Geostatistics Athens Week project

SD Team

November 17, 2021

Contents

1 The Jura data set

The Jura data set comprises seven heavy metals measured in the top soil of the swiss Jura, along with consistently coded land use and rock type factors, as well as geographic coordinates.

Variable description :

- Xloc: X coordinate, local grid km
- Yloc: Y coordinate, local grid km
- Landuse: Land use: 1: Forest, 2: Pasture (Weide(land), Wiese, Grasland), 3: Meadow (Wiese, Flur, Matte, Anger), 4: Tillage (Ackerland, bestelltes Land)
- Rock: Rock Types: 1: Argovian, 2: Kimmeridgian, 3: Sequanian, 4: Portlandian, 5: Quaternary.
- Cd: mg cadmium kgˆ-1 topsoil
- Co: mg cobalt kgˆ-1 topsoil
- Cr: mg chromium kgˆ-1 topsoil
- Cu: mg copper kgˆ-1 topsoil
- Ni: mg nickel kgˆ-1 topsoil
- Pb: mg lead kgˆ-1 topsoil
- Zn: mg zinc kgˆ-1 topsoil

You are given three different files:

- jura_pred.csv: learning dataset
- jura_grid.csv: prediction grid (contains locations and covariables)
- jura_val_loc: validation locations and covariables

2 Exploratory analysis

```
rm(list=ls()) #Clean the working directory
```

2.1 Basic statistics

1. load the dataset from jura_pred.csv (on the cloud)

```
#Data set
jura = read.csv("../../jura/jura_pred.csv")
```

2. What is the class of the dataset?

```
class(jura)
```

```
## [1] "data.frame"
```

3. What is the number of observations? What is the number of variables?

```
str(jura) #e.g.
```

```
## 'data.frame':    259 obs. of  11 variables:
##  $ Xloc   : num  2.39 2.54 2.81 4.31 4.38 ...
##  $ Yloc   : num  3.08 1.97 3.35 1.93 1.08 ...
##  $ Landuse: int  3 2 2 3 3 3 3 3 3 3 ...
##  $ Rock   : int  3 2 3 2 5 5 5 1 1 3 ...
##  $ Cd     : num  1.74 1.33 1.61 2.15 1.56 ...
##  $ Co     : num  9.32 10 10.6 11.92 16.32 ...
##  $ Cr     : num  38.3 40.2 47 43.5 38.5 ...
##  $ Cu     : num  25.72 24.76 8.88 22.7 34.32 ...
##  $ Ni     : num  21.3 29.7 21.4 29.7 26.2 ...
##  $ Pb     : num  77.4 77.9 30.8 56.4 66.4 ...
##  $ Zn     : num  92.6 73.6 64.8 90 88.4 ...
```

4. Print the name of the variables.

```
names(jura)
```

```
##  [1] "Xloc"    "Yloc"    "Landuse" "Rock"    "Cd"      "Co"      "Cr"
##  [8] "Cu"      "Ni"      "Pb"      "Zn"
```

5. Compute the minimum and maximum value for each coordinate.

```
min(jura$Xloc);min(jura$Yloc)
```

```
## [1] 0.626
```

```
## [1] 0.58
```

```
apply(jura[,1:2],2,min)
```

```
##  Xloc  Yloc
## 0.626 0.580
```

```
range(jura$Xloc)
```

```
## [1] 0.626 4.920
```

```
max(jura$Xloc);max(jura$Yloc)
```

```
## [1] 4.92
```

```
## [1] 5.69
```

```
apply(jura[,1:2],2,max)
```

```
## Xloc Yloc
## 4.92 5.69
```

```
range(jura$Yloc)
```

```
## [1] 0.58 5.69
```

6. Compute basic statistics for the seven different heavy metals (mean, min, max, quartiles and standard deviation)

```
summary(jura[,5:11])
```

```
##        Cd                Co               Cr              Cu        
##  Min.   :0.1350   Min.   : 1.552   Min.   : 8.72   Min.   :  3.96  
##  1st Qu.:0.6375   1st Qu.: 6.520   1st Qu.:27.44   1st Qu.: 11.02  
##  Median :1.0700   Median : 9.760   Median :34.84   Median : 17.60  
##  Mean   :1.3091   Mean   : 9.303   Mean   :35.07   Mean   : 23.73  
##  3rd Qu.:1.7150   3rd Qu.:11.980   3rd Qu.:42.22   3rd Qu.: 27.82  
##  Max.   :5.1290   Max.   :17.720   Max.   :67.60   Max.   :166.40  
##        Ni              Pb               Zn        
##  Min.   : 4.20   Min.   : 18.96   Min.   : 25.20  
##  1st Qu.:13.80   1st Qu.: 36.52   1st Qu.: 55.00  
##  Median :20.56   Median : 46.40   Median : 73.56  
##  Mean   :19.73   Mean   : 53.92   Mean   : 75.08  
##  3rd Qu.:25.42   3rd Qu.: 60.40   3rd Qu.: 89.92  
##  Max.   :53.20   Max.   :229.56   Max.   :219.32
```

```
apply(jura[,5:11],2,sd)
```

```
##         Cd         Co         Cr         Cu         Ni         Pb         Zn 
##  0.9151877  3.5760463 10.9575090 20.7126345  8.2328582 29.7921582 29.0192912
```

7. Compute the mean of cobalt concentration for the four different landuses

```
mean(jura[jura$Landuse==1,6],na.rm=T)
```

```
## [1] 7.694545
```

```
mean(jura[jura$Landuse==2,6],na.rm=T)
```

```
## [1] 10.06429
```

```
mean(jura[jura$Landuse==3,6],na.rm=T)
```

```
## [1] 9.393867
```

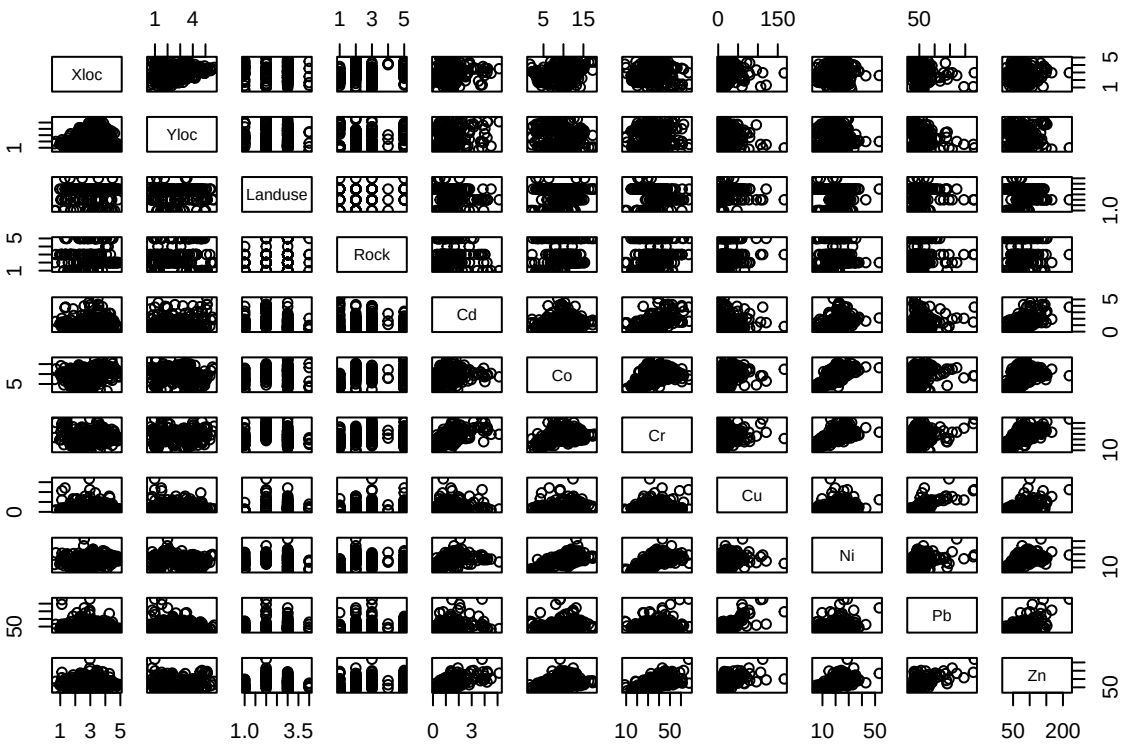
```
mean(jura[jura$Landuse==4,6],na.rm=T)
```

```
## [1] 8.372
```

2.1.1 Graphical Representations

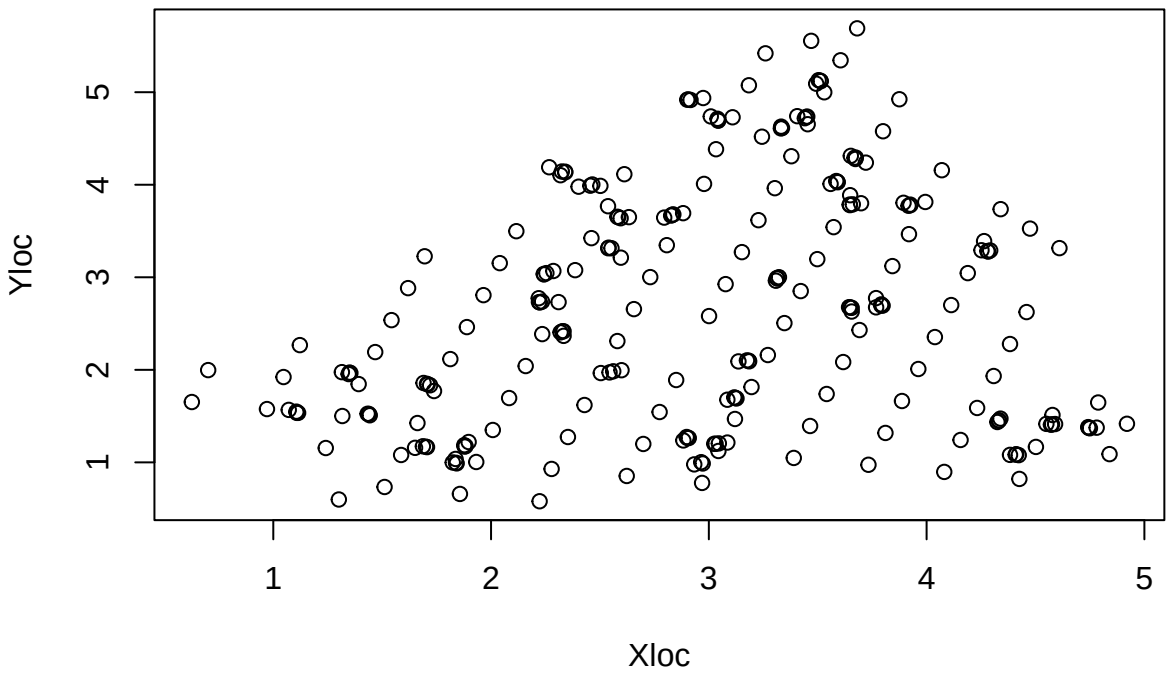
1. Apply the plot function to the whole dataset

```
plot(jura)
```



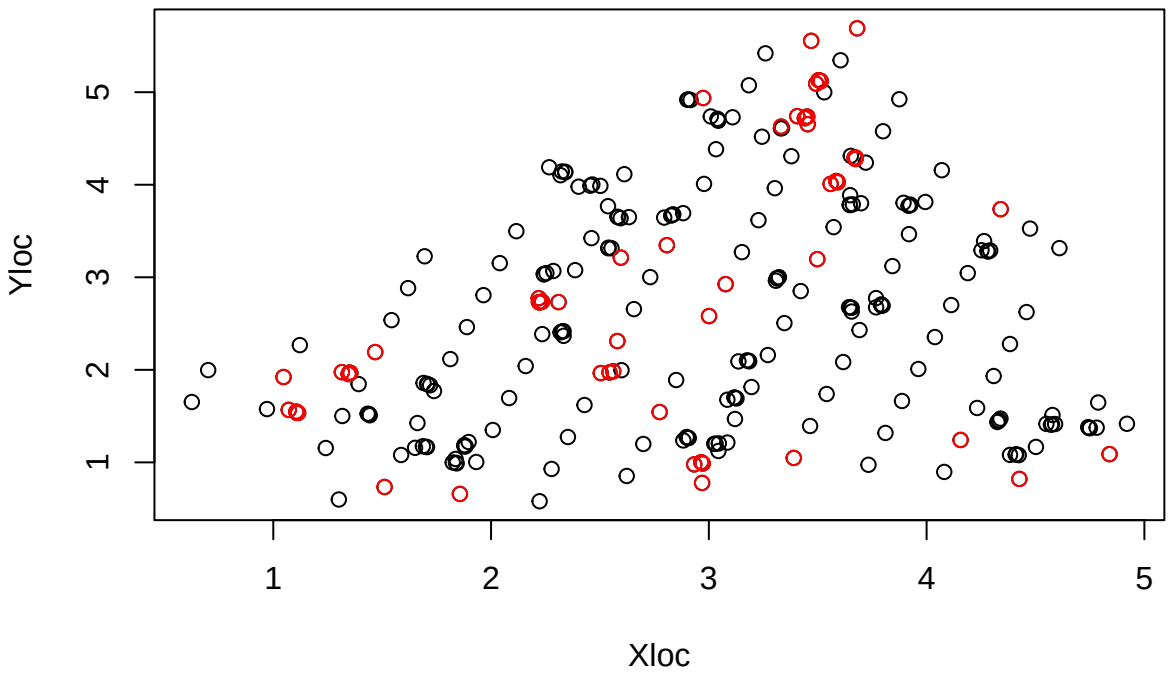
2. Plot the coordinates.

```
plot(jura[,1:2])
```



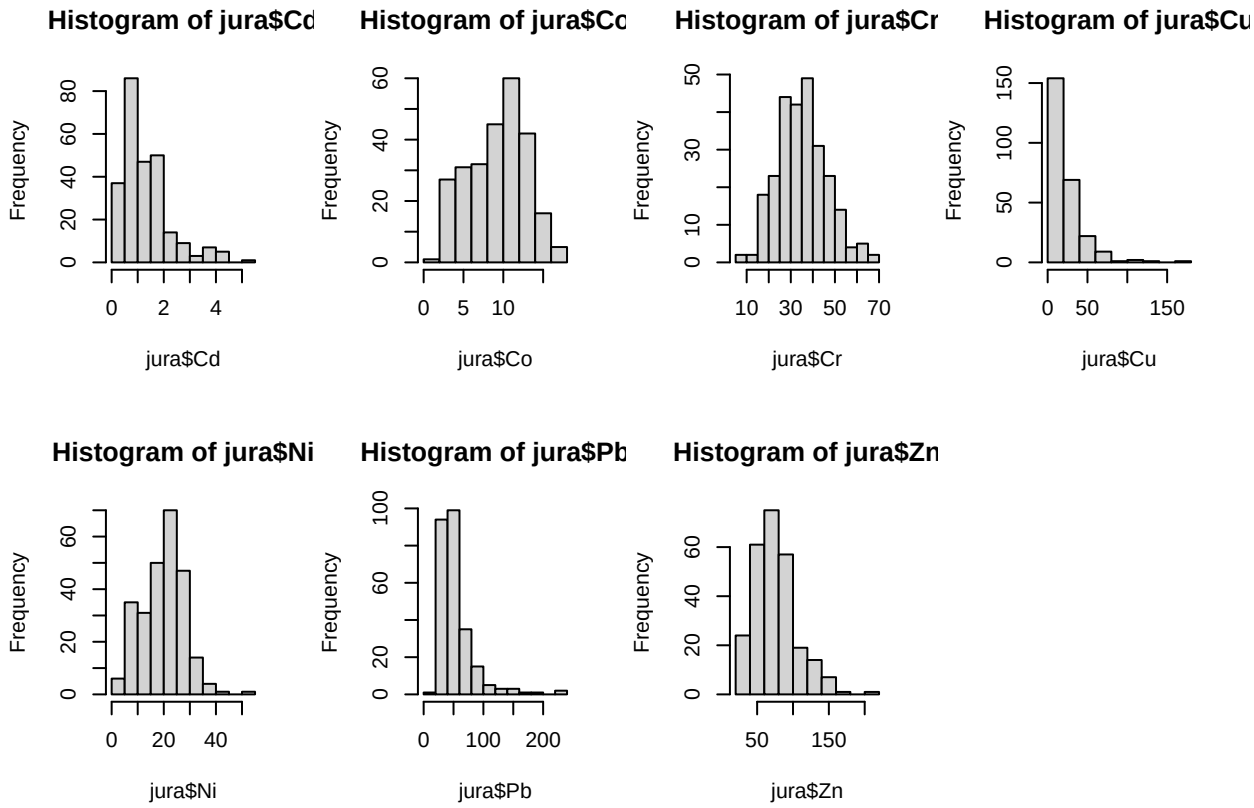
3. On the same plot, display the points with landuse 2 (pasture) in red.

```
plot(jura[,1:2])
points(jura[jura$Landuse==2,1:2],col='red')
```



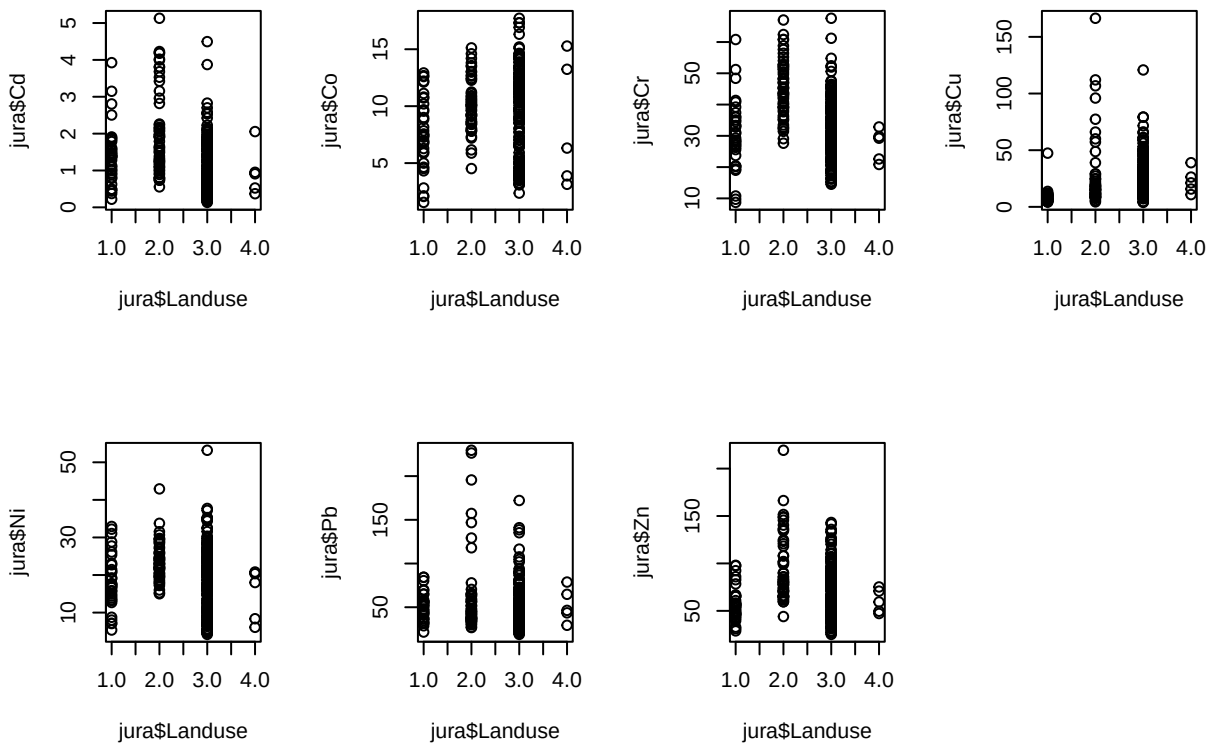
4. Plot the seven, heavy metal concentrations histograms.

```
par(mfrow=c(2,4)) # on a grid of 2x4 plots
hist(jura$Cd)
hist(jura$Co)
hist(jura$Cr)
hist(jura$Cu)
hist(jura$Ni)
hist(jura$Pb)
hist(jura$Zn)
```



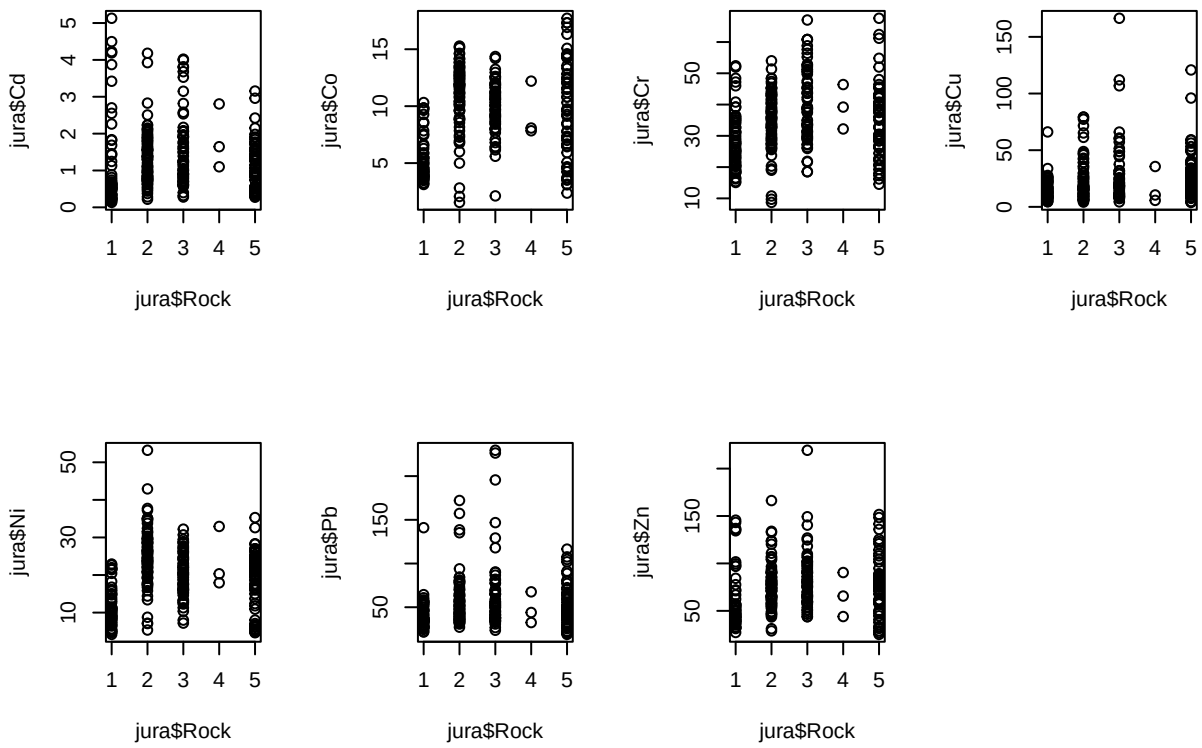
5. Plot the seven heavy metal concentrations as functions of the landuse

```
par(mfrow=c(2,4))
plot(jura$Landuse,jura$Cd)
plot(jura$Landuse,jura$Co)
plot(jura$Landuse,jura$Cr)
plot(jura$Landuse,jura$Cu)
plot(jura$Landuse,jura$Ni)
plot(jura$Landuse,jura$Pb)
plot(jura$Landuse,jura$Zn)
```



6. Plot the seven heavy metal concentrations as functions of the rocktype

```
par(mfrow=c(2,4))
plot(jura$Rock,jura$Cd)
plot(jura$Rock,jura$Co)
plot(jura$Rock,jura$Cr)
plot(jura$Rock,jura$Cu)
plot(jura$Rock,jura$Ni)
plot(jura$Rock,jura$Pb)
plot(jura$Rock,jura$Zn)
```



2.1.2 Some statistics

1. Transform the variables Landuse and Rock into factors (see ?as.factors).

```
jura$Landuse = as.factor(jura$Landuse)
jura$Rock = as.factor(jura$Rock)
```

2. Use the function aov to compute the analysis of variance of the cobalt concentrations with Landuse, Rock and their product as factors.

```
aov.co = aov(Co~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.co)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3  123.5   41.18   5.719  0.00085 ***
## Rock           4 1154.1  288.53  40.078  < 2e-16 ***
## Landuse:Rock  10  286.7   28.67   3.982 4.72e-05 ***
## Residuals    241 1735.0    7.20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3. Do the same on the other concentrations (check the histograms prior to apply a transformation if necessary).

```
aov.cd = aov(Cd~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.cd)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3  39.57  13.191  21.379 2.59e-12 ***
## Rock           4   0.84   0.209   0.339    0.851
## Landuse:Rock  10  26.99   2.699   4.375 1.20e-05 ***
## Residuals    241 148.69   0.617
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.cr = aov(Cr~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.cr)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3   7526  2508.5  32.317  < 2e-16 ***
## Rock           4   2044   510.9   6.582 4.83e-05 ***
## Landuse:Rock  10   2701   270.1   3.480 0.000268 ***
## Residuals    241  18707    77.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.cu = aov(Cu~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.cu)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3   8154  2717.9   6.985 0.000159 ***
## Rock           4   3774   943.5   2.425 0.048768 *
## Landuse:Rock  10   4982   498.2   1.280 0.242080
## Residuals    241  93776   389.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.ni = aov(Ni~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.ni)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3   1188   396.1  11.271 6.04e-07 ***
## Rock           4   5765  1441.4  41.015  < 2e-16 ***
## Landuse:Rock  10   2064   206.4   5.874 6.23e-08 ***
## Residuals    241   8469    35.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.pb = aov(Pb~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.pb)
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## Landuse        3   4465  1488.2   1.749 0.15764
## Rock           4  11722  2930.5   3.444 0.00927 **
```

```
## Landuse:Rock  10   7736   773.6   0.909 0.52532
## Residuals    241 205071   850.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.zn = aov(Zn~Landuse+Rock+Landuse*Rock,data=jura)
summary(aov.zn)
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## Landuse        3  31159   10386  16.840 5.79e-10 ***
## Rock           4  11417    2854   4.628  0.00129 **
## Landuse:Rock  10  26051    2605   4.224 2.03e-05 ***
## Residuals    241 148640     617
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3 Utilities

Load RGeostats.

```
library(RGeostats)
constant.define("asp",1) #environment variable for the scale ratio between axes
```

Load the data set and the prediction grid

```
#Data set
jura_tot = read.csv("../jura/jura_pred.csv")
#Prediction grid
grid = read.csv("../jura/jura_grid.csv")
names(jura_tot)
names(grid)
```

Coding of the factors

```
jura_tot$Landuse = as.factor(jura_tot$Landuse)
jura_tot$Rock = as.factor(jura_tot$Rock)

grid$Landuse = as.factor(grid$Landuse)
grid$Rock = as.factor(grid$Rock)
```

Change the names of the modalities to be consistent with their names on the grid

```
levels(jura_tot$Landuse)
levels(grid$Landuse)
levels(jura_tot$Landuse)=c("Forest","Pasture","Meadow","Tillage")

levels(jura_tot$Rock)
levels(grid$Rock)
levels(jura_tot$Rock)=c("Argovian","Kimmeridgian","Sequanian","Portlandian","Quaternary")
```

Separate the data set in two sets : the training set and the validation set. For the project, you should use the full data set for the training. You submit the prediction on Kaggle for a set of locations on which you only know the locations and the factors.

```
set.seed(1234) #Set the seed of the random generators

ntot = nrow(jura_tot)
ntrain = 200
nval = ntot - ntrain

indtrain = sample(ntot,ntrain)
indval = setdiff(1:ntot,indtrain)

jura = jura_tot[indtrain,]
val_loc = jura_tot[indval,1:4]

#val contains the values to predict. For the project, these values will be on Kaggle
#(for other locations) and you won't know them
#You will have the locations and covariables at the unknown locations by the following command :

#val_loc = read.csv("jura/jura_val_loc.csv")

val = cbind(1:nval,jura_tot[indval,]$Co)
```

RGeostats target grid and utilities for display

```
gridtemp = db.create(grid)
gridtemp = db.locate(gridtemp,c("Xloc","Yloc"),"x")
nx = c(length(unique(grid[,1])),length(unique(grid[,2])))
gridrg = db.grid.init(gridtemp,nodes=nx)
gridrg = migrate(gridtemp,gridrg,names=4:gridtemp$natt,radix="")
gridrg = db.rename(gridrg,2:3,c("Xloc","Yloc"))
gridrg = db.sel(gridrg,!is.na(gridrg["Landuse"])&!is.na(gridrg["Rock"]))

add.variable =function(var,grid,varname,gridtemp.=gridtemp)
{
  tt=db.add(gridtemp.,var)
  tt=db.rename(tt,tt$natt,varname)
  grid = migrate(tt,grid,names=tt$natt,radix="")
  grid
}
```

4 Interpolation exercise

Provide the maps of the cobalt concentration over the Swiss Jura obtained with several regression/interpolation methods, e.g.:

- anova
- linear regression on the coordinates
- local polynomial regression
- Nearest neighbour
- Inverse distance

- ...

1. Prediction by the mean

```
mean((val[,2]-mean(val[,2]))^2)
```

2. ANOVA

```
#Fit the anova model
aov.co = aov(Co~Landuse+Rock,data=jura)
summary(aov.co)

#Prediction on the validation locations
res.aov.val=predict(aov.co,val_loc)

##
#Compute score
mean((res.aov.val-val[,2])^2)


#Prediction on the grid
res.aov=predict(aov.co,grid)

##
#Add to the RGeostats db and display
gridrg = add.variable(res.aov,gridrg,"aov.predict")
plot(gridrg,pos.legend=1)
```

3. Linear regression of functions of coordinates

```
#Fit the linear model

trend = lm(Co~Xloc+Yloc+Xloc*Yloc+I(Xloc^2)+I(Yloc^2),data=jura)
summary(trend)

#Prediction on the validation locations
res.trend.val=predict(trend,val_loc)

##
#Compute score
mean((res.trend.val-val[,2])^2)
##

#Prediction on the grid
res.trend=predict(trend,grid)

#Add to the RGeostats db and display
gridrg = add.variable(res.trend,gridrg,"trend.predict")
plot(gridrg,pos.legend=1)
```

4. N-Nearest neighbours

```
ns =  3 #number of samples to consider for the prediction
datrg = db.create(jura)
datrg = db.locate(datrg,2:3,"x")
datrg = db.locate(datrg,7,"z")
m = model.create(1)
neigh=neigh.create(radius=100,nmini=ns,nmaxi=ns)


###
valrg=db.create(val_loc)
valrg=db.locate(valrg,2:3,"x")
res.val = kriging(datrg,valrg,m,neigh)
mean((res.val[,6]-val[,2])^2,na.rm=T)
###


res = kriging(datrg,gridrg,m,neigh)
plot(res)
```

5. Inverse distance

```
degree = 4
datrg = db.create(jura)
datrg = db.locate(datrg,2:3,"x")
datrg = db.locate(datrg,7,"z")


###
valrg=db.create(val_loc)
valrg=db.locate(valrg,2:3,"x")
res.val = invdist(datrg,valrg,exponent = degree)
mean((res.val[,6]-val[,2])^2)

res = invdist(datrg,gridrg)
plot(res)
```

6. Improve the prediction by using other parameterizations for the previous methods.
7. Use other methods (Local Polynomial Regression, Random Forests, ...).

5 Univariate analysis

5.1 Variography

5.1.1 Experimental variogram (isotropic case)

1. Compute and plot the experimental variogram of the cobalt concentration (using `vario.calc()`). Try different values of lag and comment the results.
2. Print the number of pairs of points used to compute the variogram values (using the options `npairpt=TRUE,npairdw=TRUE` in `plot()`) for different values of lag and comment the results.

5.1.2 Experimental variogram (anisotropic case)

1. Compute and plot the variogram maps (using `vmap.calc()`) the Cobalt concentration order to check for anisotropies. Comment
2. Compute and plot directional variograms (according to the anisotropy directions determined with the maps).

5.1.3 Model adjustement

The function `melem.name()` gives the models available in RGeostats.

1. Adjust a model using the function `model.auto()` (isotropic and anisotropic cases) on experimental variograms and print the model caracteristics.
2. Try imposing different structures or combinations of structures.
3. Compare the models adjusted on the experimental variogram and the variogram map (using `vmap.auto()`).

5.2 Prediction

5.2.1 Ordinary Kriging

1. Compute and plot the ordinary kriging of the cobalt over the prediction grid. Plot the associated standard deviation map.

2. Try several variogram models (basic structures, anisotropy), and neighborood options. Compute the prediction scores. Comment the results.

5.2.2 Universal kriging

Use the indicators of the different levels of the factors (*Rock* and *Landuse*) as covariates to compute the universal kriging prediction.

Transformation of the Rock factor into indicators

```
indiccut = limits.create(mini=c(1,2,3),maxi=c(2,3,4))

jurarg_KU=db.indicator(jurarg,indiccut,name="Rock")
jurarg_KU = db.locate(jurarg_KU,"Indicator*","f")
jurarg_KU = db.locate(jurarg_KU,"Co","z")

gridrg_KU=db.indicator(gridrg,indiccut,name="Rock")
gridrg_KU = db.locate(gridrg_KU,"Indicator*","f")
gridrg_KU = db.locate(gridrg_KU,"Co","z")

val_locrg_KU=db.indicator(val_locrg,indiccut,name="Rock")
val_locrg_KU = db.locate(val_locrg_KU,"Indicator*","f")
val_locrg_KU = db.locate(val_locrg_KU,"Co","z")
```

Variogram of the residuals

```
drift = c("1","f1","f2","f3")
vres = vario.calc(jurarg_KU,nlag=10,uc=drift)
plot(v)
plot(vres,add=T,col=2)
mres=model.auto(vres,struct=c(1,2))
```

Universal kriging on the grid

```
neigh=neigh.create(type=0)
res_gridKU=kriging(jurarg_KU,gridrg_KU,mres,neigh,uc=drift)
plot(res_gridKU)
```

Plot the associated standard deviation map.

Universal kriging on the validation set and prediction score

```
res_valKU=kriging(jurarg_KU,val_locrg_KU,mres,neigh,uc=drift)
mean((res_valKU[,"Kriging*estim"]-val[,2])^2)
```

1. Add the *Landuse* predictor to the model.
2. Try several variogram models (basic structures, anisotropy), and neighborood options. Compare the prediction scores. Comment the results.

Optional:

You can do the same with the interaction. Hint: consider the product *Landuse* x *Rock* as a new factor. You will have to group some of the levels so as to have well balanced groups. See the functions *replicates* and *TukeyHSD*.

6 Multivariate analysis

6.1 Variography

1. Compute the empirical directional variograms and covariograms of a carefully chosen (justify) set of variables.

What would you conclude about anisotropy ?

2. Fit a model (with *model.auto*).

6.2 Prediction

1. Interpolate *Co* on the grid using Ordinary Cokriging (function *kriging*) and plot the resulting map as well as the standard deviation map. Compute the prediction score.
2. You can also try to implement the universal cokriging (*optional*).

7 Maximum Likelihood estimation

1. Compute the maximum likelihood estimator of the parameters of (some of) your favorite univariate model(s) for the Cobalt concentration. In particular, to improve the prediction, add the explanatory variable *Landuse* to the model and estimate its parameters by maximum likelihood.
2. Compare the models with and without *Landuse* through a likelihood ratio test.
3. Compute the prediction map and the prediction at the validation locations for each model. Compute the prediction score.
4. Try the **SPBayes** package for the multivariate approach (*optional*)).

8 Bayesian (*optional*)

1. To improve the prediction, add the explanatory variables to the model and estimate the posterior distribution of the parameters.
2. The Bayesian approach is available in the multivariate case in the package **SPBayes**. Give it a try!

9 Conditional simulations

The information threshold for the concentration of cobalt in soils is *12 mg/kg*.

1. Generate 100 conditional simulations of the Cobalt concentrations over the swiss Jura according to your favorite model.

Example with the ordinary kriging (where mres is the fitted variogram of the cobalt)

```
#res_simu = simtub(jurarg_KU,gridrg_KU,mres,neigh,nbsimu=100,nbtuba=1000)
```

2. Compute the mean surface of the area of exceedance as well as its associated centered 95% confidence interval.

```
#res_simu[,"Simu.Co.S*"] = res_simu[,"Simu.Co.S*"]>12
#res_simu[,"Simu.Co.S1"] <- rowMeans(res_simu[,"Simu.Co.S*"])
```

3. Compute and plot the exceedance probability map. Comment.

```
#plot(res_simu,name="Simu.Co.S1",pos.legend=1)
```

10 Summary – Discussion

11 Appendix: description of the predictions submitted on kaggle (models, parameters) and corresponding prediction maps.