Exercise 1: Use R to plot two Matern semi-variograms on the same graph, one with smoothness parameter $\nu = k = 1/2$, another with $\nu = k = 5$; in each case use range parameter $a = \phi = 1$. Use a nugget $\tau^2 = 1.0$ and a partial sill of $\sigma^2 = 1.5$. Which of the two semi-variograms corresponds to smoother realizations, and how can you tell?

Solution:
Firstly we must convert the values from the matern function which calculates a co-variogram into our desired semi-variogram. Recall the the relationship between the two variograms,

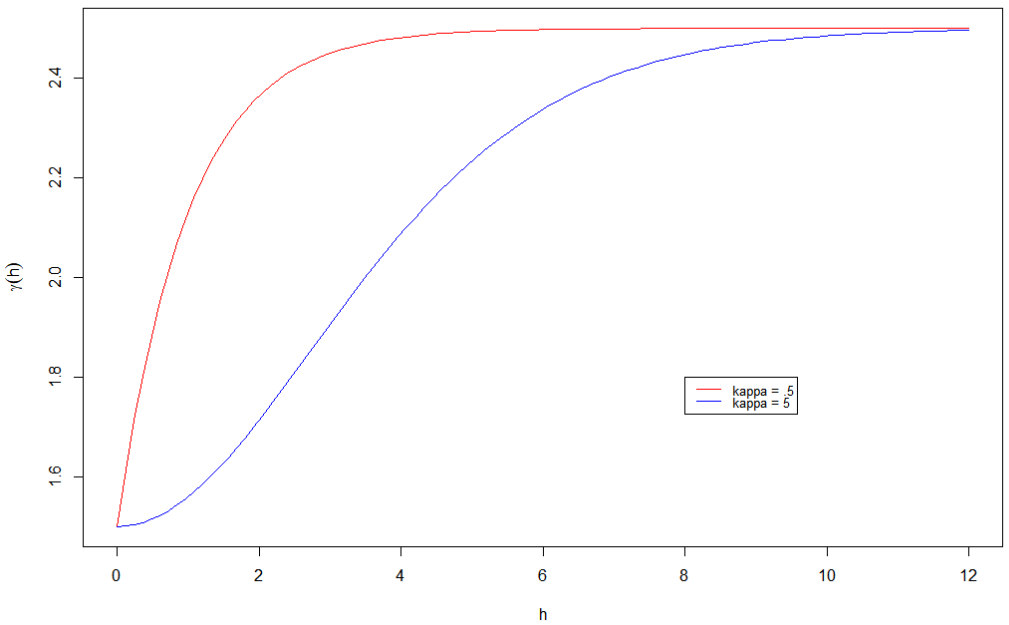
$$\gamma(h) = C(0) - C(h),$$

where we think of $C(0)$ as the sill. Since we want a nugget of $\tau^2 = 1.0$ and a partial sill of $\sigma^2 = 1.5$, the function we will use to plot the semivariograms looks like this,

$$curve((\sigma^2 - \text{matern}(x, \phi, \kappa)) + \tau^2, \text{from} = 0, \text{to} = \text{upperbound}).$$

Plotting the two Matern semi-variograms we get the following,
Code:

```
curve(1.5 - (matern(x, phi = 1, kappa = .5)) + 1, from = 0, to = 12,
            col = 'red' ,ylab = expression(gamma(h)), xlab = expression(h))

curve(1.5 - (matern(x, phi = 1, kappa = 5)) + 1,
            add = TRUE, col = 'blue')

legend(8, 1.8, legend = c('kappa = .5', 'kappa = 5'),
            col = c('red', 'blue'), lty=1, cex=0.8)
```



The goal of any covariogram is to quantify the similarity (or difference) between observations as a function of distance. Comparing the two Matern semi-variograms we find that the one with $\kappa = .5$ has a smaller range than the one with $\kappa = 5$. Interpreting this we see that the rate at which observations become significantly different as a function of time is greater when $\kappa = .5$. Naturally this leads to a more uneven, discrete looking realization.

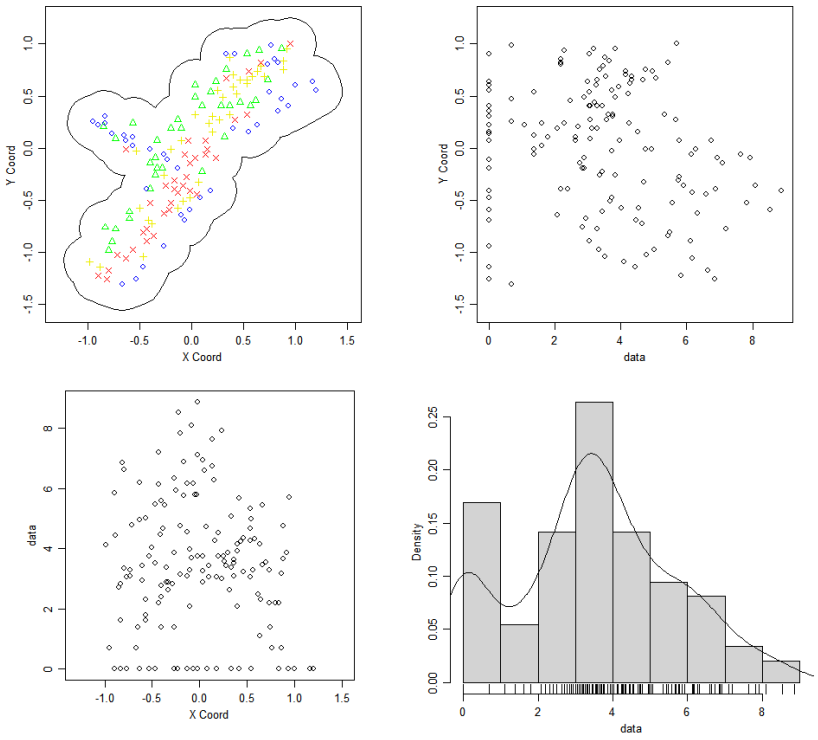
Exercise 2: Continuing analysis using R fo the scallop data set using centered lats longs.

a Add borders to the data set that has centered latitudes and longitudes.

Solution:
Reading in the data and adding borders using my own add borders function() we get the following,
Code:

```
scallops <- read.table("scallops.txt", header=TRUE)

scallops$logcatch <- log( 1 + scallops$tcatch )
scallops$clons <- scallops$long - mean(scallops$long)
scallops$clats <- scallops$lat - mean(scallops$lat)

geo.scallops <- as.geodata(cbind(scallops$clons,
                                 scallops$clats,
                                 scallops$logcatch))

geo.scallops$borders <- get.borders(cbind(scallops[,10],scallops[,11]),
                                    concavity_param = 1, frac = .25)

plot(geo.scallops)
```

Figure 1: Plot of scallops geoR object with new borders.



b Use the pred_grid function to create a suitable grid of prediction locations (using range of values for the centered latitudes and longitudes).

Solution:
We get the range over the centered latitudes and longitudes, then divide it by 100 to fined the necessary spacing, for at least a 100 x 100 resolution.
Code:

```
# > range(scallops$clons)
#      [1] -0.9825453  1.2007847
#
# > range(scallops$clats)
#      [1] -1.313176  1.003494

## Add .2 to either side of the interval
## Then compute the range and divide by 100
abs(-1.182545-1.400785)/100
# .0258
abs(-1.513176 - 1.203494)/100
# .0271

# Setting the step size to .02 seems good and ensured at least 100X100
```

```
my_grid <- pred_grid(c(-1.182545, 1.400785), c(-1.513176, 1.203494 ), by = .02)

lons_p <- my_grid[,1]
lats_p <- my_grid[,2]
```

c Use the krige.control function followed by krige.conv to carry out universal kriging for log(catch) at the prediction locations in (b). The R code on pages 96 and 97 may be helpful, as they show you how to handle non-constant trend. You'll probably just need trend.d = '2nd', trend.1 = 'cte' rather than the additional codes in the notes.

Solution:
Recalling the previous analysis on the scallop data where we computed several estimators for the variogram. I recall that in my analysis a 2nd order gaussian model seemed to fit the best, with the WLS and ML estimators being relatively similar. For this problem I will use the ML estimator, with $\sigma^2 = 3.60472588$, $\phi = .13487626$, and $\tau^2 = .03058721$. Consider the following code,
Code:

```
my_kr_obj <- krige.control(
  type.krige="OK", # Universal kriging
  trend.d = "2nd",
  trend.l = "2nd",
  cov.model="gaussian",
  cov.pars=c(3.60472588,.13487626),
  nugget=.03058721 )

my_kr_results <- krige.conv( geo.scallops,
                             krige=my_kr_obj,
                             locations= cbind(lons_p, lats_p))
```

d State the estimated regression equation, $\mathbb{E}(Y(s)) =$?

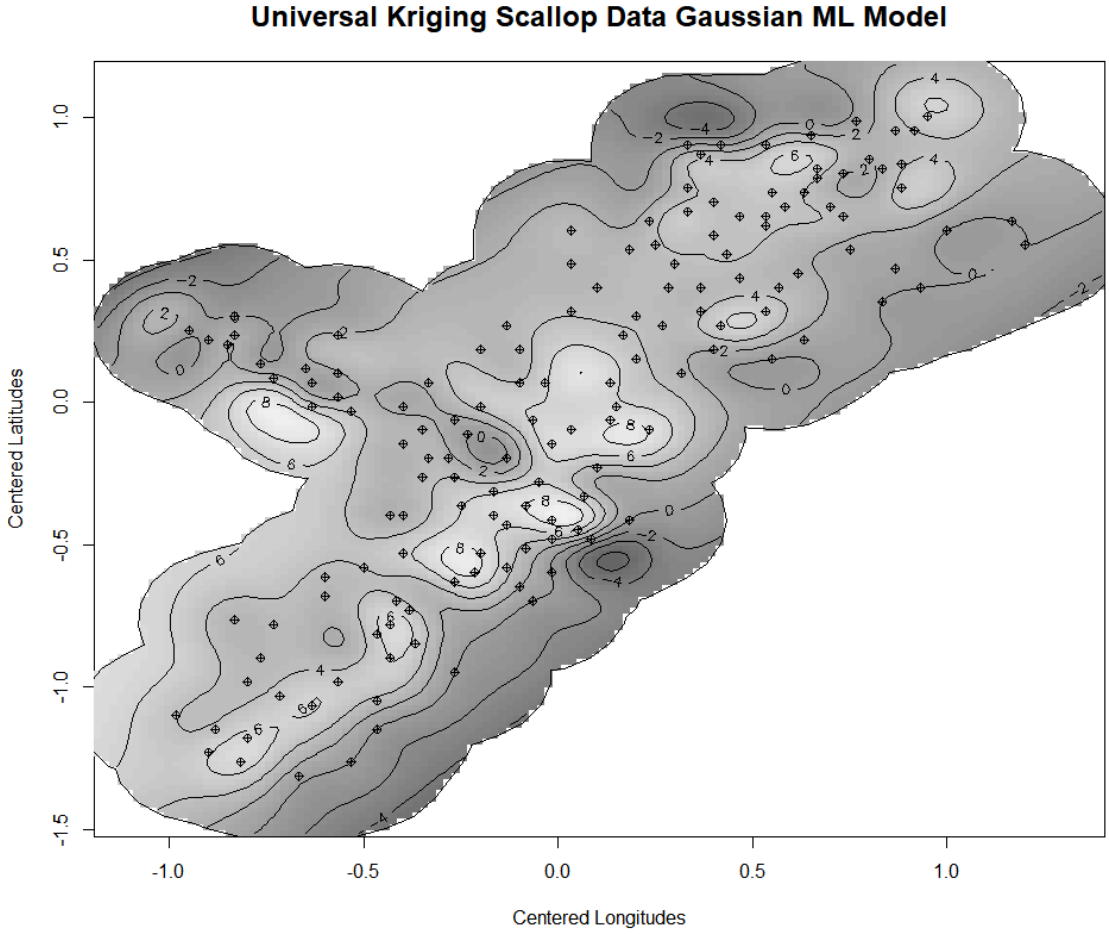
Solution:
We can pull the regression coefficients from the my_kr_results object. Doing so we get the following,

$$\mathbb{E}(Y(s)) = 4.161+(-2.729)lons+(0.727)lats+(-4.406)lons^2+(-7.153)lats^2+(11.765)lonslats$$

e Plot the resulting smoothed map using the centered coordinates.

Solution:
Using the given one_plot() function we get the following plot.
Code:

```
my_grays <- gray( seq(25/63, 59/63, length=40))
one_plot(my_kr_results$predict,
         -1.182545 , 1.400785,
         -1.513176, 1.203494, .02,
         'Centered Longitudes', 'Centered Latitudes',
         geo.scallops$borders, my_grays,
         geo.scallops$coords,
         'Universal Kriging Scallop Data Gaussian ML Model')
```

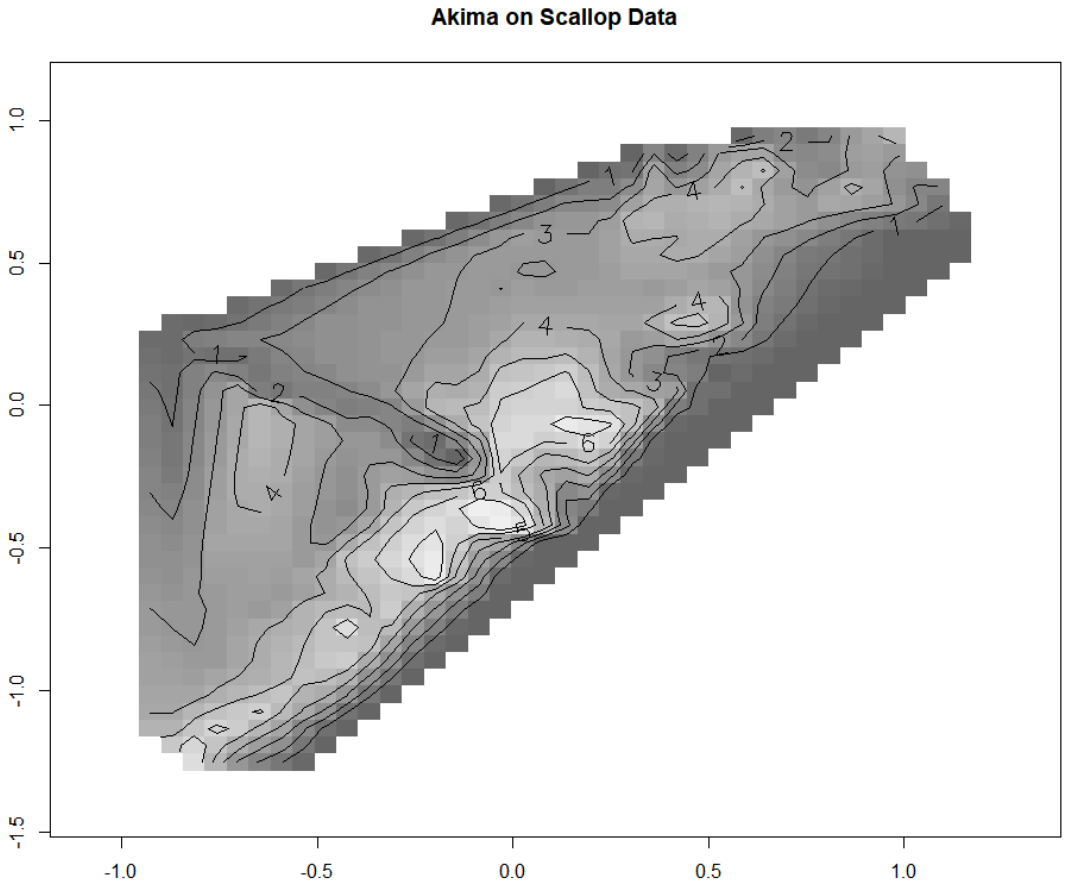


f Create a linear interpolation plot of log(catch) using the interp() function in the akima package.

Solution:

Code:

```
lons <- geo.scallops$coords[,1]
lats <- geo.scallops$coords[,2]

bar <- interp(lons, lats, geo.scallops$data, linear = TRUE)

image(bar, xlim =c( -1.182545 , 1.400785), ylim = c(-1.513176, 1.203494),
      col = my_grays, main = 'Akima on Scallop Data' )

contour(bar, labcex =1.5, add = TRUE)
```



g Comment briefly on your result in (e) and (f). (Are the plots the same? Are there any noticeable differences?)

Solution:
When we create a linear interpolation of the data, the we can only get a certain amount of smoothness which is predicated on the number of samples, and their proximity. The kriging plot does a better job at creating a smooth map. Beyond that

consider the area around point 6 from the (f) plot. We can see that in the kriging plot we can see that this area isn't nearly as dark and it has to do with the two 'humps' in the data that are next to it influencing the prediction at that point.

Exercise 3: Kriging weights. Use the spherical semi-variogram model with sill = 4.0, range = 2.0, and nugget = 0 for the following locations:

longitude	0	0	1	1	1.1	0.9
latitude	0	1	1	0	-0.1	-0.1

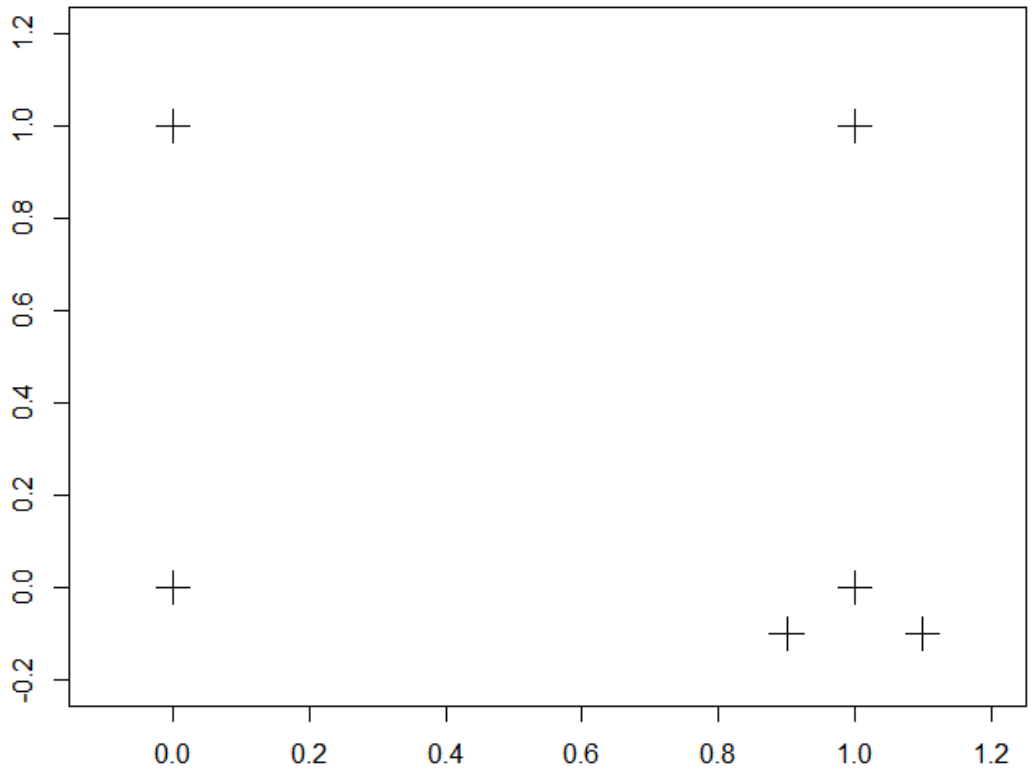
Construct a plot for these locations and calculate the kriging weights at each of the following five locations, s_0

longitude	.5	.01	.25	1.1	1.05
latitude	.5	.9	.75	-0.1	-0.05

In each case, comment on the weights, using your plot to assist in your description. For example, for each s_0, which observation locations have the largest weights, and does this make sense?

Solution:
First we begin by creating our kriging object with the desired spherical semi-variogram using krig.control(), then plotting our given locations,
Code:

```
my_kr_obj <- krige.control(
  type.krige="OK",cov.model="spherical",
  cov.pars=c(4,2),
  nugget=0)

lon <- c(0,0,1,1, 1.1, .9)
lat <- c(0,1,1,0, -.1,-.1)
locs <- cbind(lon, lat)
plot(lon, lat, xlim=c(-.1,1.2),
     ylim=c(-.2,1.2), xlab="",ylab="",
     pch = 3, cex = 2 )
```

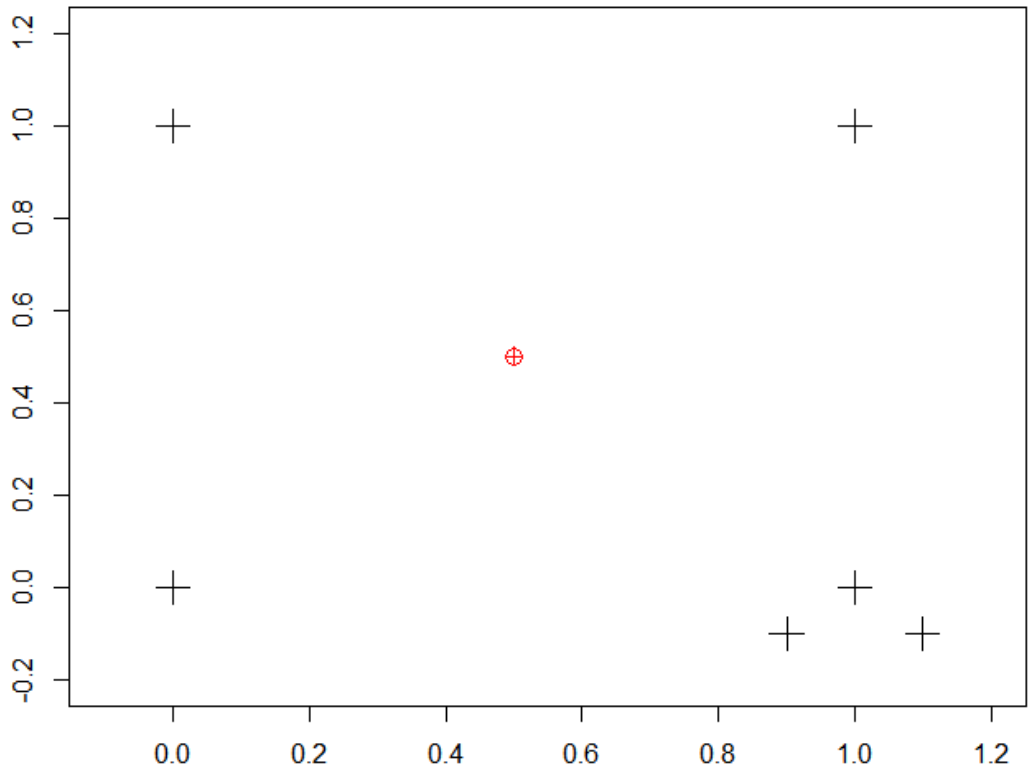


Computing the kriging weights for (.5, .5) we get the following with the order corresponding to our initial table.

$$[0.237, 0.251, 0.254, 0.267, -0.149, 0.139]$$

Code:

```
## Weights for (.5, .5)
#  0.237  0.251  0.254  0.267 -0.149  0.139
signif(krweights(locs, loc=c(0.5,0.5), krige=my_kr_obj), 3)
points(0.5,0.5, pch=10, col = 'red', cex = 1.5);
```



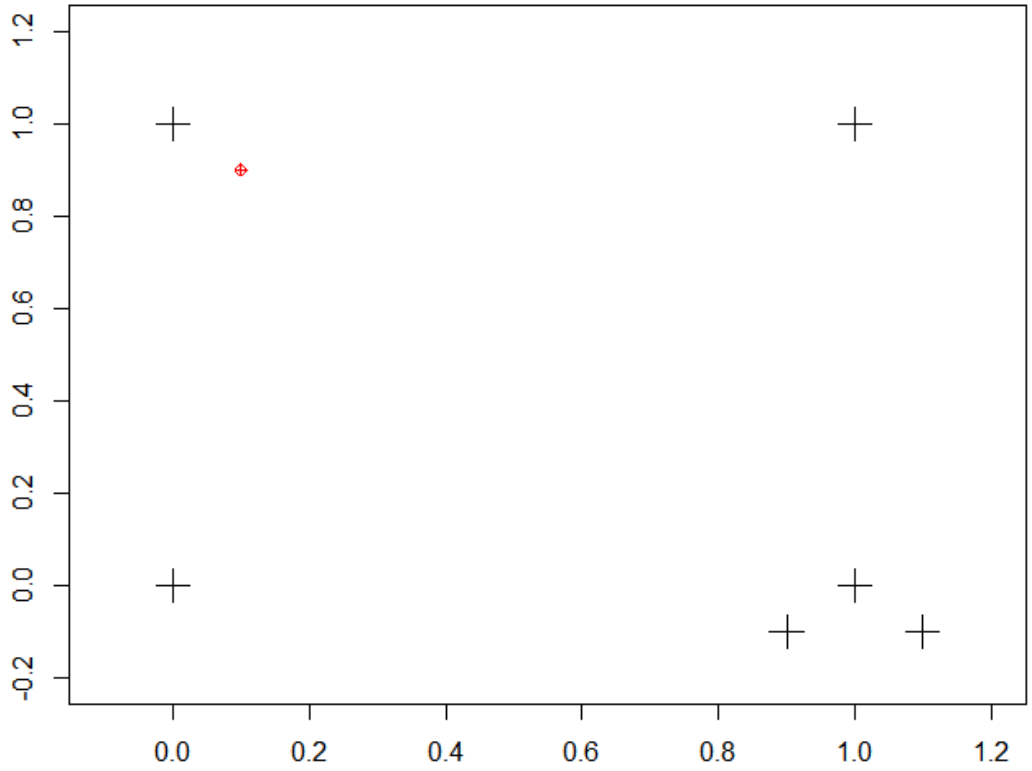
This makes sense, looking at the plot the point (.5, .5) is really in the middle of the plot with the first four locations being equidistant from the (.5, .5) point, and the last cancelling each other out.

Computing the kriging weights for (.1, .9) we get the following with the order corresponding to our initial table.

$$[0.0793, 0.8280, 0.0811, 0.0420, -0.0440, 0.0140]$$

Code:

```
## Weights for (.1, .9)
#  0.0793  0.8280  0.0811  0.0420 -0.0440  0.0140
signif(krweights(locs, loc=c(0.1,0.9), krige=my_kr_obj), 3)
points(0.1,0.9, pch=10, col = 'red');
```



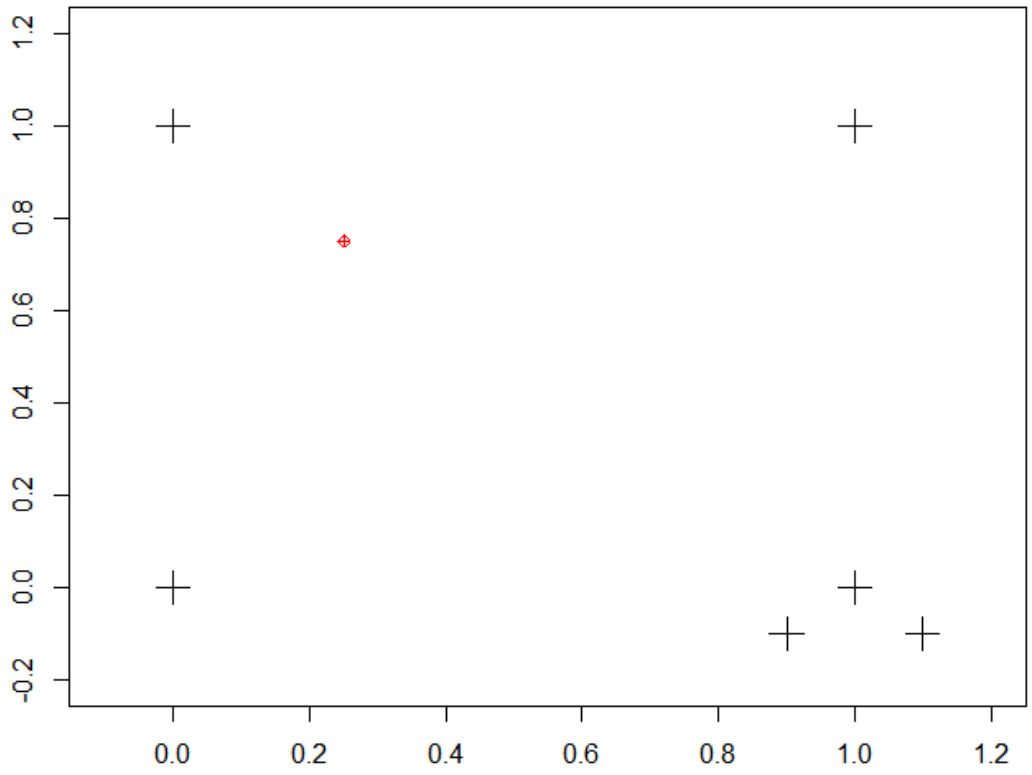
Given that (.1, .9) is closest to (0 , 1) it makes sense that the second weight is the largest. We also see that the first and third weights are relatively the same since they correspond to the (0,0) and (1,1) respectively. The fourth and fifth weight cancel each other out and the last weight sort of makes up for the difference between the first and third.

Computing the kriging weights for (.25, .75) we get the following with the order corresponding to our initial table.

$$[0.1750, 0.5820, 0.1810, 0.1130, -0.0969, 0.0462]$$

Code:

```
## Weights for (.25, .75)
#   0.1750  0.5820  0.1810  0.1130 -0.0969  0.0462
signif(krweights(locs, loc=c(0.25,0.75), krige=my_kr_obj), 3)
points(0.25,0.75, pch=10, col = 'red');
```



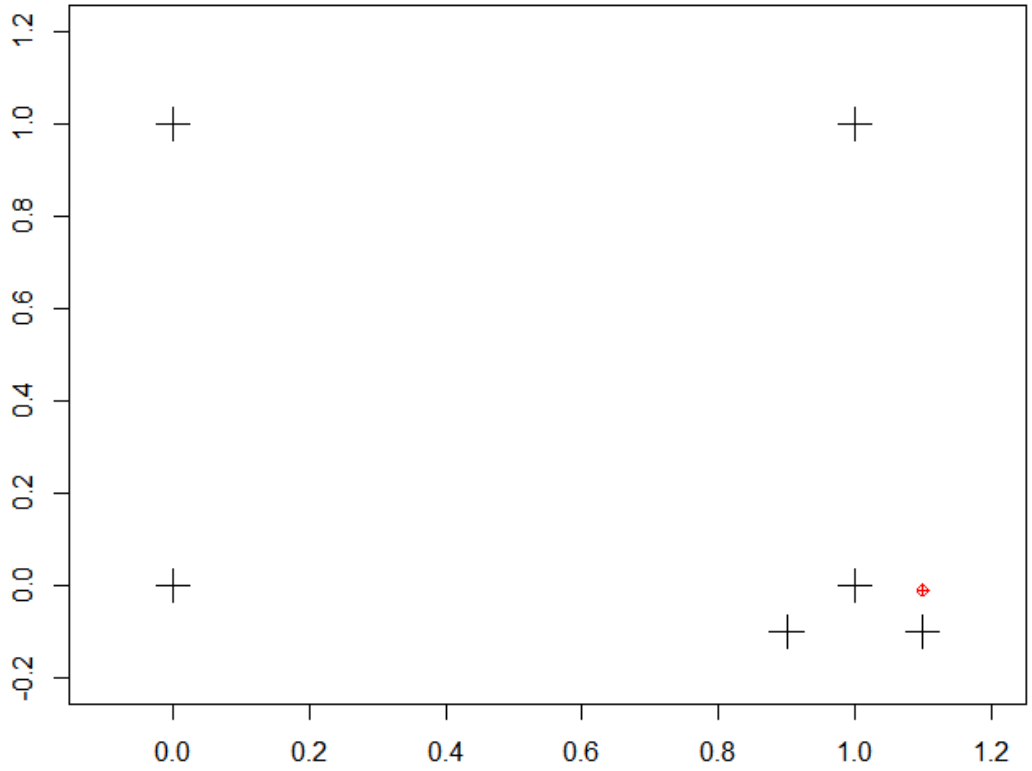
This point results in similar weight to the last point with the second weight being a little smaller and the rest of them generally increasing. We still have the fourth and fifth weight, however I imagine the weight of the three locations in the lower right is increased from the last point.

Computing the kriging weights for (1.1, -.01) we get the following with the order corresponding to our initial table.

$$[-0.006830, -0.000545, 0.034600, 0.501000, 0.555000, -0.083900]$$

Code:

```
## Weights for (1.1, -.01)
# -0.006830 -0.000545  0.034600  0.501000  0.555000 -0.083900
signif(krweights(locs, loc=c(1.1,-.01), krige=my_kr_obj), 3)
points(1.1,-.01, pch=10, col = 'red');
```



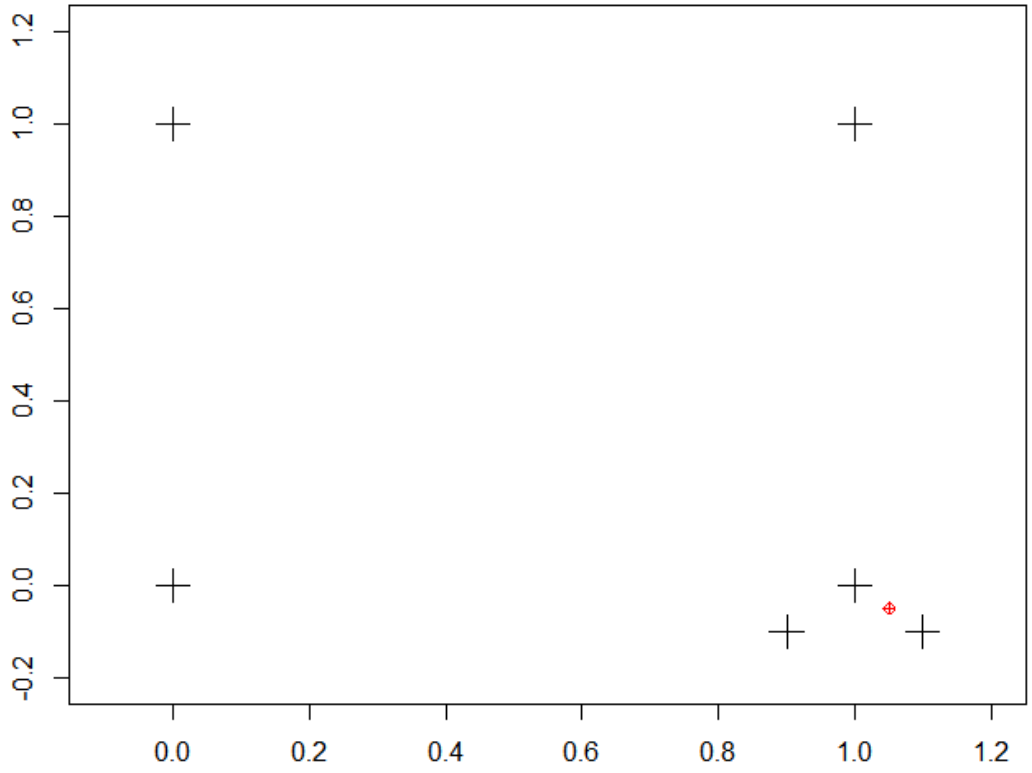
Given that (1.1, -.01) is nearest to the three locations in the lower right corner it makes sense that all other weights should be very small. It seems as though weight 3 and specifically 6 might be cancelling each other out with the rest of the influence from the three grouped locations being spread between weight 4 and 5.

Computing the kriging weights for (1.05, -.05) we get the following with the order corresponding to our initial table.

$$[-0.00514, -0.00158, 0.00155, 0.46600, 0.48600, 0.05300]$$

Code:

```
## Weights for (1.1, -.01)
# -0.006830 -0.000545  0.034600  0.501000  0.555000 -0.083900
signif(krweights(locs, loc=c(1.1,-.01), krige=my_kr_obj), 3)
points(1.1,-.01, pch=10, col = 'red');
```



Again we have a similar situation as before with the point (1.05, -.05) being really close to the three location in the lower right.