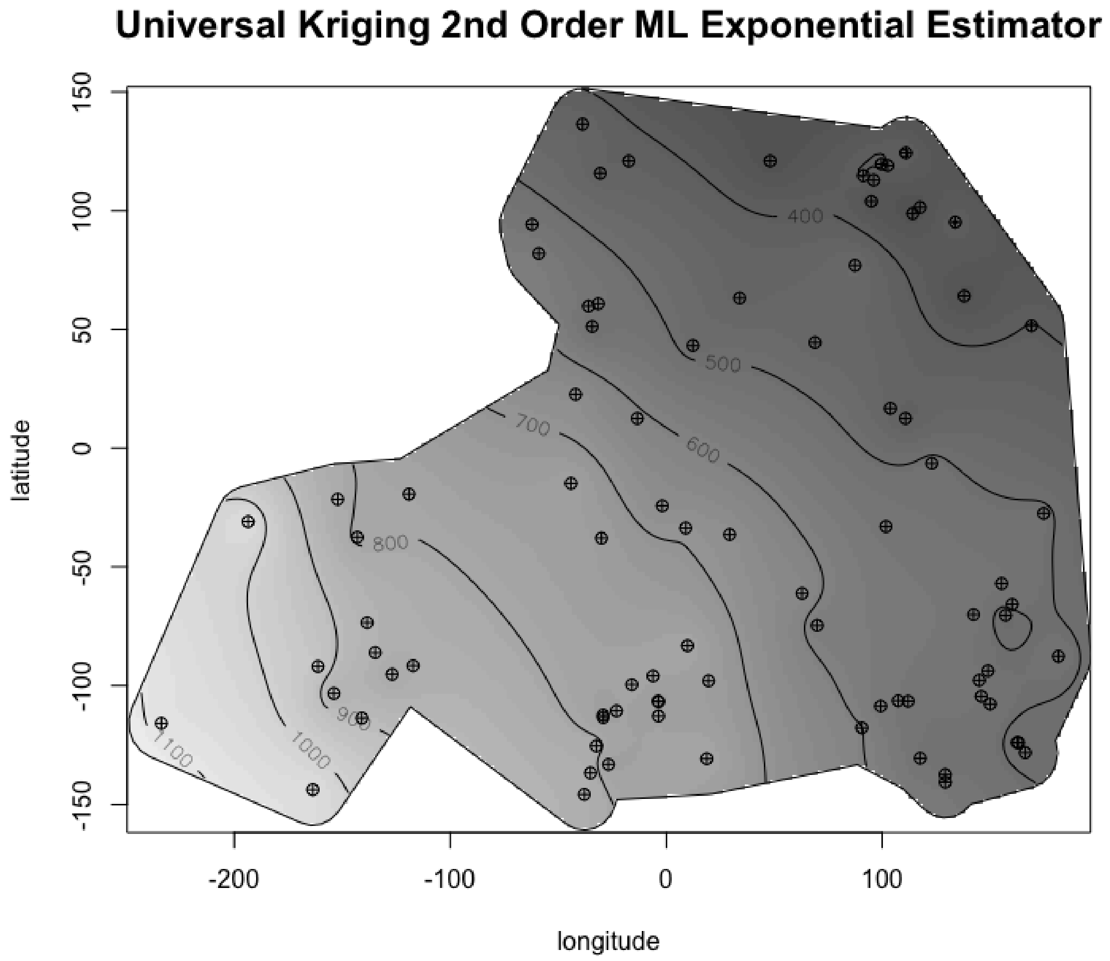
Figure 14: ML vs WLS exponential semivariograms.



Code:

```
## AIC Table
AIC(ML_exponential)
AIC(ML_gaussian)
AIC(ML_spherical)

plot(robust_vario)
lines(ML_exponential, col = 'blue')
lines(WLS_exponential)
legend(x = 0, y = 8000, legend = c('ML', 'WLS'),
       col = c('blue', 'black'),
       lty=1, cex=0.8)

####### Summary Report for ML Exponential Model.
> summary(ML_exponential)
Summary of the parameter estimation
-----------------------------------
Estimation method: maximum likelihood

Parameters of the mean component (trend):
   beta0    beta1    beta2    beta3    beta4    beta5
617.2769  -1.1527  -1.3063   0.0010  -0.0027   0.0024

Parameters of the spatial component:
   correlation function: exponential
      (estimated) variance parameter sigmasq (partial sill) =  2869
      (estimated) cor. fct. parameter phi (range parameter)  =  21.23
   anisotropy parameters:
      (fixed) anisotropy angle = 0  ( 0 degrees )
      (fixed) anisotropy ratio = 1

Parameter of the error component:
      (estimated) nugget =  519.7

Transformation parameter:
      (fixed) Box-Cox parameter = 1 (no transformation)

Practical Range with cor=0.05 for asymptotic range: 63.59205
```

Since we are including a second order trend we will continue by using universal kriging to create a smoothed map of the wolfcamp data. Using the krig.control() and krig.conv() functions we get the following,



Code:

```
## Generating the kriging object.
my_kr_obj <- krige.control(
    type.krige = 'OK',
    trend.d = '2nd',
    trend.l = '2nd',
    cov.model = 'exponential',
    cov.pars = ML_exponential$cov.pars,
    nugget = ML_exponential$nugget,
)

## Pulling the bounding lon, lat pairs.
lonmin <- min(wolfcamp$borders[,1])
lonmax <- max(wolfcamp$borders[,1])
latmin <- min(wolfcamp$borders[,2])
latmax <- max(wolfcamp$borders[,2])

## Resolution was computed by dividing the range of both
## the lat and lon by 100 and picking the smaller one
## for me this generates a ~200*400 pred grid
res = 2

## Generating the prediction grid
my_grid <- pred_grid(
  c(lonmin,lonmax),
  c(latmin,latmax),
  by = res )


## Generating prediction surface
my_kr_results <- krige.conv(wolfcamp,
                            krige = my_kr_obj,
                            locations = my_grid)


## Ploting the prediction surface
my_grays <- gray( seq(25/63, 59/63,length = 40 ))
one_plot(my_kr_results$predict,
         lonmin, lonmax,
         latmin, latmax, res,
         'longitude', 'latitude',
         wolfcamp$border, my_grays,
         wolfcamp$coords,
         'Universal Kriging 2nd Order ML Estimator')
```

Calling my_kr_results$beta.est we get the final estimated regression equation,

$$\mathbb{E}(Y(s)) = 617.3 - 1.2lon - 1.3lat + .00098lon^2 - .0027lat^2 + .0024lonlat.$$

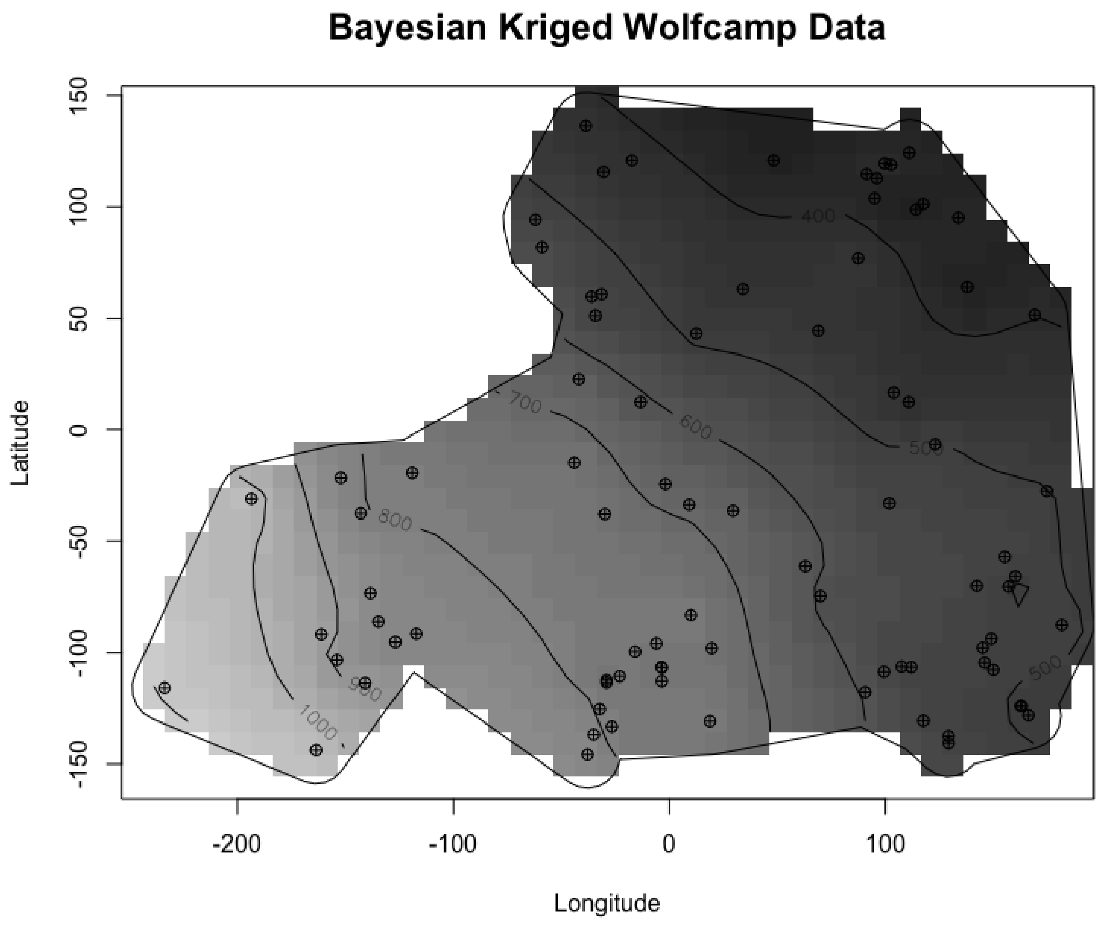
This final fitted model is kind of similar to our non spatial second order model,

$$\mathbb{E}(Y(s)) = 620.3 - 1.0lon - 1.3lat + .00089lon^2 - .0029lat^2 + .0032lonlat.$$

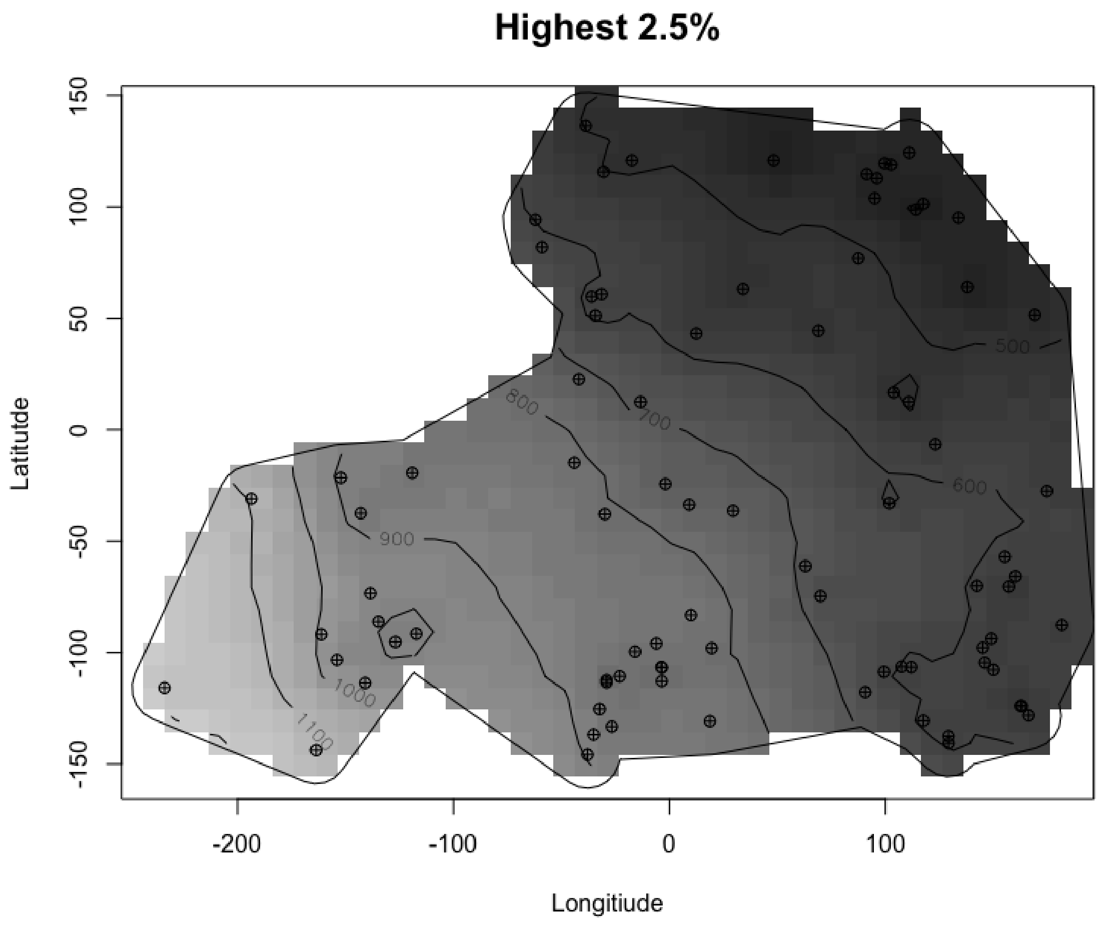
They both exhibit the same significance in the lat^2 and interaction terms.

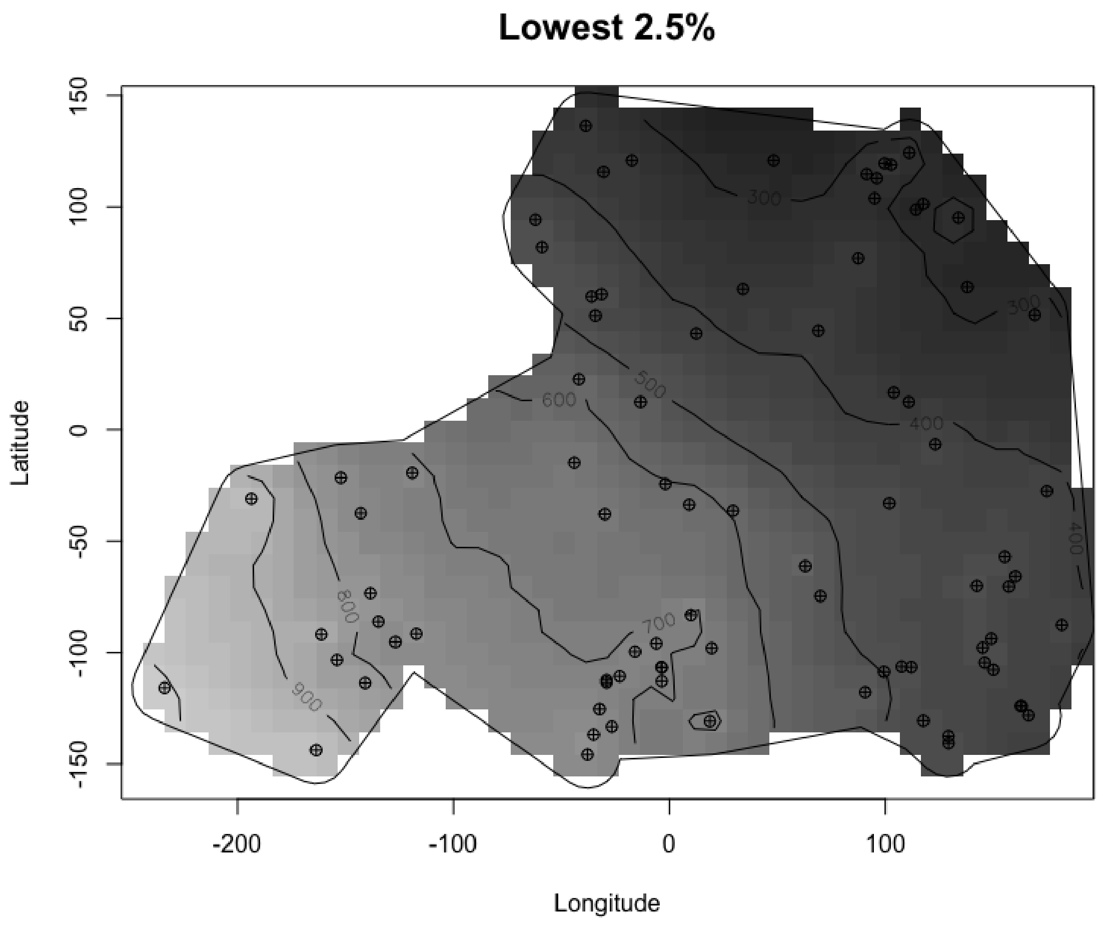
Exercise 3: Fit a Bayesian model to these data by modifying the R code, Bayesian_kriging.r, which is posted on Canvas. Be sure to include diagnostic plots shown in class (traceplots and density plots for the model parameters) and a final smoothed map.

Solution:
Using the Bayesian_kriging.r R code to fit a Bayesian model, we get the following smoothed map(A lower resolution was used to save computational time),

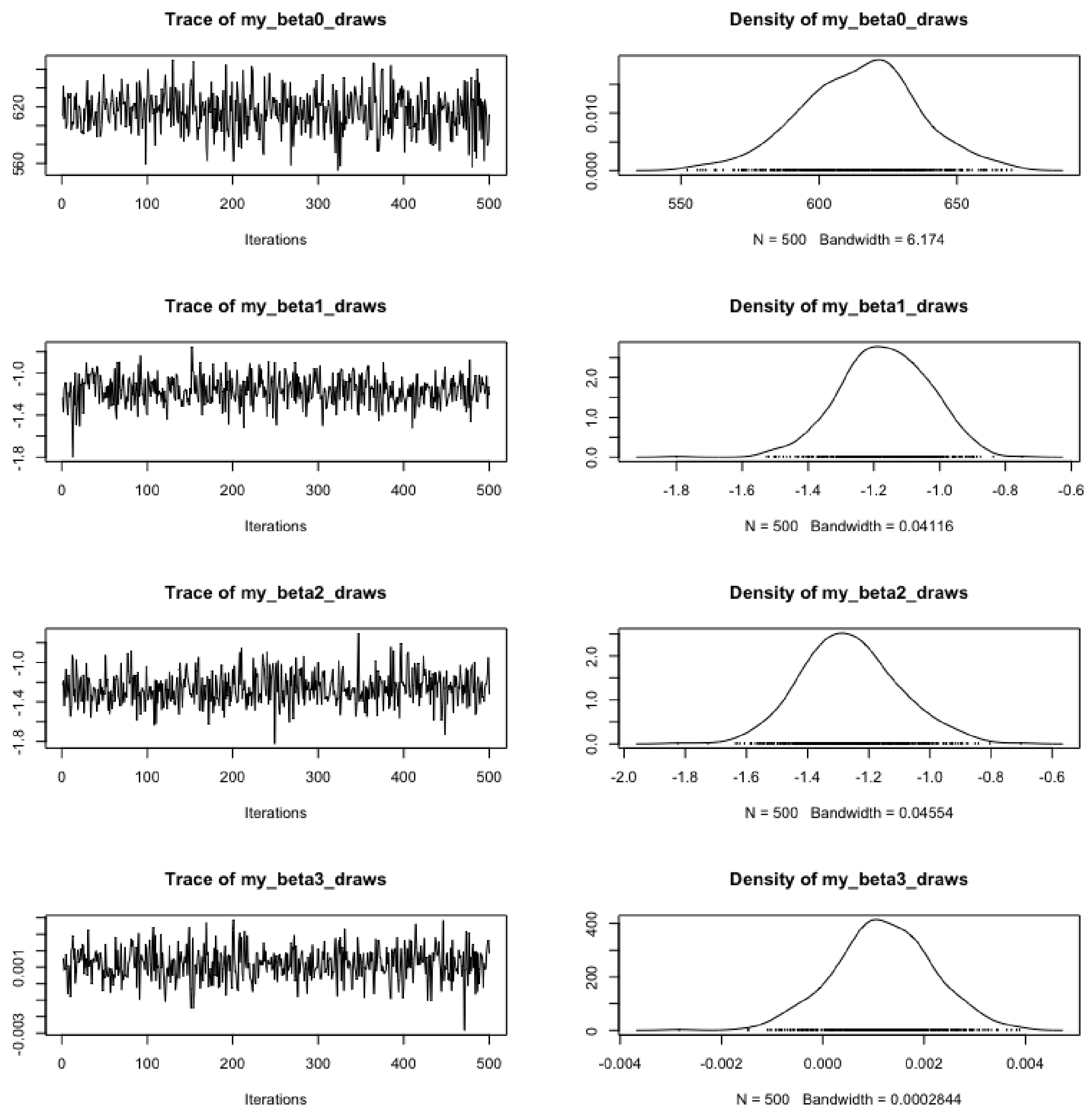


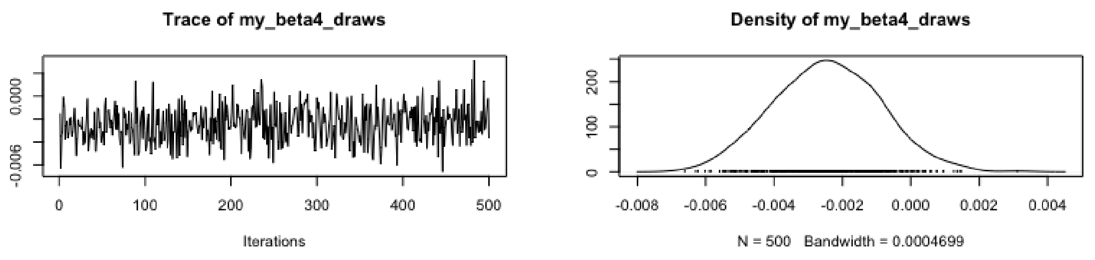
The Bayesian model produced the following 95% credible set, smoothed maps

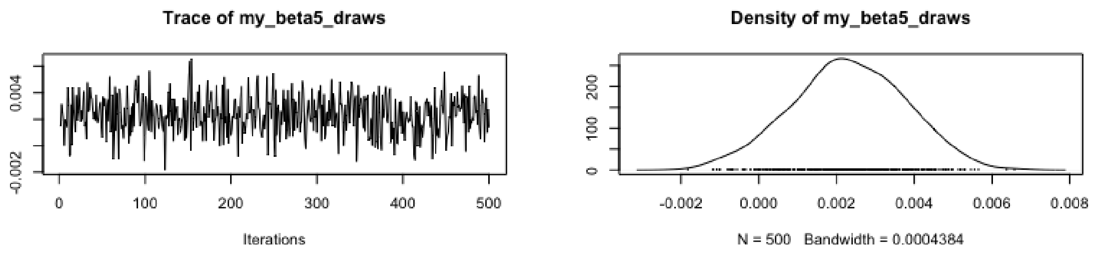

Highest 2.5%
Latitude
Longitiude
150
100
50
0
-50
-100
-150
-200
-100
0
100
500
600
700
800
900
1000
1100

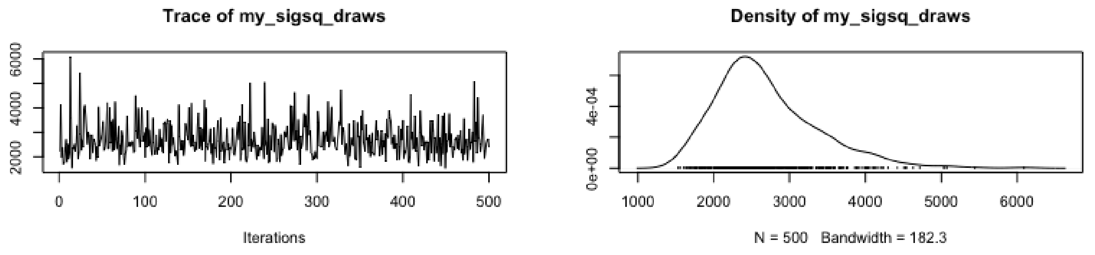


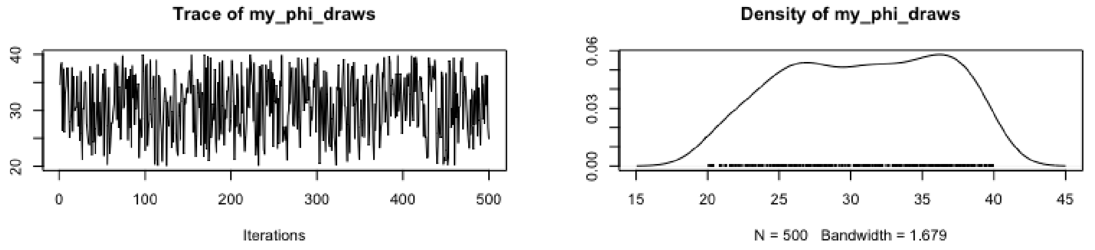
The Bayesian model also produced the following trace plots for each parameter from the 5000 posterior distribution samples (Draws were plotted at interval of every 10 draws, hence 500 iterations). Looking at traceplots for the β_i parameters, show relatively good convergence. Of the 6 parameters, it seems as though β_0 has the highest variance, yet the density looks very normal this is likely do to the fact that our location data are not centered or scaled(the plot looks to have more variance and poor convergence). We also see β_1 exhibit some level of burn-in, around iteration 250 we see the the parameter level off. The σ^2 trace plot looks normal given that it is a non-negative parameter. The variance exhibited in the ϕ trace plot tells us that our data is not saying much about the value of the range. This conclusion makes sense given the shape of our empirical semivariogram, it seems as though since it is very flat, in large part, what is going to determine the shape of the estimated semivariogram is the sill and the nugget and range parameters from 0 to 100 will still fit fairly well.

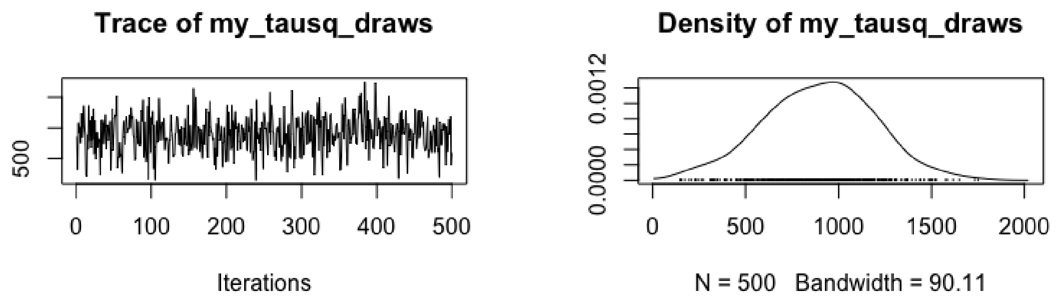

Trace of my_beta0_draws
Density of my_beta0_draws
N = 500 Bandwidth = 6.174
Trace of my_beta1_draws
Density of my_beta1_draws
N = 500 Bandwidth = 0.04116
Trace of my_beta2_draws
Density of my_beta2_draws
N = 500 Bandwidth = 0.04554
Trace of my_beta3_draws
Density of my_beta3_draws
N = 500 Bandwidth = 0.0002844
Iterations


Trace of my_beta4_draws
Iterations
Density of my_beta4_draws
N = 500 Bandwidth = 0.0004699


Trace of my_beta5_draws
Iterations
Density of my_beta5_draws
N = 500 Bandwidth = 0.0004384


Trace of my_sigsq_draws
Iterations
Density of my_sigsq_draws
N = 500 Bandwidth = 182.3


Trace of my_phi_draws
Iterations
Density of my_phi_draws
N = 500 Bandwidth = 1.679


Trace of my_tausq_draws
Iterations
Density of my_tausq_draws
N = 500 Bandwidth = 90.11

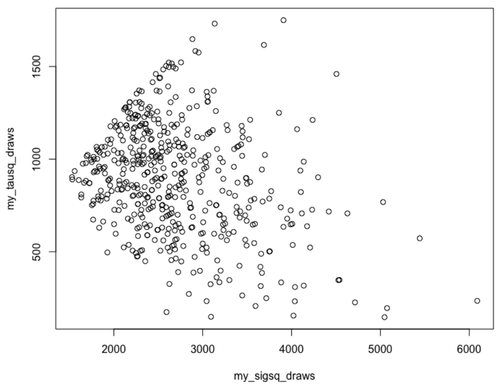
Code:

```
myGeodata <- wolfcamp
myName <- "wolfcamp"

## Specifying the prior distributions (Discussed in question 4)
my_prior_control <- prior.control(
  beta.prior = "normal",
  beta = c(600, 0.0, 0.0, 0.0, 0.0, 0.0),
  beta.var.std = 10000*diag(6),

  sigmasq.prior="reciprocal",
  phi.discrete = seq(20, 40, length=600),
  tausq.rel.prior="uniform",
  tausq.rel.discrete=seq(from=.01,to=.6,length=63))

## Specifying the type and trend of the variogram
my_model_control <- model.control(trend.d = "2nd",
    trend.l = "2nd",
    cov.model = "exponential")

## Specifying the number of samples taken from the posterior
## and predictive dists.
my_output_control <- output.control( n.posterior=5000,
  n.predictive = 1000 )

## Specifying prediction grid
lonmin <- min(wolfcamp$borders[,1])
lonmax <- max(wolfcamp$borders[,1])
latmin <- min(wolfcamp$borders[,2])
latmax <- max(wolfcamp$borders[,2])

boxwid = 10
my_grid <- pred_grid(
  c(lonmin,lonmax),
  c(latmin,latmax), by=boxwid )


## Carrying the MCMC algorithm.
my_bayes_results <- krige.bayes(wolfcamp,
  locations = my_grid,
  model = my_model_control,
  prior = my_prior_control,
  output = my_output_control)



## Plotting Kriged smoothed surface.
one_plot( my_bayes_results$predictive$mean,
          lonmin,lonmax, latmin,latmax, boxwid,
          "Longitude",'Latitude',
          myGeodata$borders,my_colors, myGeodata$coords,
          "Bayesian Kriged Wolfcamp Data" )


## Plotting Trace Plots for each Parameter.
```

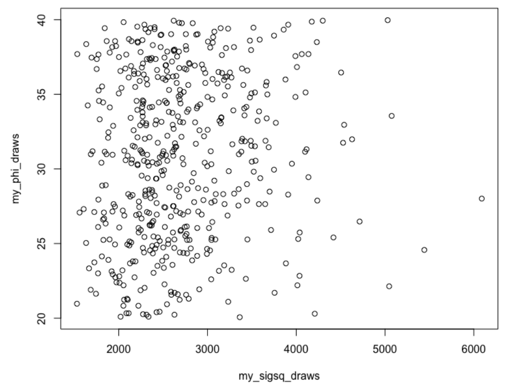
```
plot(mcmc( cbind(my_beta0_draws,
                 my_beta1_draws,
                 my_beta2_draws,
                 my_beta3_draws)))

plot(mcmc( cbind(my_beta4_draws,
                 my_beta5_draws,
                 my_sigsq_draws,
                 my_phi_draws)))

plot(mcmc( cbind(my_tausq_draws)))



##############################################
# credible_sets
if(TRUE) {
  nro <- nrow(my_bayes_results$predictive$simulations)
  lows  <- rep( NA, nro )
  highs <- rep( NA, nro )
  for( i in 1:nro ) { # extract one row (i.e. one
    # prediction location, and find percentiles:
    this_site <- my_bayes_results$predictive$simulations[i,]
    ordered <- sort(this_site)
    lows[i]  <- quantile( ordered, 0.025 )
    highs[i] <- quantile( ordered, 0.975 )
  }

  one_plot( highs,
          lonmin,lonmax, latmin,latmax, boxwid,
          "x coord","y coord",
          myGeodata$borders,my_colors, myGeodata$coords,
          "highest 2.5%" )

  one_plot( lows,
            lonmin,lonmax, latmin,latmax, boxwid,
            "x coord","y coord",
            myGeodata$borders,my_colors, myGeodata$coords,
            "lowest 2.5%" )

  diffs <- highs-lows
  one_plot( diffs,
            lonmin,lonmax, latmin,latmax, boxwid,
            "x coord","y coord",
            myGeodata$borders,my_colors, myGeodata$coords,
            "diffs" )


}
```

Considering the correlation between our semivariogram parameters we get the following plots.

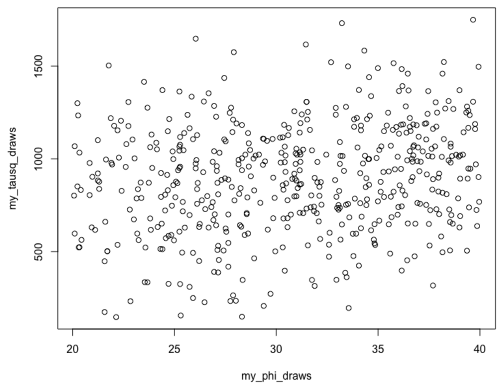
Figure 15: Scatterplots for Semivariogram parameters.



(a) σ^2 vs τ^2



(b) σ^2 vs ϕ



(c) ϕ vs τ^2

As we can see the plots show relatively little correlation, with the most significant computed correlation being -.29 between σ^2 and τ^2.

Code:

```
## Correlation Analysis.
plot( my_sigsq_draws , my_phi_draws )

plot( my_sigsq_draws , my_tausq_draws )

plot( my_phi_draws , my_tausq_draws )

cor( my_sigsq_draws , my_phi_draws ) # 0.1042849
cor( my_sigsq_draws , my_tausq_draws ) # −0.2915101
cor( my_phi_draws , my_tausq_draws ) # 0.1908251
```

Exercise 4: State the statistical model you are using. This means: state the likelihood and state the prior distribution for the various parameters in the model. Page 118 of the lecture notes shows an example of what I mean by this. Also, be sure to state the posterior distribution, for example, as show on page 119 of the class notes.

Solution:
Stating our likelihood and priors similarly to page 118 we get the following. Note that $\hat{\beta} = [\beta_0, ...\beta_5]$, n is the number of observations in the wolfcamp datam X is our second order design matrix, and Σ is our covariance matrix (I think we should be replacing it with S^2).

$$\begin{aligned} Y &\sim MVN_n\left(X\hat{\beta}, \Sigma\right) \\ \hat{\beta} &\sim MVN_6\left([600, 0, 0, 0, 0, 0], 10000I_6\right) \\ \sigma^2 &\sim 1/\sigma^2(\sigma^2 > 0) \\ \phi &\sim uniform[20 \dots 60] \\ \omega &\sim uniform[0.01 \dots 0.6] \end{aligned}$$

Stating the posterior distribution we get the following. Let $\theta = (\hat{\beta}, \sigma^2, \phi, \omega)$.

$$\begin{aligned} p(\theta|y) &\propto L(\theta)\pi(\hat{\beta})\pi(\sigma^2)\pi(\phi)\pi(\omega) \\ &\propto \frac{1}{\sqrt{(2\pi)^n}\det(\Sigma)} \exp(-\frac{1}{2}(y - X\hat{\beta})^T\Sigma^{-1}(y - X\hat{\beta})) \\ &\times \frac{1}{\sqrt{2\pi(10^4)}} \exp\left(-\frac{1}{2(10^4)}(\beta_0 - 600)^2\right) \times \prod_{i=1}^{5} \frac{1}{\sqrt{2\pi(10^4)}} \exp\left(-\frac{1}{2(10^4)}\beta_i^2\right) \\ &\times \frac{1}{\sigma^2}(\sigma^2 > 0) \\ &\times I(\phi \in [20 \dots 60]) \\ &\times I(\omega \in [0.01 \dots .6]) \end{aligned}$$

Exercise 5: Summarize the (marginal) posterior distributions of the model parameters in a table; one row for each paramater and the following columns for each:

1. posterior mean
2. posterior standard deviation
3. posterior median

4. 95% credible set
5. Monte Carlo error
6. Effective sample size.

Solution:
By calling summary on the mcmc(my_bayes_results$posterior$sample) we get a fairly thorough summary report. Note that this command is creating an mcmc object on all 5000 samples from the posterior distribution therefore it includes no thinning, and any burn-in samples. From our trace plots we saw that really only β_1 exhibited any burn in, so for the following table we used the 'start = ' paramater to remove those samples.

Figure 16: Marginal Posterior Distributions Summary

	μ	σ	median	95% CS	MC err.	Effec.samp.size
β_0	614.4	22.10	614.6	(570.3, 658.2)	.3125	5000
β_1	-1.16	.133	-1.16	(-1.42, -.896)	.00188	5000
β_2	-1.27	.157	-1.267	(-1.57, -.940)	.00222	5000
β_3	.00119	.000965	.00116	(-.00068, .000317)	.0000136	5000
β_4	-.00248	.00164	-.00249	(-.00567, .000754)	.0000232	5000
β_5	.00230	.00148	.002308	(-.000598, .00523)	.0000209	4883
σ^2	2687	690.3	2559	(1728, 4410)	9.76	5000
ϕ	31.1	5.490	31.45	(20.90, 39.63)	.0776	5000
ω	.362	.149	.372	(.0764, .591)	.00211	5000

Code:

```
> summary(mcmc(my_bayes_results$posterior$sample))
> summary(mcmc(my_bayes_results$posterior$sample, start = 100))
> effectiveSize(mcmc(my_bayes_results$posterior$sample))
> effectiveSize(mcmc(my_bayes_results$posterior$sample, start = 100))
```

Exercise 6: Provide a summary of your results; specifically,

1. How many iterations did you use for your MCMC, and how much thinning (if any) did you do? What about burn-in?

Solution:
Our MCMC algorithm utilized 5000 iterations to sample the posterior distribution and 1000 iterations of kriged surfaces sampled from the predictive posterior distribution. We used some thinning to visualize the trace plots as pointed out previously, and we also experience some burn-in on the β_1 parameter. When computing the summary report for the marginal posterior distributions we did not use any thinning, and we computed the values for the β_1 distribution excluding the burn-in samples (although it didn't change really).

2. Discuss the convergence of the MCMC for the Bayesian model. Do the various plots indicate any issues with convergence? What do the Monte Carlo errors suggest about convergence or lack of convergence?

Solution:
Generally the trace plots do not exhibit any issues with convergence. Notably the traceplot for the ϕ parameter seems to exhibit a wide range of values, which is more indicative of behavior/quality of our data rather than any non-convergence that might appear from an ill-calibrated priori. The β_1 traceplot showes signs of burn-in however quickly levels off, indicateing good convergence. Using the crude 10% test that we discussed during lecture, we find that β_3 could likely be ran a little longer since, $.0000119 < .0000136$, however the difference seems really small and the trace plot really shows no sign of poor convergence.

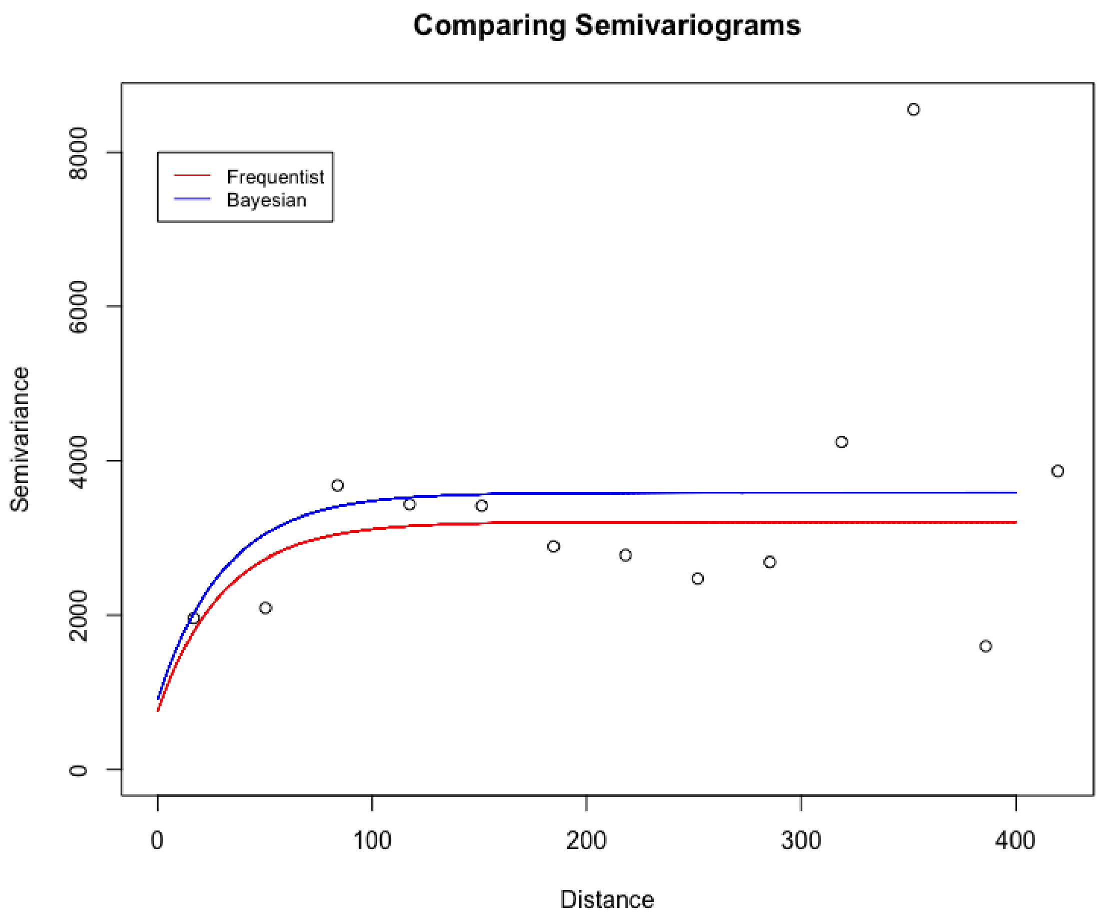
Exercise 7: Discuss (briefly) whether your results for the frequentist and Bayesian approaches are similar; are there any major differences? (A table plus discussion is the best way to do this.)

Solution:
Considering the model parameters for both approaches,

Figure 17: Model Parameters Frequentist vs Bayesian

	β_0	β_1	β_2	β_3	β_4	β_5	σ^2	ϕ	τ^2
Frequentist	617.1	-1.15	-1.31	.0010	-.0027	.0024	2458	30.9	752.123
Bayesian	614.4	-1.16	-1.27	.0012	-.0025	.0023	2687	31.1	899.415

We see the biggest differences in the variogram parameters of our model. Its clear that the Bayesian model has a significantly higher nugget and larger range. Graphing the two semivariograms we get the following,



Code:

```
plot(robust_vario , main = 'Comparing Semivariograms',
     ylab = 'Semivariance', xlab = 'Distance')

Freq_param <- c(2458, 30.9, 752.123)
Bays_param <- c(2687, 31.1, 899.415)
x <- seq(0, 400, .01);

y_freq <- Freq_param[3] + Freq_param[1]*(1 - exp(-x/Freq_param[2]))
y_bays <- Bays_param[3] + Bays_param[1]*(1 - exp(-x/Bays_param[2]))
lines(x, y_freq, col = 'red')
lines(x, y_bays, col = 'blue')

legend(x = 0, y = 8000, legend = c('Frequentist', 'Bayesian'),
       col = c('red', 'blue'),lty=1, cex=0.8)
```

It seems like the Bayesian model puts a larger emphasis on the larger greater lags, opting for a larger nugget and larger range. This should result in smoother realizations outside of our sample areas, as discussed the problem 1 of the previous homework(comparing the matern semivariograms). To me the results are not significancy better from the bayesian method, to warrant such significant increase in computation time. The drawback of having to limit the resolution of the smoothed map is frustrating, when the model, at least to me does not appear to be significantly better.