C173 Final Project

Daren Sathasivam

2025-03-12

Clean data

```r
a6 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/soil_complete.txt", header=TRU
str(a6)
```

```
## 'data.frame':    155 obs. of  6 variables:
##  $ x      : int  181072 181025 181165 181298 181307 181390 181165 181027 181060 181232 ...
##  $ y      : int  333611 333558 333537 333484 333330 333260 333370 333363 333231 333168 ...
##  $ cadmium: num  11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
##  $ copper : int  85 81 68 81 48 61 31 29 37 24 ...
##  $ lead   : int  299 277 199 116 117 137 132 150 133 80 ...
##  $ zinc   : int  1022 1141 640 257 269 281 346 406 347 183 ...
```

```r
# Repeated locations. Use one or average values
nrow(a6) == nrow(unique(a6[, c("x", "y")]))
```

```
## [1] TRUE
```

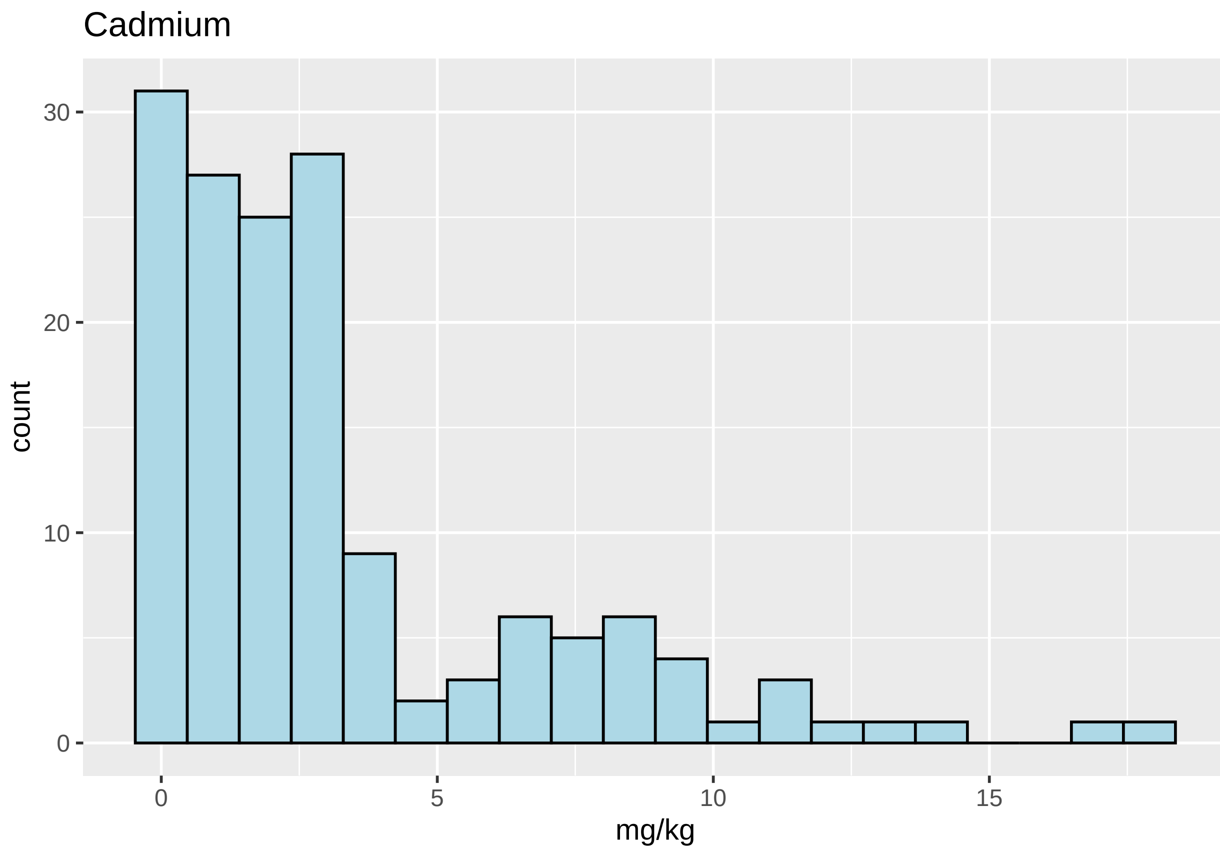
```r
# Aggregate data i.e. per year or per month(if time is involved)
```

Non-Spatial Data Analysis

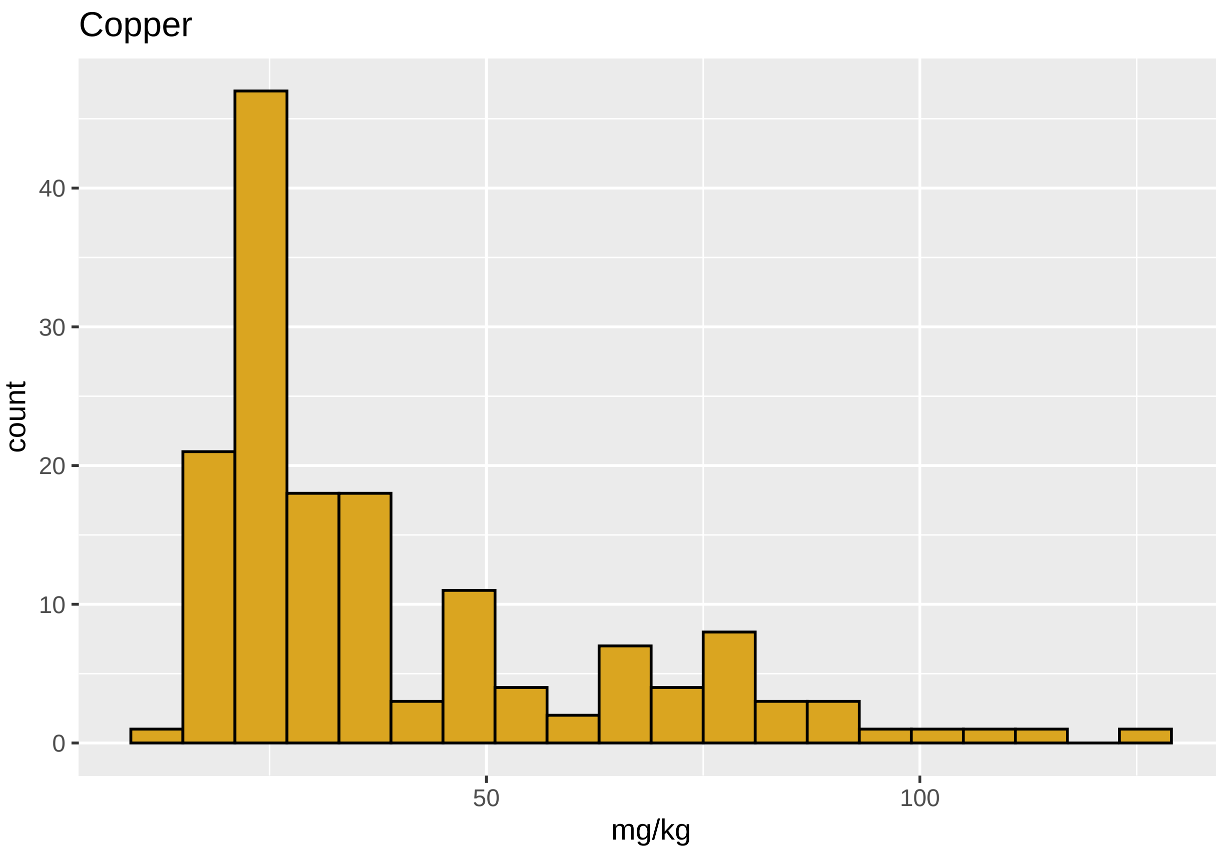
Variable Descriptions:

- 1. *x*
  - Description: The x-coordiante of the sampling location.
  - Type: Integer
- 2. *y*
  - Description: The y-coordinate of the sampling location.
  - Type: Integer
- 3. *cadmium*
  - Description: Cadmium concentration in the soil sample. It is the target variable due to its carcinogenic effects and persistence in ecosystems.
  - Relevance: Heavy metal pollutant which can be toxic and regulated in soil due to health risks.
  - Type: Numeric(floating point)
  - Typical Units: (Assumed) milligrams per kilogram (mg/kg)
- 4. *copper*
  - Description: Copper concentration in the soil sample
  - Relevance: a trace metal which can be toxic to plants and animals in excess.
  - Type: Integer
  - Typical Units: (Assumed) milligrams per kilogram (mg/kg)
- 5. *lead*
  - Description: Lead concentration in the soil sample
  - Relevance: A toxic metal which can poise a health concern for humans after prolonged and increased exposure.
  - Type: Integer
  - Typical Units: (Assumed) milligrams per kilogram (mg/kg)
- 6. *zinc*
  - Description: Zinc concentration in the soil sample
  - Relevance: A trace element in soil that is essential for plant growth but can be harmful in excess amounts.
  - Type: Integer
  - Typical Units: (Assumed) milligrams per kilogram (mg/kg)

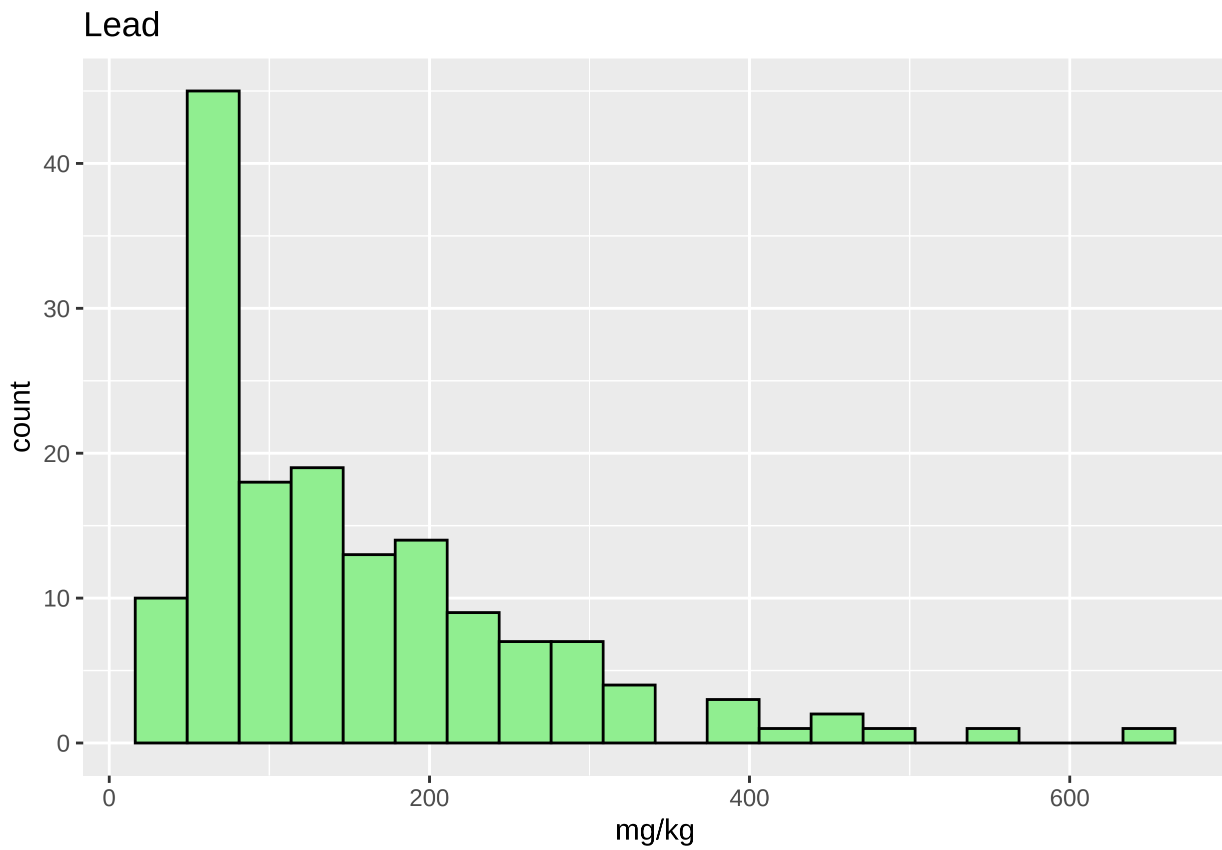
```r
library(ggplot2)
library(tidyr)
library(e1071)
# Histograms
par(mfrow = c(1, 2))
# Cd
ggplot(a6, aes(x = cadmium)) +
  geom_histogram(bins = 20, fill = "lightblue", color = "black") +
  labs(title = "Cadmium", x = "mg/kg")
```



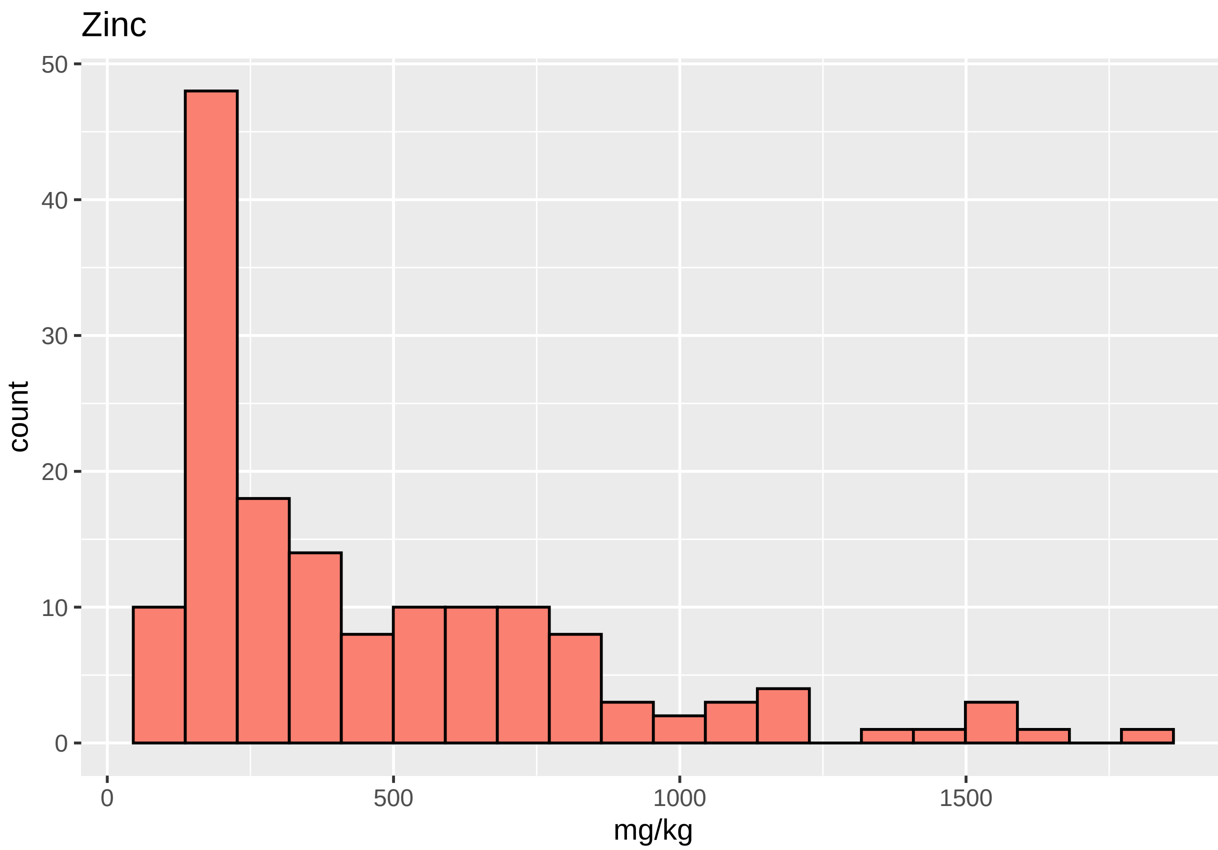

```r
# Cu
ggplot(a6, aes(x = copper)) +
  geom_histogram(bins = 20, fill = "goldenrod", color = "black") +
  labs(title = "Copper", x = "mg/kg")
```



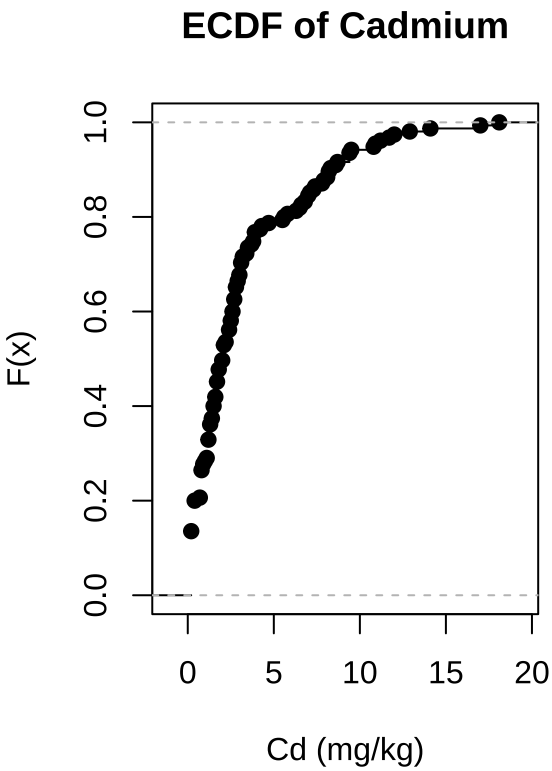

```r
# Pb
ggplot(a6, aes(x = lead)) +
  geom_histogram(bins = 20, fill = "lightgreen", color = "black") +
  labs(title = "Lead", x = "mg/kg")
```

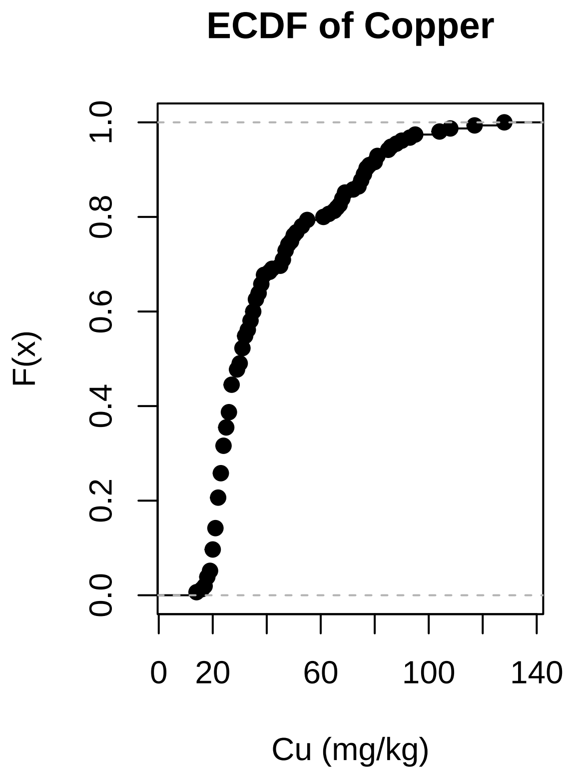



```r
# Zn
ggplot(a6, aes(x = zinc)) +
  geom_histogram(bins = 20, fill = "salmon", color = "black") +
  labs(title = "Zinc", x = "mg/kg")
```

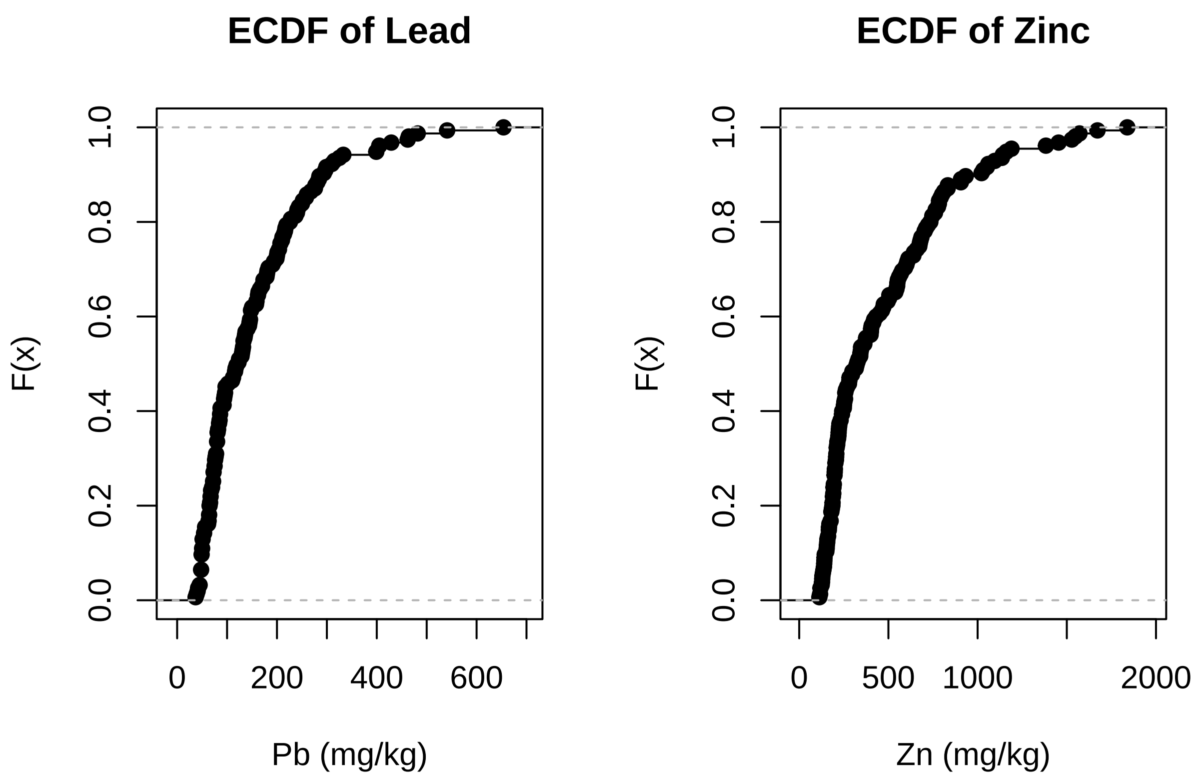



```r
# ECDFs
# Cd
plot(ecdf(a6$cadmium),
     main = "ECDF of Cadmium",
     xlab = "Cd (mg/kg)",
     ylab = "F(x)")
# Cu
plot(ecdf(a6$copper),
     main = "ECDF of Copper",
     xlab = "Cu (mg/kg)",
     ylab = "F(x)")
```

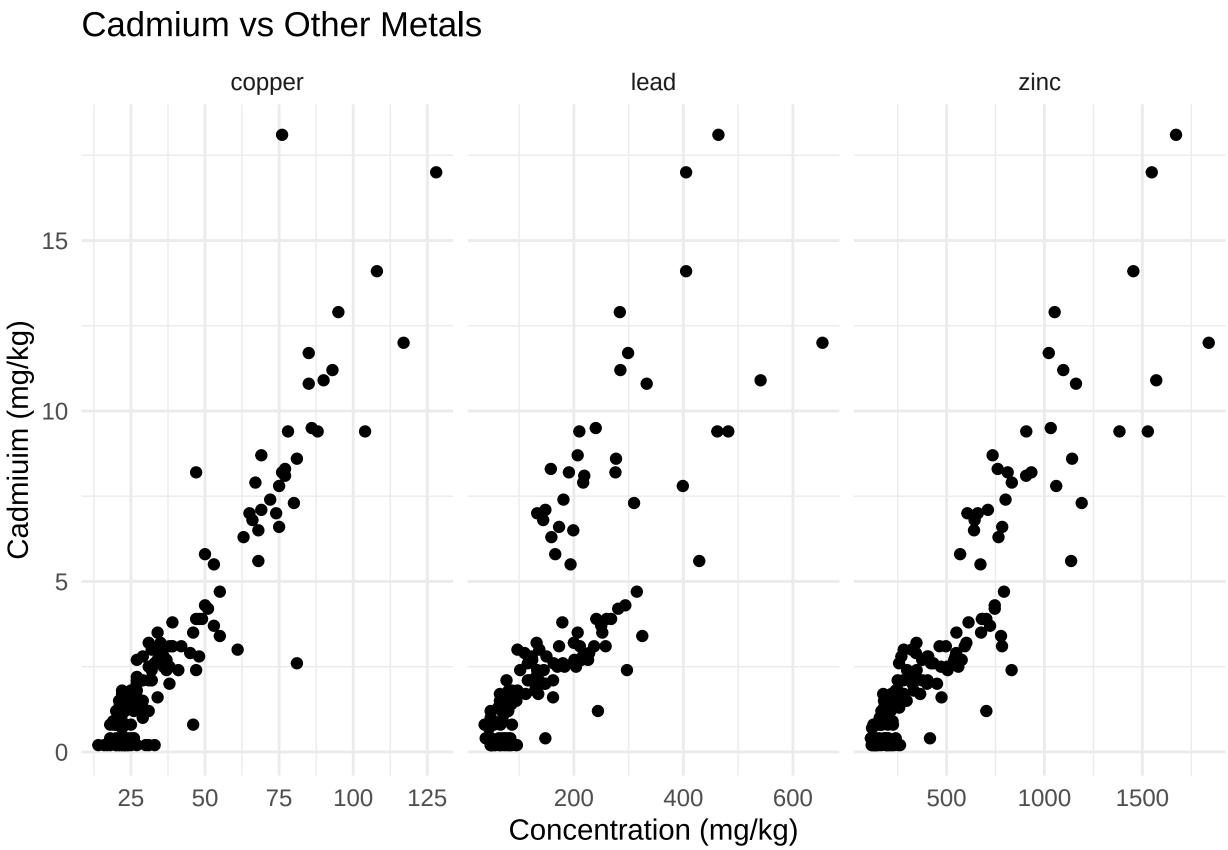





```
# Pb
plot(ecdf(a6$lead),
     main = "ECDF of Lead",
     xlab = "Pb (mg/kg)",
     ylab = "F(x)")
# Zn
plot(ecdf(a6$zinc),
     main = "ECDF of Zinc",
     xlab = "Zn (mg/kg)",
     ylab = "F(x)")
```




```r
# Scatterplots
a6_long <- a6 %>%
  pivot_longer(
    cols = c("copper", "lead", "zinc"),
    names_to = "metal",
    values_to = "concentration"
  )
ggplot(a6_long, aes(x = concentration, y = cadmium)) +
  geom_point() +
  facet_wrap(~ metal, scales = "free_x") +
  labs(title = "Cadmium vs Other Metals",
       x = "Concentration (mg/kg)",
       y = "Cadmiuim (mg/kg)") +
  theme_minimal()
```




```r
# Descriptive statistics
summary(a6)
```

```
##        x                y             cadmium           copper
##  Min.   :178605   Min.   :329714   Min.   : 0.200   Min.   : 14.00
##  1st Qu.:179371   1st Qu.:330762   1st Qu.: 0.800   1st Qu.: 23.00
##  Median :179991   Median :331633   Median : 2.100   Median : 31.00
##  Mean   :180005   Mean   :331635   Mean   : 3.246   Mean   : 40.32
##  3rd Qu.:180630   3rd Qu.:332463   3rd Qu.: 3.850   3rd Qu.: 49.50
##  Max.   :181390   Max.   :333611   Max.   :18.100   Max.   :128.00
##       lead            zinc
##  Min.   : 37.0   Min.   : 113.0
##  1st Qu.: 72.5   1st Qu.: 198.0
##  Median :123.0   Median : 326.0
##  Mean   :153.4   Mean   : 469.7
##  3rd Qu.:207.0   3rd Qu.: 674.5
##  Max.   :654.0   Max.   :1839.0
```

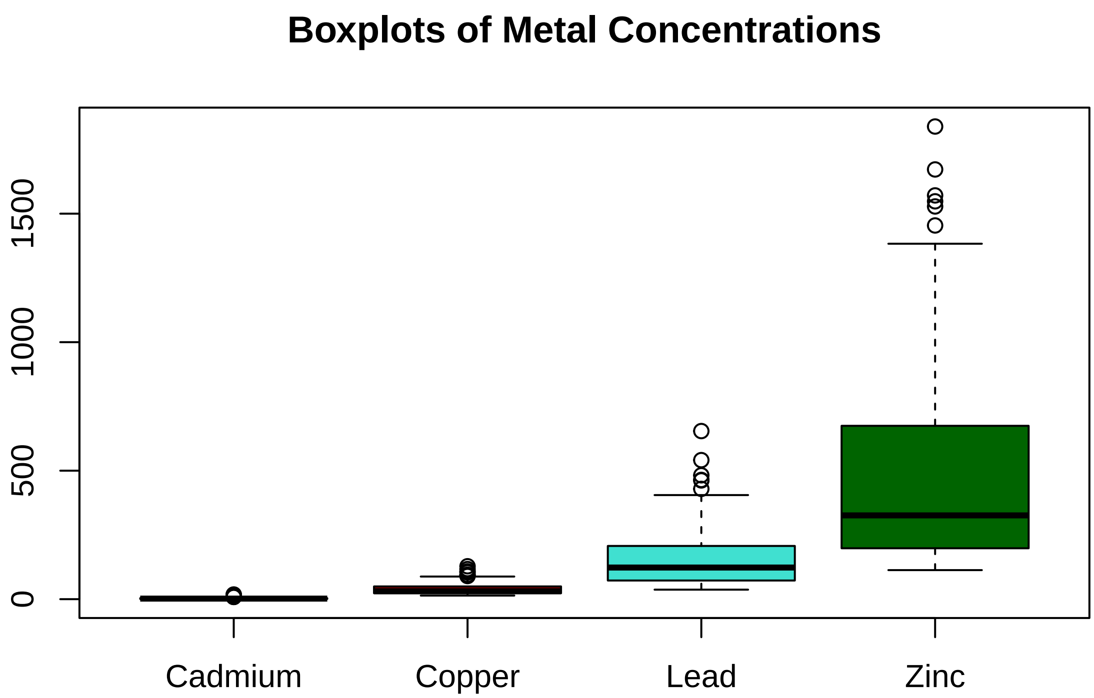
```r
sapply(a6[c("cadmium", "copper", "lead", "zinc")], sd)
```

```
##    cadmium     copper       lead       zinc
##   3.523746  23.680436 111.320054 367.073788
```

```r
sapply(a6[c("cadmium", "copper", "lead", "zinc")], function(x) c(skewness = skewness(x), kurtosis = kur
```

```
##           cadmium   copper     lead     zinc
## skewness 1.761609 1.386803 1.620493 1.457816
## kurtosis 3.051046 1.222590 3.062538 1.824645
```

```r
# Etc.
par(mfrow = c(1, 1))
# Boxplot
boxplot(a6$cadmium, a6$copper, a6$lead, a6$zinc,
        names = c("Cadmium", "Copper", "Lead", "Zinc"),
        main = "Boxplots of Metal Concentrations",
        col = c("black", "darkred", "turquoise", "darkgreen"))
```

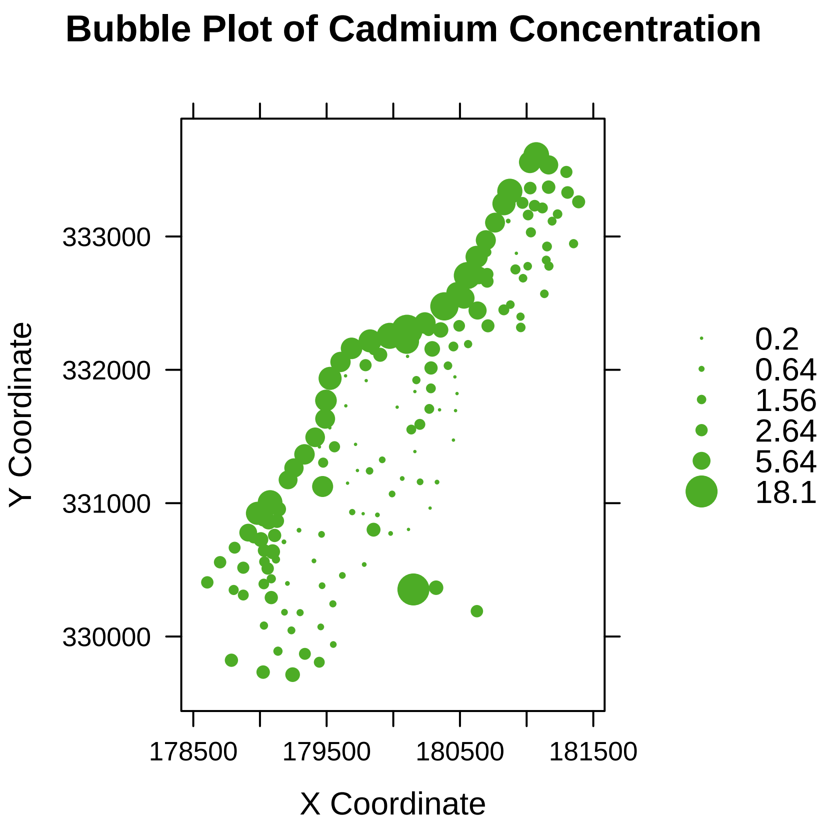



```r
# Correlation matrix
cor_matrix <- cor(a6[c("cadmium", "copper", "lead", "zinc")])
round(cor_matrix, 2)
```

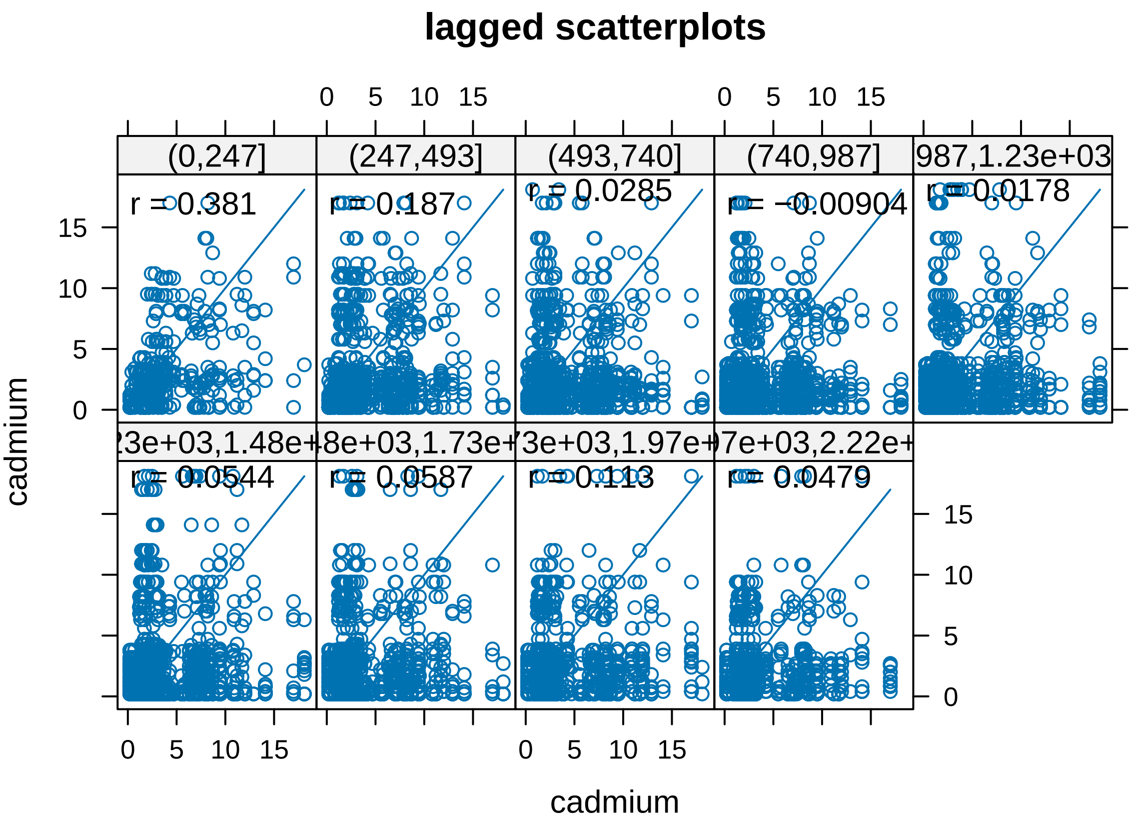
```
##         cadmium copper lead zinc
## cadmium    1.00   0.93 0.80 0.92
## copper     0.93   1.00 0.82 0.91
## lead       0.80   0.82 1.00 0.95
## zinc       0.92   0.91 0.95 1.00
```

Construct Bubble Plots and H-scatterplots

```
library(sp)
library(gstat)
# Bubble plot
a6_2 <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c173_c273/soil_complete.txt", header=
coordinates(a6_2) <- ~ x + y
bubble(a6_2, "cadmium",
       key.entries = quantile(a6$cadmium, probs = seq(0, 1, 0.2)),
       xlab = "X Coordinate", ylab = "Y Coordinate",
       main = "Bubble Plot of Cadmium Concentration",
       maxsize = 2,
       scales = list(draw = TRUE))
```




```
# H-scatterplot
max_dist <- max(spDists(a6_2))
h_breaks <- seq(0, max_dist / 2, length.out = 10)
hscat(cadmium ~ 1, data = a6_2, breaks = h_breaks)
```


lagged scatterplots
(0,247]
(247,493]
(493,740]
(740,987]
987,1.23e+03
r = 0.381
r = 0.187
r = 0.0285
r = −0.00904
r = 0.0178
r = 0.0544
r = 0.0587
r = 0.113
r = 0.0479
cadmium
cadmium

Variograms

```r
library(geoR)
```

```
## --------------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.9-4 (built on 2024-02-14) is now loaded
## --------------------------------------------------------------
```

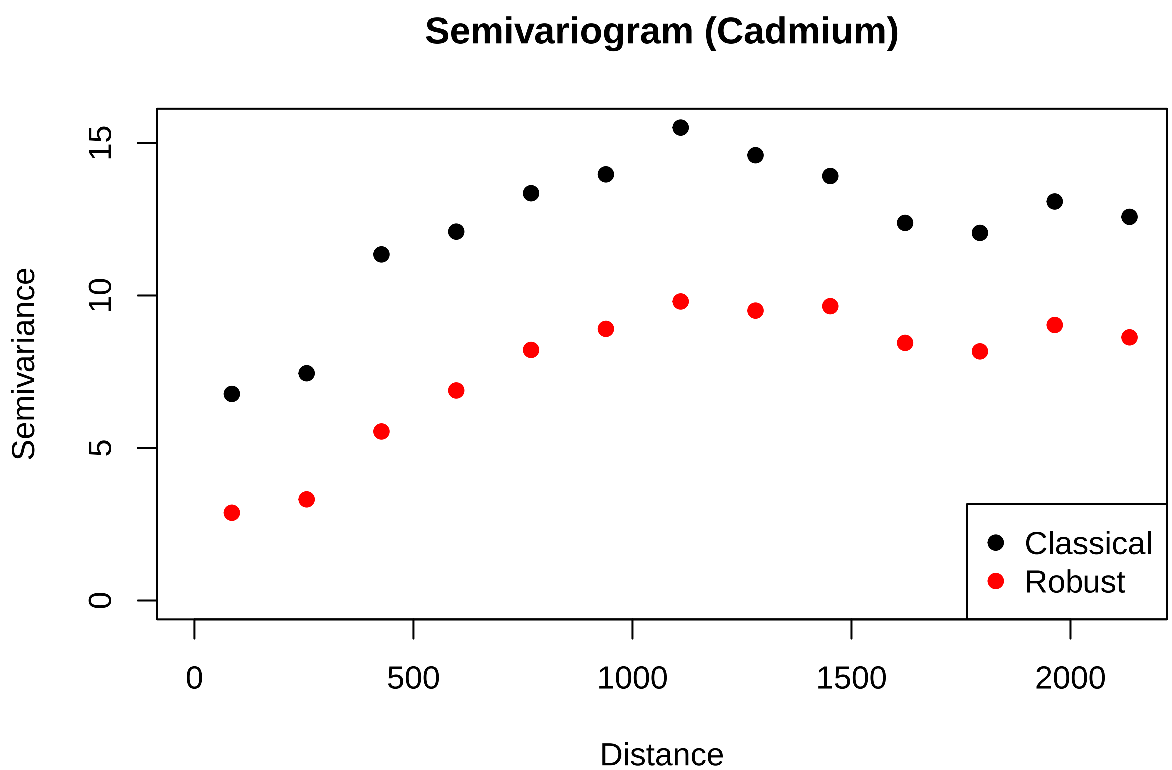
```r
# --- Compute empirical semivariogram --- #
soil_geodata <- as.geodata(a6, coords.col = 1:2, data.col = 3)
soil_max_dist <- max(dist(soil_geodata$coords))
# Classical
soil_classical <- variog(soil_geodata, max.dist = soil_max_dist / 2, estimator.type = "classical")
```

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

```r
# Robust
soil_robust <- variog(soil_geodata, max.dist = soil_max_dist / 2, estimator.type = "modulus")
```

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

```r
# Plot
plot(soil_classical, pch = 19,
     main = "Semivariogram (Cadmium)",
     xlab = "Distance",
     ylab = "Semivariance")
points(soil_robust$u, soil_robust$v, pch = 19, col = "red")
legend("bottomright",
       legend = c("Classical", "Robust"),
       pch = 19,
       col = c("black", "red"))
```




```r
# --- Fit Theoretical Models --- #
# OLS, WLS, MLE, By-eye, etc
### --- Classical --- ###
plot(soil_classical, main = "Classical Semivariogram for Soil Data with Various Fits")
# By-eye spherical
lines.variomodel(cov.model = "sph", cov.pars = c(12, 1000), nug = 2, max.dist = 2500, col = "red", lty =
# Spherical using default weights
soil_fit_default <- variofit(soil_classical, cov.model = "sph", ini.cov.pars = c(12, 1000), nugget = 2)
```

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

```r
soil_fit_default
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
##   4.5998   9.0462 891.7863
## Practical Range with cor=0.05 for asymptotic range: 891.7863
##
## variofit: minimised weighted sum of squares = 8368.817
```

```r
lines(soil_fit_default, col = "blue", lty = 1)
# Cressie's Weights
soil_fit_cressie <- variofit(soil_classical, cov.model = "sph", ini.cov.pars = c(12, 1000), nug = 2, we
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```r
soil_fit_cressie
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
##   5.0508   8.7007 936.0401
## Practical Range with cor=0.05 for asymptotic range: 936.0401
##
## variofit: minimised weighted sum of squares = 54.1002
```

```r
lines(soil_fit_cressie, col = "lightgreen", lty = 2)
# Equal weights
soil_fit_equal <- variofit(soil_classical, cov.model = "sph", ini.cov.pars = c(12, 1000), nug = 2, weig
```

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

```r
soil_fit_equal
```

```
## variofit: model parameters estimated by OLS (ordinary least squares):
## covariance model is: spherical
## parameter estimates:
##    tausq  sigmasq      phi
##   5.0932   8.4275 905.6494
## Practical Range with cor=0.05 for asymptotic range: 905.6494
##
## variofit: minimised sum of squares = 12.0672
```

```r
lines(soil_fit_equal, col = "orange", lty = 2)
# MML
soil_fit_mml <- likfit(soil_geodata, cov.model = "sph", ini.cov.pars = c(12, 1000), nug = 2, lik.method
```

```
## 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.
```

```r
soil_fit_mml
```

```
## likfit: estimated model parameters:
##       beta      tausq    sigmasq        phi
## "   4.803" "   1.551" "  19.367" "1000.000"
## Practical Range with cor=0.05 for asymptotic range: 999.9999
##
## likfit: maximised log-likelihood = -376
```

```r
lines(soil_fit_mml, col = "purple", lty = 2)
# Exp
```

```r
soil_fit_exp <- variofit(soil_classical, cov.model = "exp", ini.cov.pars = c(12, 1000), nug = 2)
```

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

```r
soil_fit_exp
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: exponential
## parameter estimates:
##    tausq  sigmasq      phi
##   1.8731  11.8671 280.2701
## Practical Range with cor=0.05 for asymptotic range: 839.6143
##
## variofit: minimised weighted sum of squares = 10692.93
```

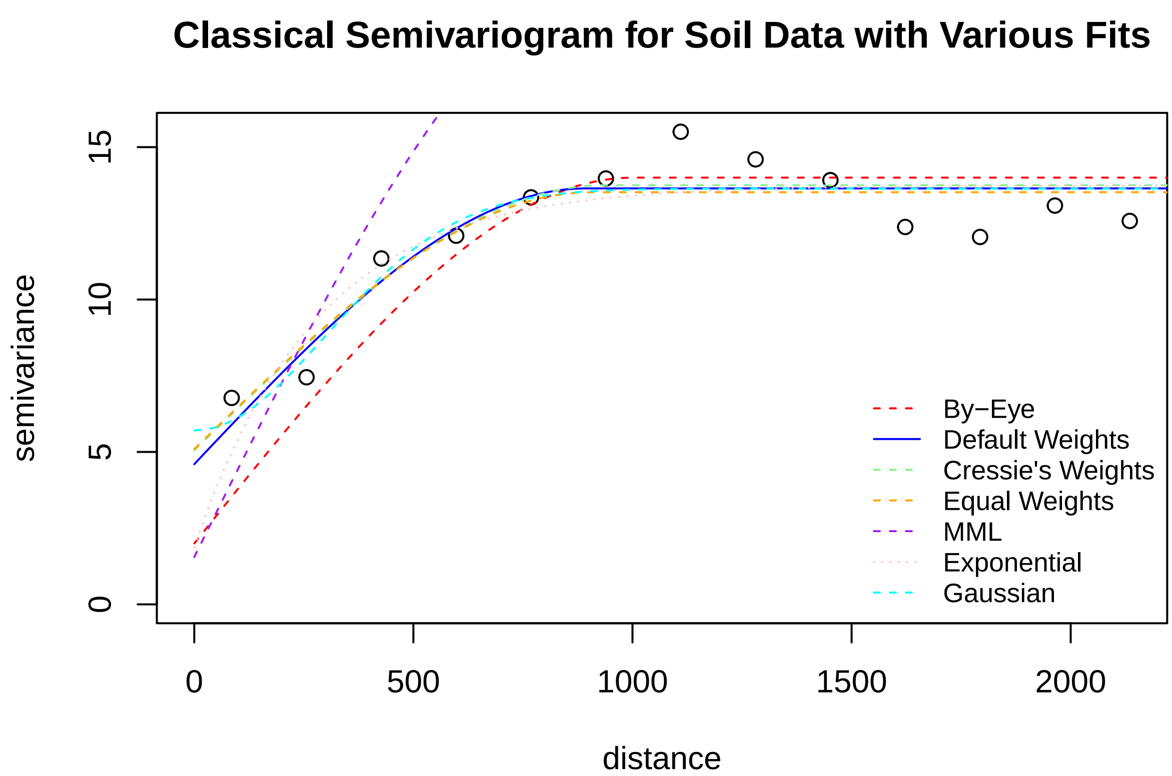
```r
lines(soil_fit_exp, col = "pink", lty = 3)
# Gaussian
soil_fit_gaus <- variofit(soil_classical, cov.model = "gau", ini.cov.pars = c(12, 1000), nug = 2)
```

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

```r
soil_fit_gaus
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: gaussian
## parameter estimates:
##    tausq  sigmasq      phi
##   5.7057   7.9333 425.1269
## Practical Range with cor=0.05 for asymptotic range: 735.8174
##
## variofit: minimised weighted sum of squares = 8081.331
```

```r
lines(soil_fit_gaus, col = "cyan", lty = 2)

legend("bottomright", legend = c("By-Eye",
                                 "Default Weights",
                                 "Cressie's Weights",
                                 "Equal Weights",
                                 "MML",
                                 "Exponential",
                                 "Gaussian"),
       col = c("red", "blue", "lightgreen", "orange", "purple", "pink", "cyan"),
       lty = c(2, 1, 2, 2, 2, 3, 2),
       bty = "n",
       cex = 0.8)
```




```
### --- Robust --- ###
plot(soil_robust, main = "Robust Semivariogram for Soil Data with Various Fits")
# By-eye spherical
lines.variomodel(cov.model = "sph", cov.pars = c(8.5, 1000), nug = 0, max.dist = 2500, col = "red", lty
# Spherical using default weights
soil_fit_default <- variofit(soil_robust, cov.model = "sph", ini.cov.pars = c(8.5, 1000), nugget = 0)
```

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

```
soil_fit_default
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##      tausq   sigmasq       phi
##     1.0643    8.0437 1107.9921
## Practical Range with cor=0.05 for asymptotic range: 1107.992
##
## variofit: minimised weighted sum of squares = 2052.782
```

```
lines(soil_fit_default, col = "blue", lty = 1)
# Cressie's Weights
soil_fit_cressie <- variofit(soil_robust, cov.model = "sph", ini.cov.pars = c(8.5, 1000), nug = 0, weig
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
soil_fit_cressie
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: spherical
## parameter estimates:
##     tausq   sigmasq       phi
##    1.6862    7.4763 1189.4223
## Practical Range with cor=0.05 for asymptotic range: 1189.422
##
## variofit: minimised weighted sum of squares = 49.8558
```

```r
lines(soil_fit_cressie, col = "lightgreen", lty = 2)
# Equal weights
soil_fit_equal <- variofit(soil_robust, cov.model = "sph", ini.cov.pars = c(8.5, 1000), nug = 0, weights
```

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

```r
soil_fit_equal
```

```
## variofit: model parameters estimated by OLS (ordinary least squares):
## covariance model is: spherical
## parameter estimates:
##     tausq   sigmasq       phi
##    1.4992    7.5492 1129.7170
## Practical Range with cor=0.05 for asymptotic range: 1129.717
##
## variofit: minimised sum of squares = 3.3052
```

```r
lines(soil_fit_equal, col = "orange", lty = 2)
# MML
soil_fit_mml <- likfit(soil_geodata, cov.model = "sph", ini.cov.pars = c(8.5, 1000), nug = 0, lik.metho
```

```
## 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.
```

```r
soil_fit_mml
```

```
## likfit: estimated model parameters:
##       beta      tausq    sigmasq        phi
## "   4.803" "   1.551" "  19.366" "1000.000"
## Practical Range with cor=0.05 for asymptotic range: 1000
##
## likfit: maximised log-likelihood = -376
```

```r
lines(soil_fit_mml, col = "purple", lty = 2)
# Exp
soil_fit_exp <- variofit(soil_robust, cov.model = "exp", ini.cov.pars = c(8.5, 1000), nug = 0)
```

```
## variofit: covariance model used is exponential
```

```
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```r
soil_fit_exp
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: exponential
## parameter estimates:
##    tausq  sigmasq      phi
##   0.0000   9.3816 418.2434
## Practical Range with cor=0.05 for asymptotic range: 1252.945
##
## variofit: minimised weighted sum of squares = 4300.207
```

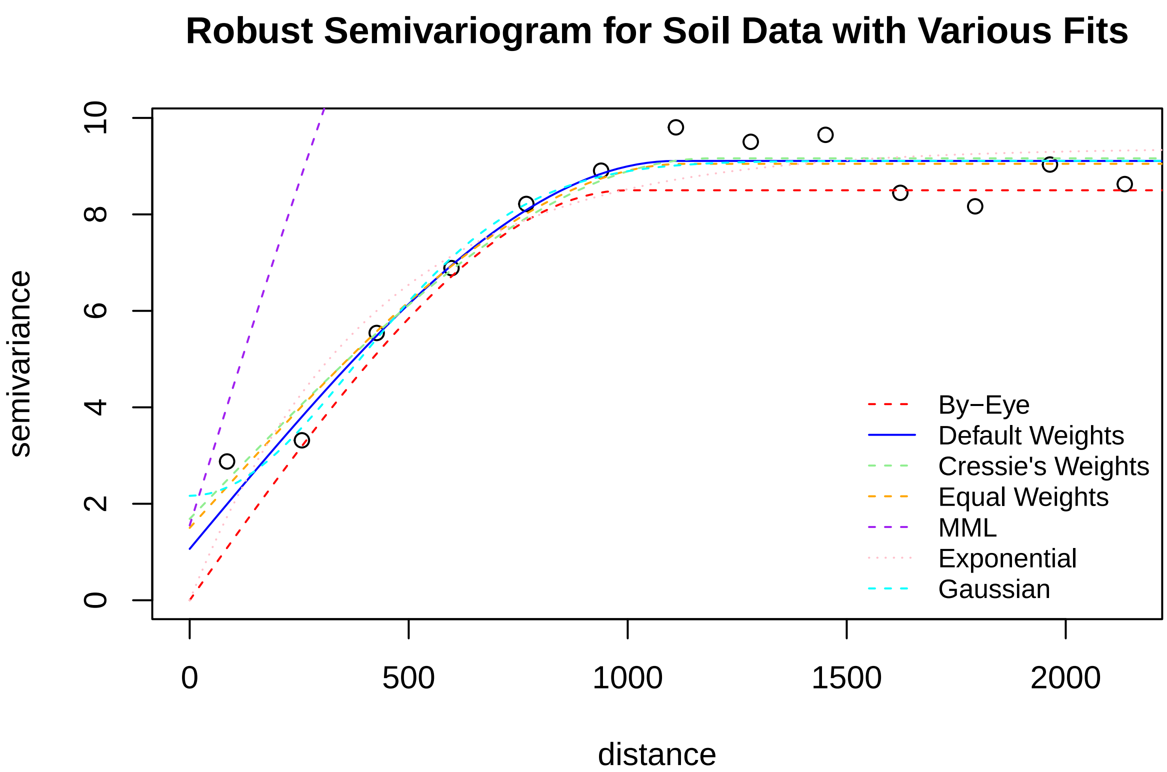
```r
lines(soil_fit_exp, col = "pink", lty = 3)
# Gaussian
soil_fit_gaus <- variofit(soil_robust, cov.model = "gau", ini.cov.pars = c(8.5, 1000), nug = 0)
```

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

```r
soil_fit_gaus
```

```
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: gaussian
## parameter estimates:
##    tausq  sigmasq      phi
##   2.1648   6.9457 536.7084
## Practical Range with cor=0.05 for asymptotic range: 928.9448
##
## variofit: minimised weighted sum of squares = 2012.337
```

```r
lines(soil_fit_gaus, col = "cyan", lty = 2)

legend("bottomright", legend = c("By-Eye",
                                 "Default Weights",
                                 "Cressie's Weights",
                                 "Equal Weights",
                                 "MML",
                                 "Exponential",
                                 "Gaussian"),
       col = c("red", "blue", "lightgreen", "orange", "purple", "pink", "cyan"),
       lty = c(2, 1, 2, 2, 2, 3, 2),
       bty = "n",
       cex = 0.8)
```

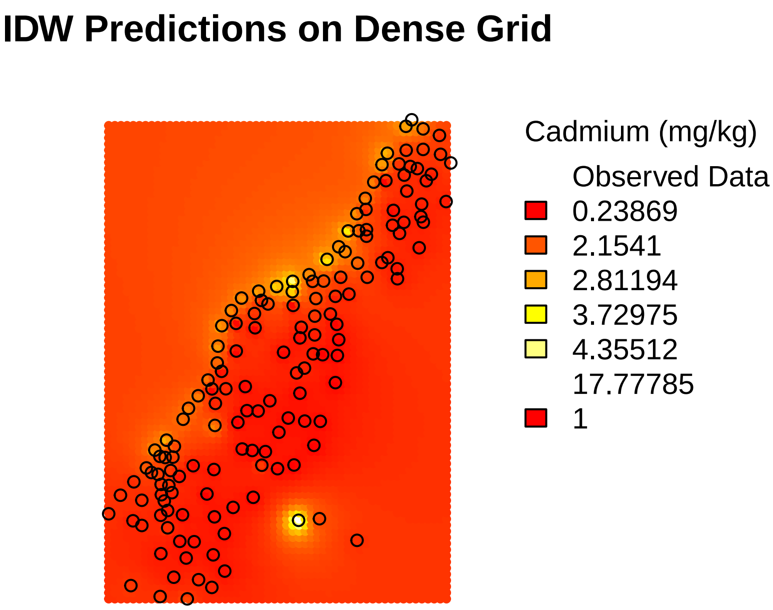

Robust Semivariogram for Soil Data with Various Fits
semivariance
distance
0
2
4
6
8
10
0
500
1000
1500
2000
By−Eye
Default Weights
Cressie's Weights
Equal Weights
MML
Exponential
Gaussian

Predictions on a dense grid using the iverse distance interpolation method.

```
# str(a6_2)
# Interpolation method predictions
x.range <- range(a6_2$x)
y.range <- range(a6_2$y)
dense_grid <- expand.grid(x = seq(from = x.range[1], to = x.range[2], by = 50), y = seq(from = y.range[
coordinates(dense_grid) <- ~ x + y

# Predict data points - idw = inverse distance weighted predictor
idw_pred <- idw(cadmium ~ 1, locations = a6_2, newdata = dense_grid)
```

```
## [inverse distance weighted interpolation]
```

```
# Plot
plot(dense_grid, pch = ".", col = "gray", main = "IDW Predictions on Dense Grid")
points(idw_pred, pch = 19, cex = 0.5, col = heat.colors(100)[cut(idw_pred$var1.pred, breaks = 100)])
points(a6_2, pch = 1, col = "black", cex = 0.8)
legend("topright", legend = c("Observed Data", round(quantile(idw_pred$var1.pred, probs = seq(0, 1, 0.2
```

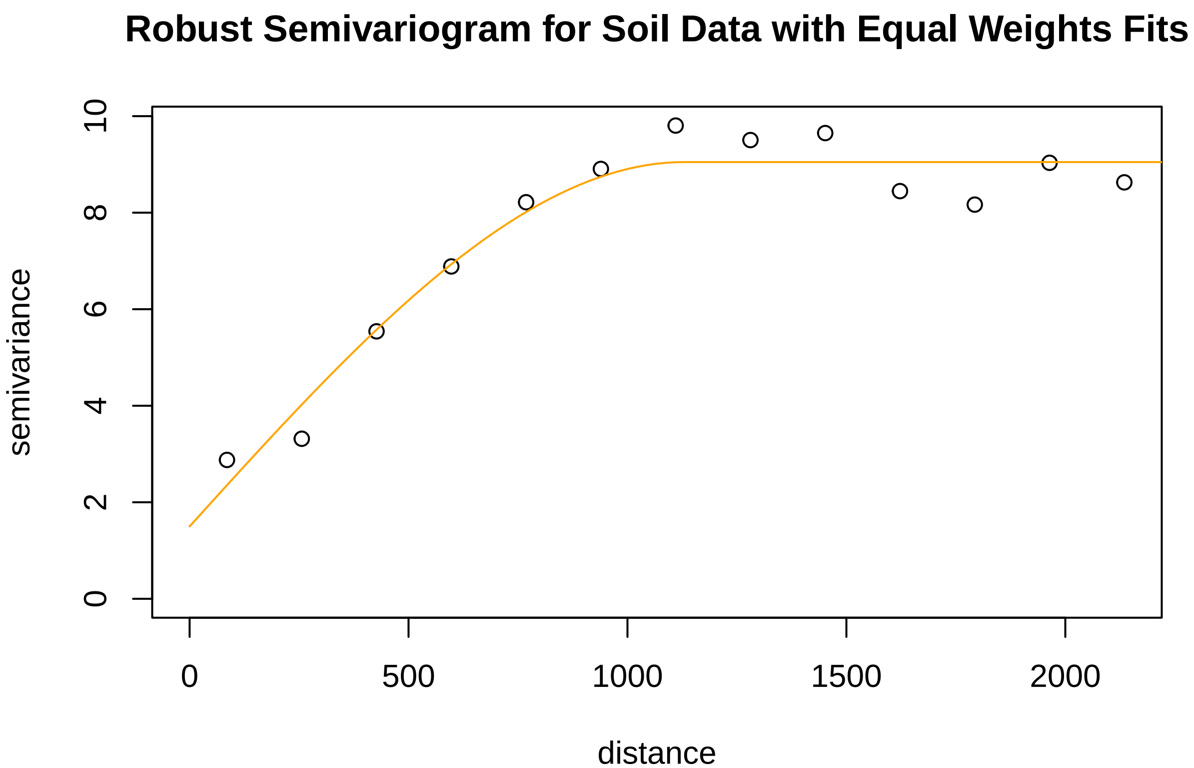


Predictions using Different Types of Kriging

```r
# Lab 20 - kriging in R & HW9, HW7, Lab6, Lab7

# Prepare data
coordinates(a6) <- ~ x + y
gridded(dense_grid) <- TRUE
# Fit variogram - use equal weights on robust variogram
plot(soil_robust, main = "Robust Semivariogram for Soil Data with Equal Weights Fits")
soil_fit_equal <- variofit(soil_robust, cov.model = "sph", ini.cov.pars = c(8.5, 1000), nug = 0, weights
```

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

```r
lines(soil_fit_equal, col = "orange", lty = 1)
```




```r
vgm_model_equal <- vgm(nugget = 1.5, model = "Sph", range = 1130, psill = 7.55)


# Ordinary Kriging
ok_pred <- krige(formula = cadmium ~ 1, locations = a6, newdata = dense_grid)
```

```
## [inverse distance weighted interpolation]
```

```r
summary(ok_pred)
```

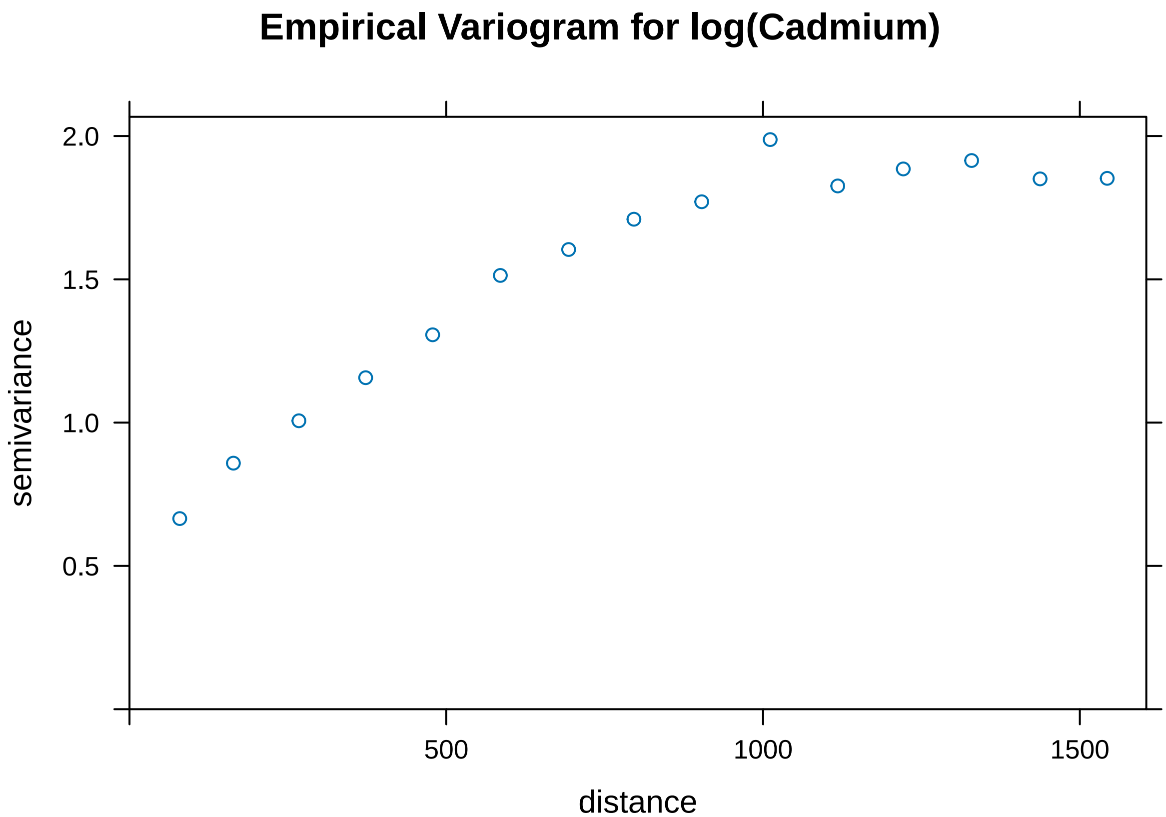
```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178580 181380
```

```
## y 329689 333589
## Is projected: NA
## proj4string : [NA]
## Number of points: 4368
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178605       50        56
## y            329714       50        78
## Data attributes:
##    var1.pred          var1.var
##  Min.   : 0.2387   Min.   : NA
##  1st Qu.: 2.3550   1st Qu.: NA
##  Median : 2.9754   Median : NA
##  Mean   : 3.3546   Mean   :NaN
##  3rd Qu.: 4.1519   3rd Qu.: NA
##  Max.   :17.7779   Max.   : NA
##                    NA's   :4368
```

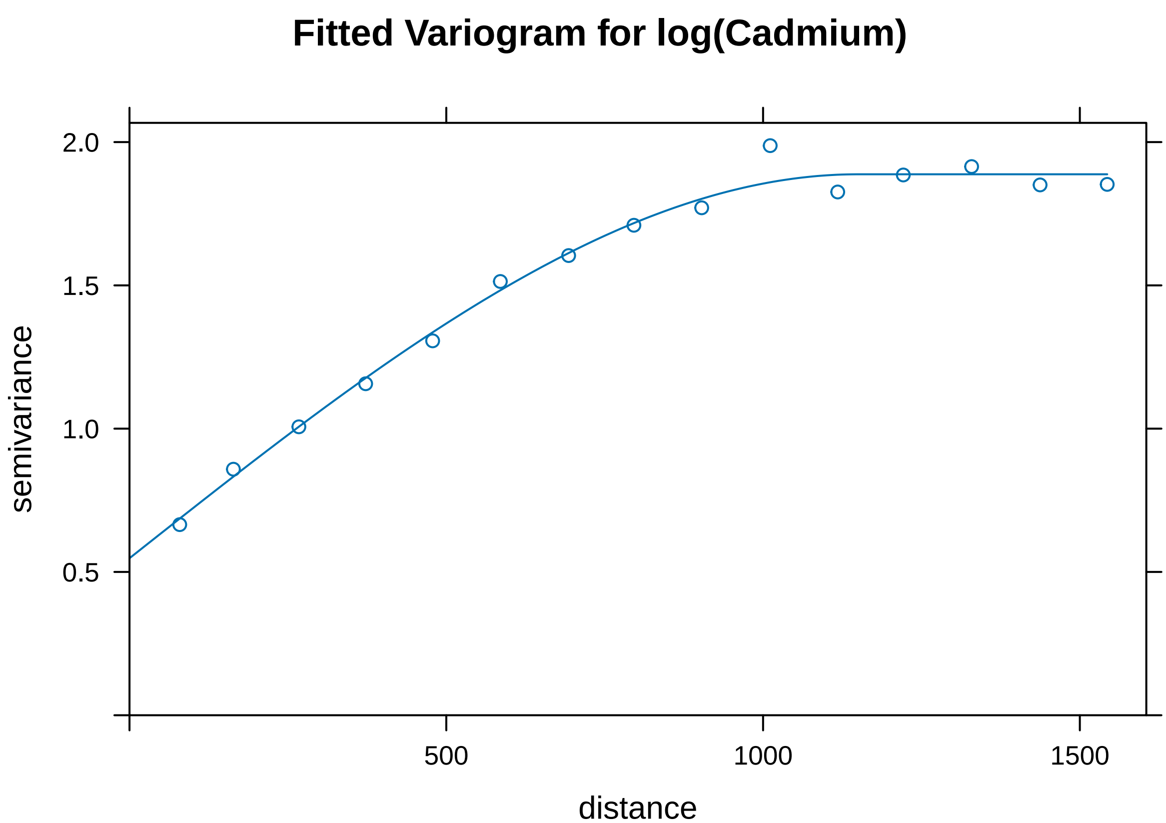
```
gridded(ok_pred) <- TRUE

cv_ok <- krige.cv(cadmium ~ 1, a6, model = vgm_model_equal)
PRESS_ok <- sum(cv_ok$residual^2, na.rm = TRUE)
cat("PRESS for Ordinary Kriging:", PRESS_ok, "\n")
```

```
## PRESS for Ordinary Kriging: 978.8476
```

```
# Log Ordinary Kriging
a6$log_cadmium <- log(a6$cadmium)
vgm_log <- variogram(log_cadmium ~ 1, data = a6)
plot(vgm_log, main = "Empirical Variogram for log(Cadmium)")
```




```
v_model_log <- fit.variogram(vgm_log, model = vgm(psill = 0.5, model = "Sph", range = 1000, nugget = 0.
plot(vgm_log, v_model_log, main = "Fitted Variogram for log(Cadmium)")
```



```r
log_ok_pred <- krige(formula = log_cadmium ~ 1, locations = a6, newdata = dense_grid, model = v_model_l
```

```
## [using ordinary kriging]
```

```r
summary(log_ok_pred)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178580 181380
## y 329689 333589
## Is projected: NA
## proj4string : [NA]
## Number of points: 4368
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178605       50        56
## y            329714       50        78
## Data attributes:
##    var1.pred          var1.var
##  Min.   :-1.2089   Min.   :0.0000
##  1st Qu.: 0.2199   1st Qu.:0.8499
##  Median : 0.9117   Median :1.2100
##  Mean   : 0.8247   Mean   :1.3113
##  3rd Qu.: 1.5226   3rd Qu.:1.7885
##  Max.   : 2.6912   Max.   :2.0219
```

```r
gridded(log_ok_pred) <- TRUE

cv_log_ok <- krige.cv(formula = log_cadmium ~ 1, locations = a6, model = v_model_log)
PRESS_log_ok <- sum(cv_log_ok$residual^2, na.rm = TRUE)
cat("PRESS for Log Ordinary Kriging:", PRESS_log_ok, "\n")
```

```
## PRESS for Log Ordinary Kriging: 125.9576
```

```r
# Simple Kriging
mean_cadmium <- mean(a6$cadmium)
sk_pred <- krige(formula = cadmium ~ 1, locations = a6, newdata = dense_grid, model = vgm_model_equal, 
```

```
## [using simple kriging]
```

```r
summary(sk_pred)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178580 181380
## y 329689 333589
## Is projected: NA
## proj4string : [NA]
## Number of points: 4368
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178605       50        56
## y            329714       50        78
## Data attributes:
##    var1.pred          var1.var
```

```
##  Min.   :-0.0165   Min.   :0.000
##  1st Qu.: 1.6881   1st Qu.:2.932
##  Median : 3.2458   Median :4.908
##  Mean   : 4.0887   Mean   :5.406
##  3rd Qu.: 6.2924   3rd Qu.:7.999
##  Max.   :13.0999   Max.   :9.050
```

```r
gridded(sk_pred) <- TRUE

cv_sk <- krige.cv(cadmium ~ 1, a6, model = vgm_model_equal, beta = mean_cadmium)
PRESS_sk <- sum(cv_sk$residual^2, na.rm = TRUE)
cat("PRESS for Simple Kriging:", PRESS_sk, "\n")
```

```
## PRESS for Simple Kriging: 981.9535
```

```r
# Log Simple Kriging
mean_log_cadmium <- mean(a6$log_cadmium)
log_sk_pred <- krige(formula = log_cadmium ~ 1, locations = a6, newdata = dense_grid, model = v_model_lo
```

```
## [using simple kriging]
```

```r
summary(log_sk_pred)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min    max
## x 178580 181380
## y 329689 333589
## Is projected: NA
## proj4string : [NA]
## Number of points: 4368
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178605       50        56
## y            329714       50        78
## Data attributes:
##    var1.pred          var1.var
##  Min.   :-1.2388   Min.   :0.0000
##  1st Qu.: 0.1128   1st Qu.:0.8494
##  Median : 0.6219   Median :1.1994
##  Mean   : 0.6882   Mean   :1.2736
##  3rd Qu.: 1.3621   3rd Qu.:1.7177
##  Max.   : 2.5918   Max.   :1.8876
```

```r
gridded(log_sk_pred) <- TRUE

cv_log_sk <- krige.cv(formula = log_cadmium ~ 1, locations = a6, model = v_model_log, beta = mean_log_ca
PRESS_log_sk <- sum(cv_log_sk$residual^2, na.rm = TRUE)
cat("PRESS for Log Simple Kriging (log-scale):", PRESS_log_sk, "\n")
```

```
## PRESS for Log Simple Kriging (log-scale): 126.5999
```

```r
# Universal Kriging
uk_pred <- krige(formula = cadmium ~ x + y, locations = a6, newdata = dense_grid)
```

```
## [ordinary or weighted least squares prediction]
```

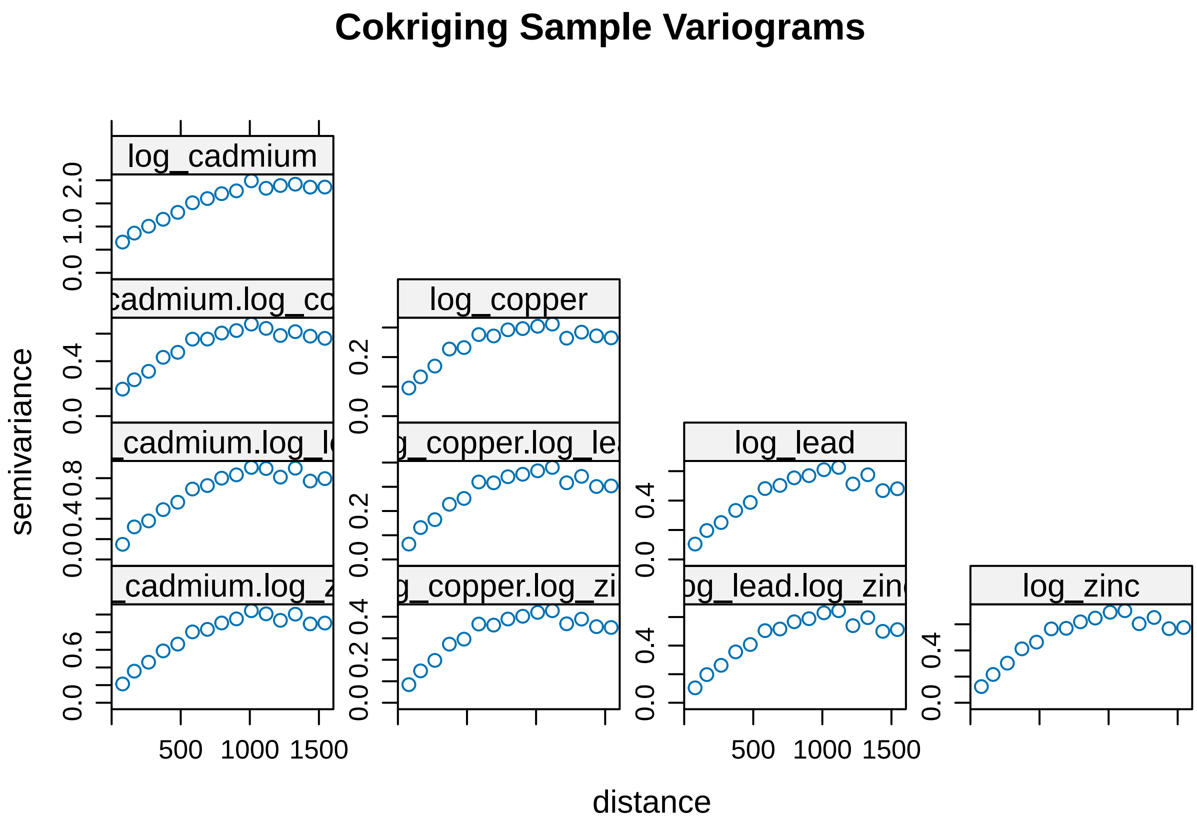
```r
summary(uk_pred)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##       min    max
## x 178580 181380
## y 329689 333589
## Is projected: NA
## proj4string : [NA]
## Number of points: 4368
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## x            178605       50        56
## y            329714       50        78
## Data attributes:
##     var1.pred          var1.var
##  Min.   :-5.7003   Min.   :10.65
##  1st Qu.: 0.6094   1st Qu.:10.80
##  Median : 3.3287   Median :10.99
##  Mean   : 3.3287   Mean   :11.26
##  3rd Qu.: 6.0479   3rd Qu.:11.52
##  Max.   :12.3576   Max.   :13.98
```

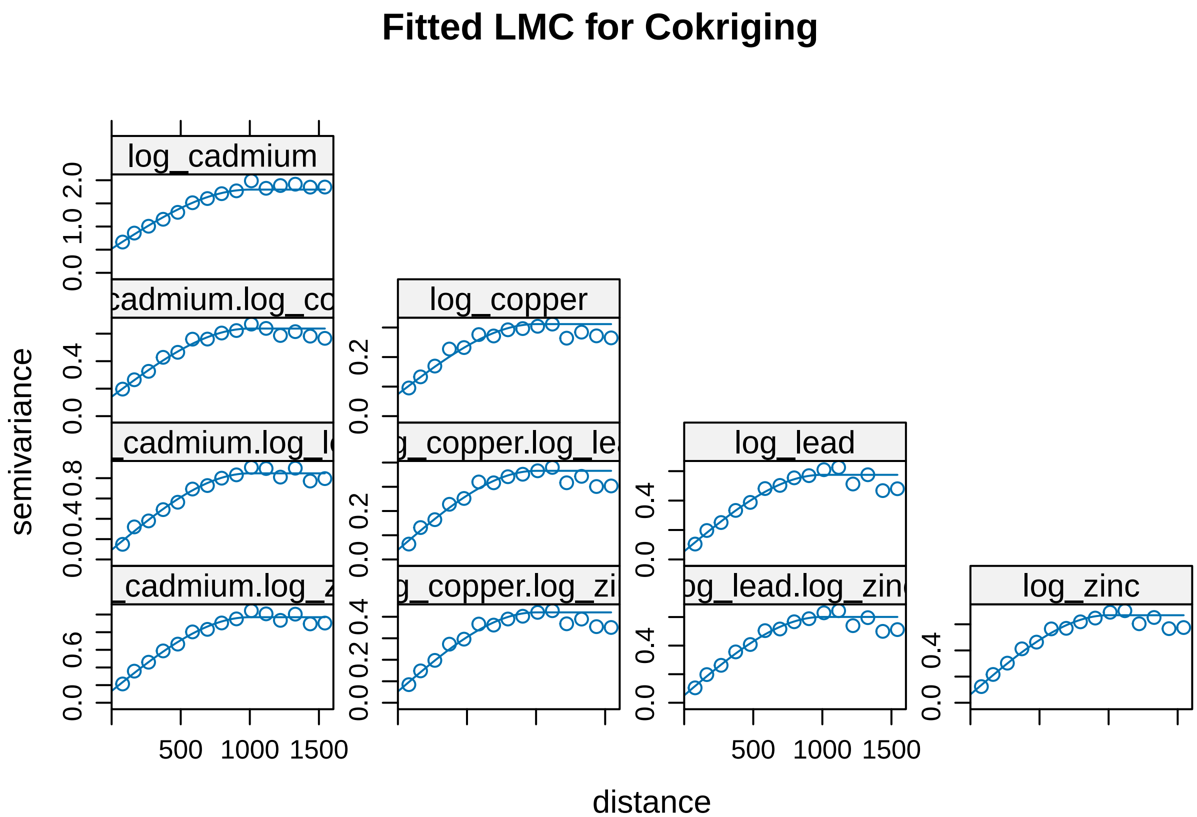
```r
gridded(uk_pred) <- TRUE

uk_cv <- krige.cv(formula = cadmium ~ x + y, locations = a6, model = vgm_model_equal)
PRESS_uk <- sum(uk_cv$residual^2, na.rm = TRUE)
cat("PRESS for Universal Kriging: ", PRESS_uk, "\n")
```

```
## PRESS for Universal Kriging:  940.693
```

```r
# Cokriging -- HW9
a6_df <- as.data.frame(a6)
a6_df$log_cadmium <- log(a6_df$cadmium)
a6_df$log_copper   <- log(a6_df$copper)
a6_df$log_lead     <- log(a6_df$lead)
a6_df$log_zinc     <- log(a6_df$zinc)
# gstat objects
g <- gstat(id = "log_cadmium", formula = log_cadmium ~ 1, locations = ~ x + y, data = a6_df)
g <- gstat(g, id = "log_copper",   formula = log_copper ~ 1, locations = ~ x + y, data = a6_df)
g <- gstat(g, id = "log_lead",     formula = log_lead ~ 1, locations = ~ x + y, data = a6_df)
g <- gstat(g, id = "log_zinc",     formula = log_zinc ~ 1, locations = ~ x + y, data = a6_df)
# variogram
vario_mult <- variogram(g)
plot(vario_mult, main = "Cokriging Sample Variograms")
```




```r
# Fit model
init_model <- vgm(psill = 0.5, model = "Sph", range = 1000, nugget = 0.1)
soil_fit <- fit.lmc(vario_mult, g, model = init_model)
plot(vario_mult, soil_fit, main = "Fitted LMC for Cokriging")
```




```r
# Predictions
ck_pred <- predict(soil_fit, newdata = dense_grid)
```

```
## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]
```

```r
summary(soil_fit)
```

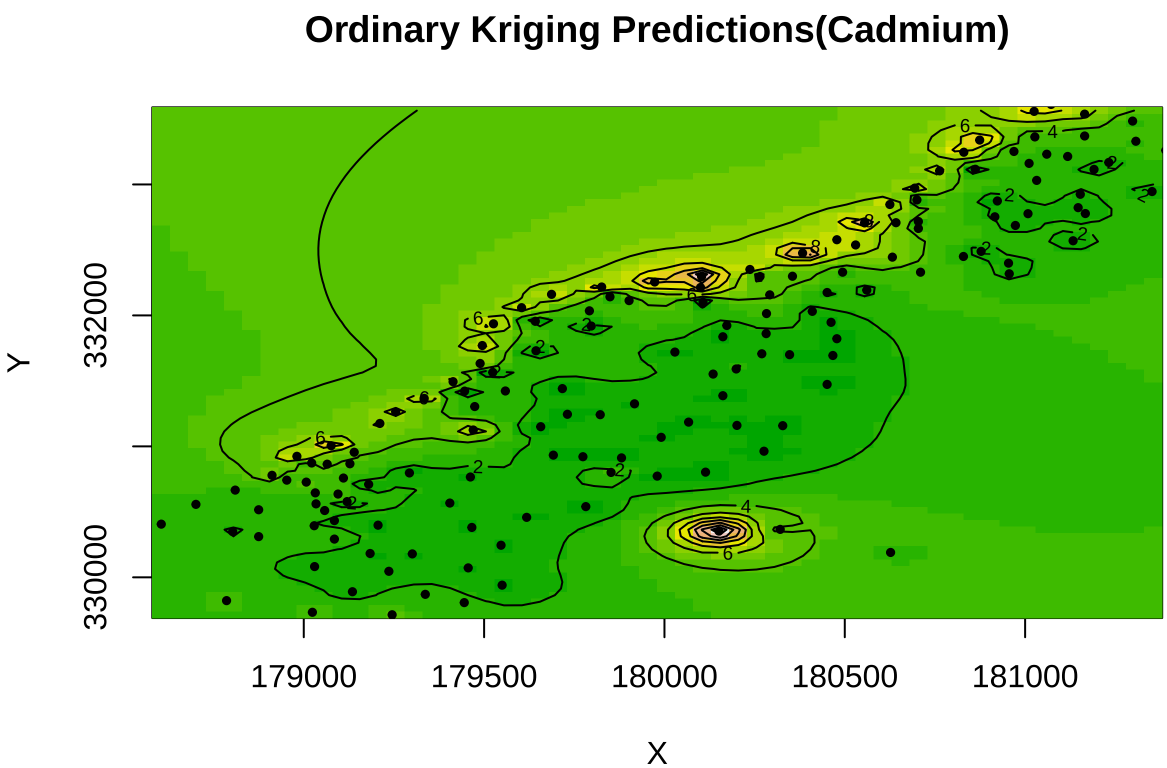
```
##           Length Class   Mode
## data       4     -none-  list
## model     10     -none-  list
## locations  2     formula call
## call       6     -none-  call
```

```r
invisible(capture.output({cv_ck <- gstat.cv(soil_fit, verbose = FALSE)}))
PRESS_ck <- sum(cv_ck$residual^2, na.rm = TRUE)
cat("PRESS for Cokriging: ", PRESS_ck, "\n")
```

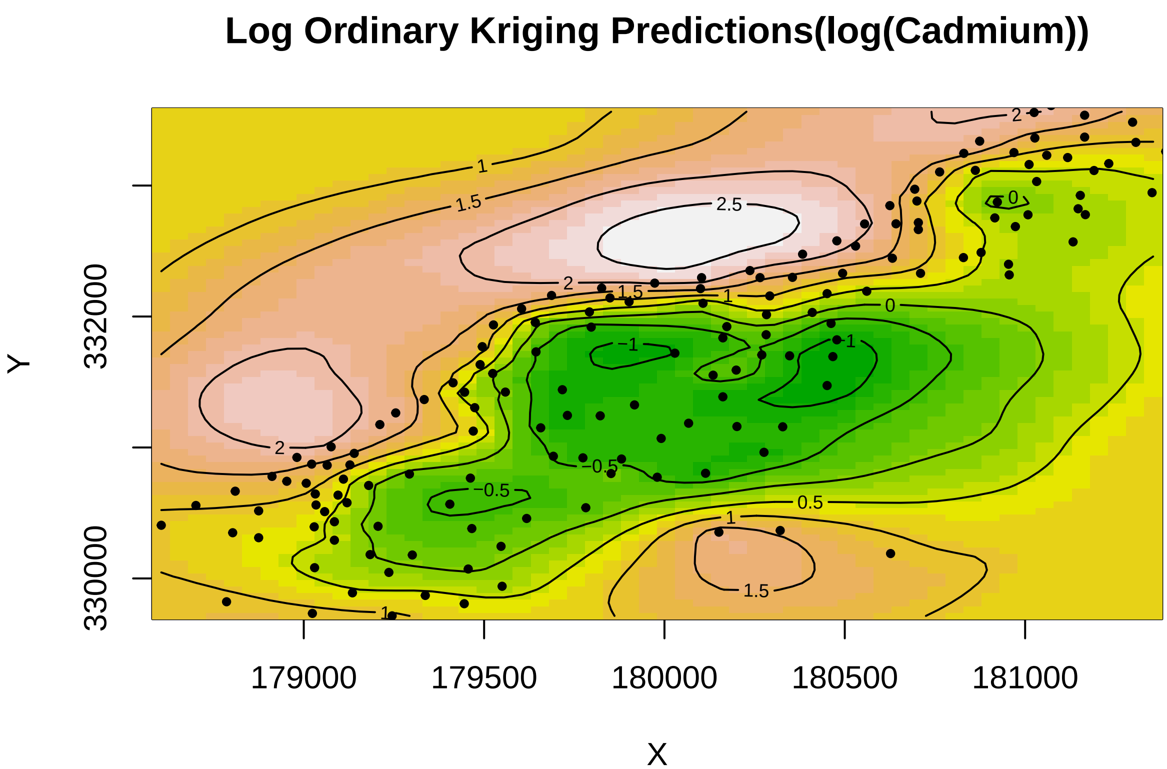
```
## PRESS for Cokriging:  41.99356
```

Construct a Raster Map and Add Contours

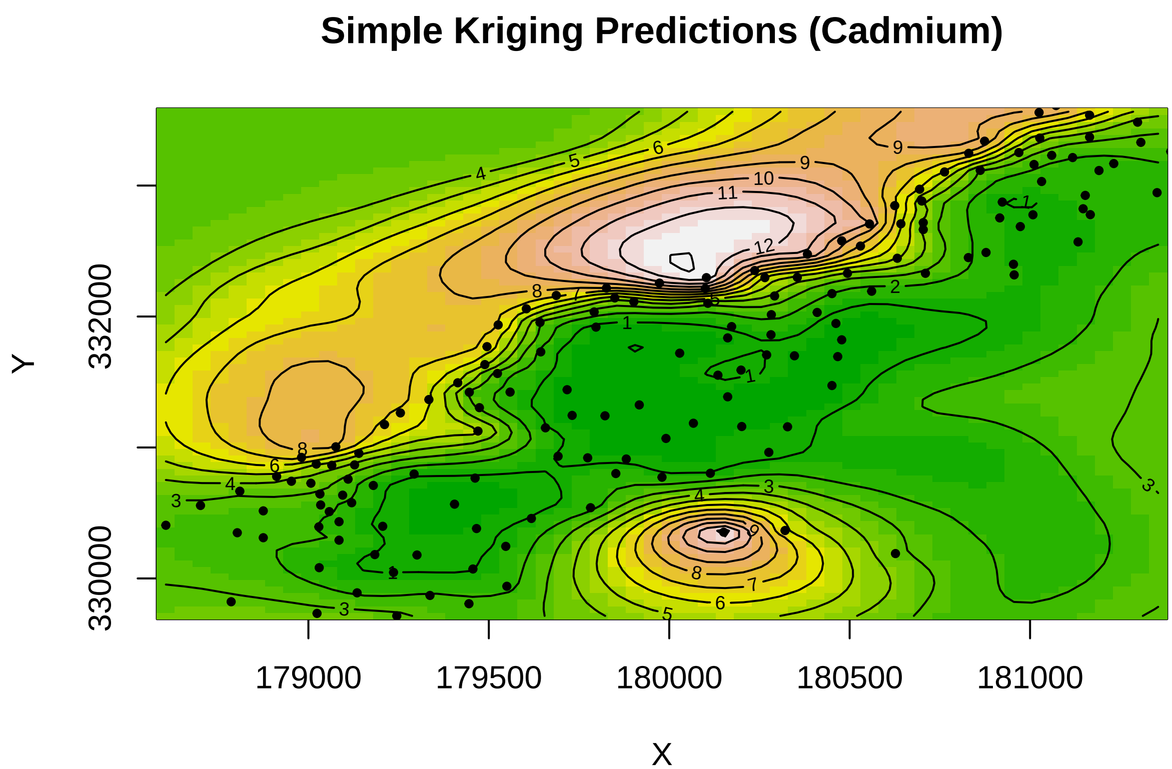
```r
# Raster Map w/ contours
x_seq <- unique(dense_grid$x)
y_seq <- unique(dense_grid$y)

# OK
ok_mat <- matrix(ok_pred$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, ok_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Ordinary Kriging Predictions(Cadmium)")
contour(x_seq, y_seq, ok_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```



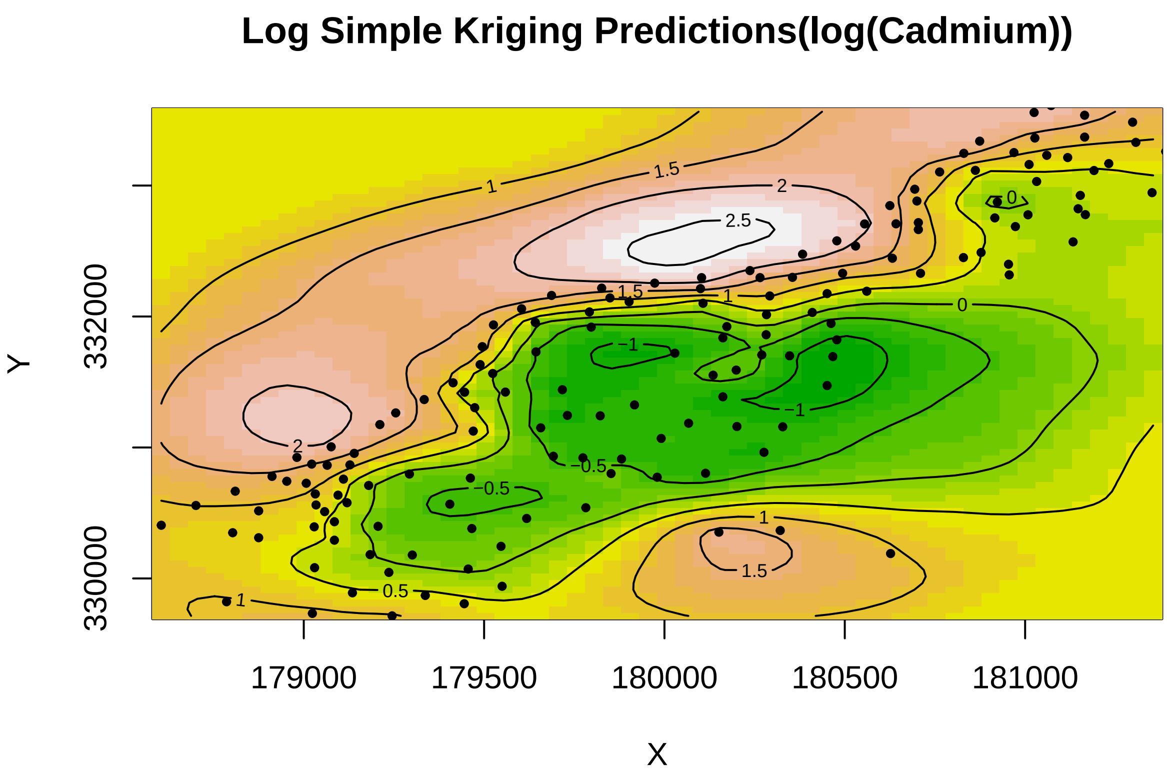

```r
# Log OK
log_ok_df <- as.data.frame(log_ok_pred)
log_ok_mat <- matrix(log_ok_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, log_ok_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Log Ordinary Kriging Predictions(log(Cadmium))")
contour(x_seq, y_seq, log_ok_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```



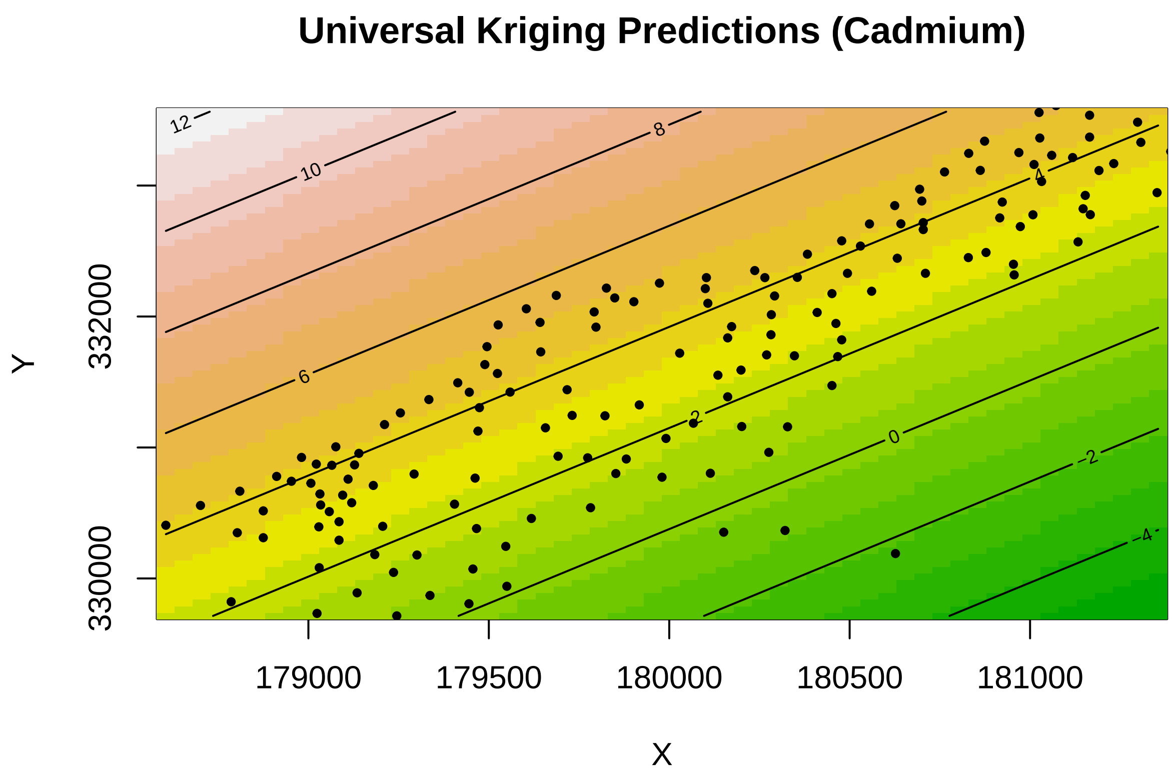

```r
# SK
sk_df <- as.data.frame(sk_pred)
sk_mat <- matrix(sk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, sk_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Simple Kriging Predictions (Cadmium)")
contour(x_seq, y_seq, sk_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```



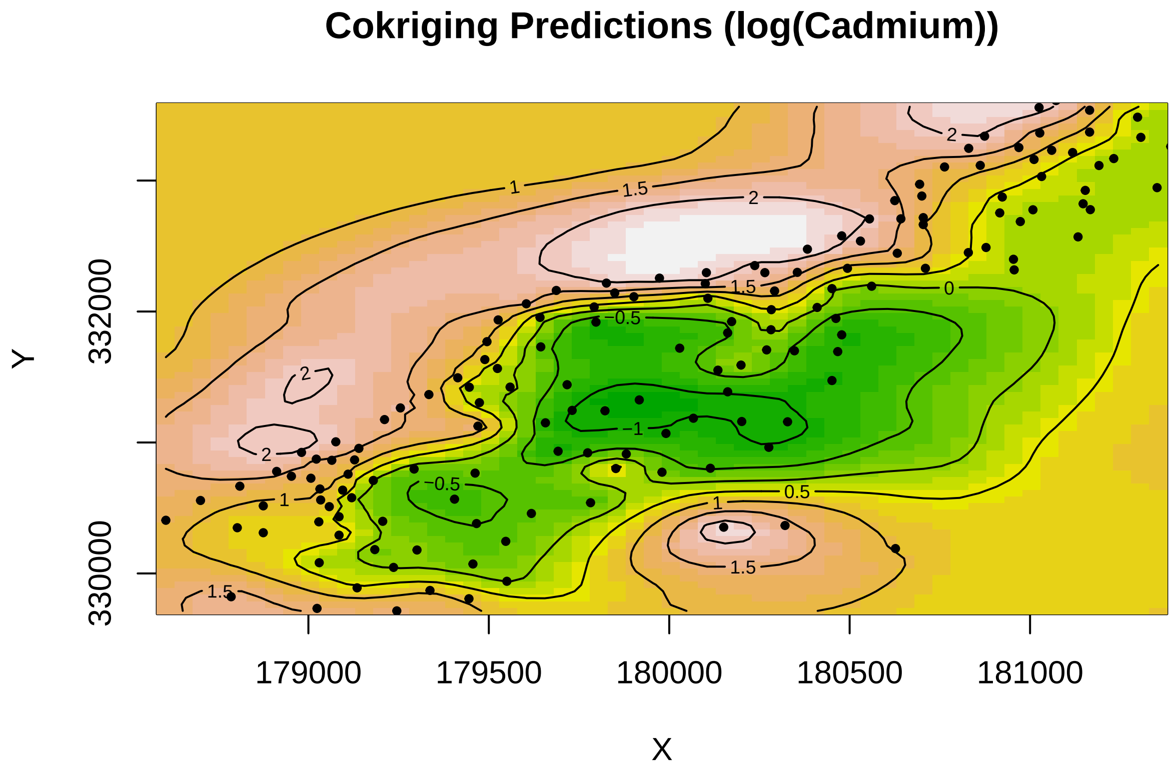

```r
# Log SK
log_sk_df <- as.data.frame(log_sk_pred)
log_sk_mat <- matrix(log_sk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, log_sk_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Log Simple Kriging Predictions(log(Cadmium))")
contour(x_seq, y_seq, log_sk_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```




```r
# UK
uk_df <- as.data.frame(uk_pred)
uk_mat <- matrix(uk_df$var1.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, uk_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Universal Kriging Predictions (Cadmium)")
contour(x_seq, y_seq, uk_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```




```r
# Cokriging
ck_df <- as.data.frame(ck_pred)
# head(ck_df)
ck_mat <- matrix(ck_df$log_cadmium.pred, nrow = length(x_seq), ncol = length(y_seq))
image(x_seq, y_seq, ck_mat,
      col = terrain.colors(20),
      xlab = "X", ylab = "Y",
      main = "Cokriging Predictions (log(Cadmium))")
contour(x_seq, y_seq, ck_mat, add = TRUE, nlevels = 10)
points(a6$x, a6$y, pch = 19, cex = 0.5)
```


Cokriging Predictions (log(Cadmium))
Y
332000
330000
X
179000
179500
180000
180500
181000
2
1.5
1
0.5
0
−0.5
−1