Comparison of universal and co-kriging of protein percent in wheat fields in Montana

Paul Hegedus
Montana State University
LRES 535

April 15, 2018

Table of Contents

Introduction

Precision agriculture aids sustainability of agricultural systems by creating more efficient systems that maximize net returns to the farmer through small scale management. Precision agriculture comprises of site specific management that informs and supports decisions based off of information processing and data collection (Lowenberg-DeBoer and Swinton 1997). Site specific management can be used to decrease agricultural inputs while retaining high outputs, increasing financial net returns of farmers. Precision agriculture has been bolstered by the advent of technology that have increased the amount of information collected on farms. Over the past few decades yield monitors have become standard in combines, allowing for the production of fine scale maps of yields across fields. More recently protein sensors have entered the market, measuring protein content of grain in a similar fashion as yield.

The amount of protein in grain is of utmost interest to producers as the prices for grain are dictated by protein content. Every grain elevator has different standards, but the principle is the same, the more protein in wheat, the higher the price received for the grower, and the lower the protein content in the wheat, the more the price is docked. This creates a vast importance on not only the quantity of food produced by a grower, but the nutritional quality as well.

The goal of this project will be to use kriging to enhance the prediction of a non- linear regression model on protein content of winter wheat in order to more accurately calculate estimated net returns for a farmer. This calculation of net return will then inform management decisions on the application of variable nitrogen rates. The objectives of this study will be twofold; to compare the accuracy of the interpolation of protein using universal kriging and co-kriging, and subsequently to investigate differences in results of a predictive model using the two interpolation techniques.

The predictive protein model in this study will be one developed for the OFPE project that utilizes a variety of geographic parameters such as slope, elevation, and aspect, as well as nitrogen applied and previous yield and NDVI data (Lawrence 2015). The importance of interpolating between protein points is that with more observed data in a model to train with, the higher the probability the model will be able to replicate that reality. More data will increase the statistical power of the model and increase confidence in having more realistic and more accurate net return calculations.

By using kriging, more points will be available to use in a predictive model that will enhance the likelihood of making predictions similar to the observed values. The interpolated points will assist in matching the observed spread of the data and result in better predictions that are used in the calculation of net return to a farmer. With accurate predictions of net return, a farmer can make more informed decisions on their management for the next season. Interpolating protein points to enhance the predictive power of the model is necessary and useful because while having a high yield is always important, it is also important to have a high enough protein yield to make grade and not get docked down to a lower price because of a low protein content. In some cases, not making the protein cutoff results in a reduction of half the harvest's value. By providing more points that increases the predictive power of the model, farmers will be able to make more confident decisions on management to will maximize their net returns.

Literature Review

The first law of geography, that everything is related but things closer together are more related than things farther apart, is the underlying principle of spatial autocorrelation, or the phenomena when points that are spatially closer are more closely related than points farther away (Tobler 1970). The presumption of spatial autocorrelation is the pretense for theories upon which spatial interpolation depends. Interpolation is the prediction of a value at a location without a measurement by using values from measured locations. Interpolation is essentially a practice that quantifies spatial dependence and uses it to fill in "gaps" between data (Miller et al. 2007). There are a variety of techniques for interpolation that fall into two broad categories; deterministic and geostatistical methods. Deterministic methods simply use mathematical equations to predict values at unknown locations, while geostatistical methods contain a deterministic component, as well as a statistical piece to predict values at unknown locations (Bailey and Gatrell 1995). A common deterministic model is inverse distance weighting (IDW), uses a weighted average based upon distance from known value to the unknown value. Inverse distance weighting, like other deterministic models, is advantageous in its ability for easy automation in application, as it can be simply pertained to a spectrum of variables (Bailey and Gatrell 1995, Gotway et al. 1996). However, geostatistical methods, which use the trend of the values between known points to estimate values between points, are widely used, such as kriging or splines. While all approaches, whether deterministic or geostatistical, were built to achieve the same goal, the approaches can vary in many ways and one method may be more appropriate for a situation than another based upon characteristics of the data (Gotway et al. 1996, Weber and Englund, 1994). For example, IDW was an accurate method for interpolation of soil organic matter (SOM) due to lack of variation in the data, however was an inaccurate prediction for soil nitrate due to greater dataset variation (Gotway et al. 1996). Comparatively though, while each dataset is unique, across differing datasets kriging produced better interpolation results than IDW (Gotway et al. 1996, Warrick et al.1988). Spatial interpolation is common practice in agriculture science, as both IDW and kriging have been used in agriculture to map nutrient distributions in fields, to create grain yield maps, to interpolate soil properties, and for a variety of other reasons (Delcourt and DeBaerdemaeker 1994, Gomez-Gil et al. 2011, Robinson and Metternicht 2006, Stafford et al. 1996). However, as far as published literature goes, I am not aware of attempts so far to interpolate the percent protein of grain in a field with kriging or co-kriging, even though interpolation has been used extensively with grain yield.

As mentioned previously, kriging is an interpolation method used for spatial predictions using existing data to predict values at unknown data points. Kriging was initially created by a South African mining engineer named Krige and was further developed into the methods used today. Because kriging utilizes known values at existing points, it should be used only when said prior data is available, and a circumstance where predicting values between the existing points is appropriate; it would be inappropriate to interpolate incidences of a discrete event in between points because kriging is only useful for continuous events across a surface (Kleijnan 2009).

More specifically, kriging is a technique for interpolating spatial data that utilizes parametric linear regressions and stochastic processes (Xiao et al. 2017). Kriging relies on the assumption of spatial autocorrelation, as the relationship of values due to proximity is essential to interpolation between observed points. Kriging is a 'weighted moving average' that uses observed values in a semivariogram to generate a predicted value based off the correlation of data and distance (Robinson and Dietrich 2016, Robinson and Metternicht 2003, Miller et al. 2007, Eldeiry and Garcia 2010). Kriging becomes more accurate with the use of the appropriate semivariogram model because the process uses the model semivariogram to form statistical weights based off distance from the surrounding observed value (Gotway et al. 1996). The accuracy of the result of kriging also depends on study design; most importantly how the known points were sampled, with evidence that grid sampling schemes generate more accurate results than simple random sampling for mapping soil properties (Gotway et al. 1996, Haining 1990, Houlong et al. 2016). Validation of the results of interpolations can be achieved in multiple ways. A common practice is cross-validation, or leave-one-out validation, where each point is left out and predicted using the rest of the validation set (Rossiter 2012). Another method is through comparison of predicted values to a validation set, where the locations of the validation set are interpolated and evaluated against the observed values at that point (Kleijnen 2017, Rossiter 2012).

There are multiple types and methods to kriging, with the common goal to reduce the error variance of predictions (Wang and Shi 2017). For example, simple kriging assumes a constant distribution of mean values (stationarity) across the interpolated surface, while ordinary kriging recalculates the mean of the area based upon the location; both interpolate with kriging but use slightly different calculations (Wood and Miller 2016). The rejection of the assumption that the mean and distribution do not vary is one reason that ordinary kriging is the most common kriging method (Kleijnan and Mehdad 2014, Simpson et al. 2001). Kriging varies in complexity, from other relatively simple kriging methods like universal kriging, which differs only in that it takes into account a trend in the data, compared to more complex methods like regression kriging that implements a linear regression and interpolates residuals from a nonspatial model (Li et al. 2011, Goovaerts 1998, Eldeiry and Garcia 2010, Yufeng et al. 2011). Kriging (as well as IDW) is considered an "exact" method because the values that are interpolated do not exceed the range of known values. While the exactness of kriging does not hinder the usefulness of the method in appropriate situations, this assumption poses a potential flaw, as an interpolated point could potentially be the real maximum or minimum value if measured (Babak 2013, Hwang et al. 2012, Babak and Deutsch 2009, Rojas-Avellaneda and Silvan-Cardenas 2006, Saito et al. 2005, Liu et al. 2015). While kriging remains a popular and useful method of interpolation, another noted downside is the dependence on Gaussian distributions and nonlinear models when there are complex interactions that are better characterized by relational dependence between variables and non-Gaussian distributions (Gaetan and Guyon 2010, Remy et al. 2009, Chile's and Delfiner 1999, Liu et al. 2015). Due to the desired development of complexity, more geostatistical methods are available, including a variant of kriging; co-kriging. As noted in the introduction, this project will utilize universal kriging to compare the result of co-kriging with yield; not only because the universal flavor is a commonly used kriging method, but because across the same soil hydraulic property dataset it has produced the best predictions of multiple kriging methods for yield across fields (Wopereis et al. 1996).

Co- kriging will be used as the method in this project due to the ability to correlate multiple sets of data and because models using co- kriging have been found to decrease approximation error (Xiao et al. 2017). Co- kriging, as the name implies, is theoretically similar to kriging, except for using the values of two or more correlated variables for predictions of unknown locations, but evidence supports that co- kriging using an auxiliary variable may increase the accuracy of interpolated points (Emamgholizadeh et al. 2017, Goovaerts 1998, Wang et al. 2013, Li et al. 2011). Co- kriging has an ideal application in situations where the variable of interest has not been sampled as extensively as the auxiliary variable, and the co-variables have a correlated relationship, such as the potential relationship between grain yield and protein (Eldeiry and Garcia 2010, Wopereis et al. 1996). Because correlated data produces more accurate models, co-kriging is preceded by the investigation of correlations with regression models in order to highlight correlations between data that is appropriate for use as an auxiliary variable in a cross- variogram for co-kriging (Emamgholizadeh et al. 2017, Vauclin et al. 1983). Using co- kriging will be ideal for this study because of the yield and protein datasets that differ in sampling intensity but are potential associated co-variables.

Kriging and co- kriging are interdisciplinary spatial interpolation methods that have been extensively applied in agricultural science through the study and mapping of soils. The limited sampling of soils is due to high costs, the time-consuming nature of the samples, and a destructive sampling process that inhibits pure replication in studies. Soil samples are taken at an extremely low frequency compared to more simply sampled data such as yield points, which are collected using automated data loggers on most modern combines. Due to these difficulties in measuring the soil, necessary interpolation is achieved through kriging or related processes to capture the inherent variability in soils. Not only does the analysis and mapping of soils dominate the application of interpolation, and specifically kriging, in the agricultural discipline, but some of the more complex variations of kriging are used (Yufeng et al. 2011). Protein content is not measured as abundantly as yield points, yet interpolation has not been published. The more accurate interpolation becomes, the more useful this technique of spatial analysis becomes to other aspects of agricultural science. Interpolated points that accurately represent the behavior of variables may provide a more detailed understanding of in-field processes at finer scales with less sampling effort. Improvements in this field will promote more informed management decisions and enhance agricultural optimization.

Methods

Study Site

Data pre-collected from the On-Farm Precision Experiment out of Montana State University was used for this study. The data were from four fields and three different farms to average out the effect of environmental variables on interpolation quality (Figure 1). All of the three farms are located around Montana; in Rapeljie, Gallatin Gateway, and Loma. The data collected from each field included measurements such as; UTM coordinates, the type of crop in a given year, crop yield in that year, protein content in that year, the amount of nitrogen applied, the previous crop, previous crop yield, and environmental variables such as elevation, slope, aspect, topographic position index, NDVI from Landsat 8, precipitation and growing degree days.

There are two types of data that were collected for each field; yield and protein. Yield was measured on the combine with a grain flow sensor connected to a yield monitor that calculates a value (bushels per acre) every three seconds in the field. Each sampling point's coordinate is stored and associated with that point's yield to generate a yield map for the field. These maps are detailed due to the volume of points sampled across an entire field.

Protein is also measured with a sensor, though not as frequently as for yield. A protein sensor on the combine measures the grain protein for a point in the field at a regular interval. The coordinates of these locations are saved, and similar to yield, a protein map can be generated for the field. However, the protein maps are not as descriptive of the field and contain large gaps between points because protein measurements are collected dramatically farther apart than yield measurements. For example, a dataset for one of the study farms includes 19,945 points for yield, and only 828 points for protein data. When making predictions, stronger results tend to occur when there are more points to generate estimates from, increasing the power of more points in describing observed values.

Analysis

To compare interpolation techniques and to determine their effect on the results of a predictive model, protein percentages from each field was interpolated using two kriging methods. For each field, to initiate the kriging process, a linear regression was used to observe any spatial trends in the data (Kravchenko and Robertson 2007). In the case that trends were found, interpolation was done on the residuals. A semivariogram for protein in each field was generated and fit with a parametric model that characterized the semivariogram's trend (Cressie 1988, Goovaerts 1989, Eldeiry and Garcia 2010). Different parametric models were fitted to the semivariogram, and the model with the lowest AIC was used for the kriging process (Robertson 1987, Eldeiry and Garcia 2010). Whereas a power test is commonly executed with the data specific to each field before every kriging process in order to determine an adequate sample size, the grid of interpolation points will be generated in this study by matching the amount of observations in the yield dataset. The 1:1 ratio of interpolated points to yield points were to merge the interpolated protein data with the yield data so that results of the predictive model using each kriging method could be adequately compared.

The process of universal kriging is relatively similar to that of other non co-kriging methods; a semivariogram is made and fit with a model that is used in the interpolation calculation, and a grid is generated to represent the interpolation points of the kriging process. Universal kriging is a type of kriging that incorporates relational trends between variables and where a model is fit to the semivariogram of the residuals. A linear multiple regression approach was used to model the protein with covariates and a weighted average is computed as the predictor (Hengl et al. 2007, Draper and Smith 1981, Christensen 1996). These residuals are subsequently used to estimate the regression coefficients using a generalized least squares method (Hengl et al. 2007). The residuals of this model are then used to create a semivariogram to which a parametric model is fit the same way as for simple kriging (Eldeiry and Garcia 2010). The calculations and equations used for the regression kriging process used are outlined by Hengl et al. (2007, n.d.) and executed with the "gstat" package (Pebesma 2004, Benedikt et al. 2016) in this study.

Co- kriging is similar to universal kriging by representing trends between variables, however the correlation between variables in co-kriging is directly characterized in the kriging process as the variable of interest being dependent on the values of another variable (Stein et al. 1989). Ordinary co- kriging has been shown to increase precision of interpolation compared to ordinary kriging due to dependence on covariables, defined through distinct semivariogram definitions (Heriawan and Koike 2008, Moukana et al. 2013). After investigating the variable in the yield dataset with the highest correlation to protein, the semivariogram was produced (Moukana et al. 2013). Like the other kriging methods, a model was fit to this semivariogram and ordinary kriging was used to interpolate points on a grid that was built in the same way as defined before. Ordinary co- kriging was applied with the "gstat" package (Pebesma 2004, Benedikt et al. 2016) and uses the specific calculations defined in the common literature (Moukana et al. 2013, Stein et al. 1989, Liang et al. 2016, Vauclin et al. 1983).

As stated in the objectives, the interpolated results from the kriging methods were used in a predictive non-linear model to estimate protein yield. For each point in the field, the model estimates protein based upon the following equation;

(1)
$$P = \alpha + \frac{(\beta - \alpha) * N}{\frac{1}{\gamma} + N}$$

Where P is protein, α are the effects when N = 0, β are the effects at N = max, gamma is a slope factor, and N is the nitrogen applied. Parameters affecting α include environmental variables such as aspect, slope, elevation, and topographic position index while parameters affecting β includes NDVI from multiple years.

Validation

The first objective remains unexplored without validation of the interpolated results. To compare the results of the two interpolation techniques multiple validation methods will be used. The first will consist of using the kriged estimates to predict values at the locations of the validation set and calculate the root mean square error (RMSE), mean error, and coefficient of determination (R^2). The second method will involve cross-validation, a practical way to validate datasets by repeatedly leaving out an observation and interpolating the value of that point (Challenor 2013, Sila et al. 2017, Zhang and Wang 2010). The results of this process will also be used to calculate RMSE, mean error, and the mean squared deviation ratio (MSDR).

To achieve the second objective, the validation data was ran used in the predictive model and the AIC was recorded for each field. This was repeated with estimates from universally kriging and co-kriging on the validation set for comparison. To investigate differences in the results of the model between interpolation techniques, the AIC was compared for each field across the control and kriged results to analyze differences in model performance between techniques (Marusteri and Bacarea 2010, R Core Team 2016).

Results

dB4

For the field dB4, universal kriging performed better in all validation statistic compared to the results of co-kriging with yield (Table 1). The root mean square error (RMSE) for universal kriging was 1.460 and 5.820 for universal co-kriging. Both kriging methods resulted in predictions lower than observed values with the mean error for universal kriging was -0.184% compared to the -3.33% for co-kriging. The coefficient of determination for co-kriging was practically 0 while the predictions from universal kriging only represented 17% of the variance in the observed data.

The differences in the results from cross-validation of the two datasets were not as large as for the validation through comparing predictions to the validation dataset (Table 2). The RMSE for co-kriging was 0.003 better than for universal kriging, a difference between 1.377 and 1.374 respectively. Both had small mean errors, -.0.007% for universal kriging and 0.025% for co-kriging. The mean squared deviation ratio (MSDR) of was also slightly better for co- kriging (0.969) compared to 0.972 for universal kriging.

Using the same hyperbolic model and as many predicted points as there are observed values in the validation dataset, predictions from universal kriging resulted in lower AIC's than co-kriging and the observed values (Table 3).

dV1

Like dB4, universal kriging performed better than co-kriging for every validation statistic in dV1 (Table 1). The difference in RMSE was about the same magnitude as dB4; 1.14 for universal kriging and 5.14 for co-kriging with yield. Both mean errors for the methods were negative, with universal kriging underestimating protein percent by an average of -.134% and -2.534% with co- kriging. The coefficient of variation, or R^2, was pretty much 0 when co-kriging with yield while universal kriging explained 29.8% of the observed data variance.

Cross-validation also supports that universal kriging performed better for every statistic compared to co-kriging with yield (Table 2). The RMSE was 0.979 for universal kriging and 1.005 for co-kriging, and the MSDR was 0.992 for universal kriging and 1.047 with co-kriging. The mean error for both kriging methods were close to 0 with 0.002 for universal kriging and -0.0009 for co-kriging.

Similar to dB4, universal kriging resulted in the lowest AIC compared to using the observed data and co-kriging (Table 3).

dV2

The same trends were found for dV2 as for dB4, with all statistical metrics supporting that universal kriging outperformed co-kriging with yield (Table 1). Validation of the kriged methods with the validation dataset shows that universal kriging had a lower RMSE compared to co-kriging (1.086 UK vs. 7.273 CK) as well as mean error (0.175 UK vs. -3.992 CK). Again, co-kriging explained virtually none of the variance in the observed data (0.04%) compared to 23.8% explained with universal kriging.

Cross-validation showed a similar theme as comparison to the validation dataset, with universal kriging representing better representations of the observed values (Table 2). The RMSE for universal kriging was 1.126 compared to 4.824 for co-kriging, the mean error was

0.004 for universal kriging compared to 0.367 for co-kriging, and the MSDR was 1.056 for universal kriging and 1.149 for co-kriging.

Using universal kriging in the predictive wheat protein model resulted in the lowest AIC compared to using the observed values and the results from co-kriging (Table 3).

dW3

Again, universal kriging was supported as a better predictor compared to a validation dataset than co-kriging with yield (Table 1). The difference in RMSE was greatest for this field with 1.381 for universal kriging and 11.391 for co-kriging. Both methods produced negative mean errors, -0.076 for universal kriging and -1.55 for co-kriging. Universal kriging also explained more of the observed variance than co-kriging, 28.4% compared to 0.0027%.

The results of cross validation also were the same as the other fields with universal kriging producing a better predictor than co-kriging with yield (Table 2). The RMSE was 1.1248 for universal kriging and 9.434 with co-kriging, mean error was 0.0005 for universal kriging and -2.059 for co-kriging, and the MSDR was 1.034 for universal kriging and 57.791 for co-kriging.

Consistent with the results from the other three fields, AIC using predicted protein from universal kriging was lower than the AIC of the predictive model when using estimates from co-kriging and the observed values in the validation dataset.

Discussion

Universal kriging characterized the observed results more so than co-kriging, insinuating that although co-kriging uses information from a more densely sampled covariable, this did not result in better estimates of protein percent spatially. Universal kriging also had smaller mean errors compared to co-kriging, generally by a wide margin. The greater mean error from co-kriging has economic implications as errors of 2, or 3% are plenty to tip the balance between feed and food grade, representing very different prices received. These results were supported by both validation methods, a comparison to a validation set and through cross-validation, increasing confidence that universal kriging is a better predictor of protein percent. The outperformance of universal kriging compared to co-kriging means that even though co-kriging involves a parameter with a higher sampling density, a lack of a strong correlation resulted in a lower quality estimate compared to universal kriging.

Estimates with universal kriging also were shown to improve the quality of a predictive protein model, resulting in the lowest AIC compared to using estimates from co-kriging and the observed values in the validation dataset. In all the fields, this meant that the estimates from universally kriged models would be beneficial to use for calculations of projected net return for producers. The combination of improved model results and the ability to predict with more points by interpolation will enhance the usefulness of precision estimates for growers.

While being more closely correlated to protein than any other parameter except nitrogen, the low correlation between protein and yield seemed to have played a factor in the accuracy of co-kriging results. With the block study design of nitrogen management, the nitrogen rates are heterogeneously clustered while the rest of the field receives a uniform rate of fertilizer. Interpolating with this would be limited in quality to the area where the experiment is in the field as elsewhere in the field there is no variation between neighbors in the amount of nitrogen applied. If nitrogen rates were varied and randomly distributed throughout a field, this may be a more appropriate dataset to krige with. There is possibility for further investigation into whether universal kriging continues to yield more similar predictions to observed data over time and space than co-kriging. This study represented fields from three different areas of Montana, however a comparison between the arid climate of Montana and more humid areas such as the Midwest may yield different results, as well as if different crops besides winter wheat were compared.

The results from the comparison of universal and co-kriging support that interpolation of protein percent in winter wheat fields was most representative of observed values with universal kriging. In all four fields analyzed universal kriging had higher R^2 values and lower RMSE and mean errors compared to co-kriging, highlighting a consistency across time and space. The addition of yield as a co-variable with protein was not found to increase the interpolation accuracy or increase the quality of a predictive wheat model, suggesting that a more closely related covariable, such as nitrogen, would be beneficial to increasing interpolation performance and model quality.

Figures

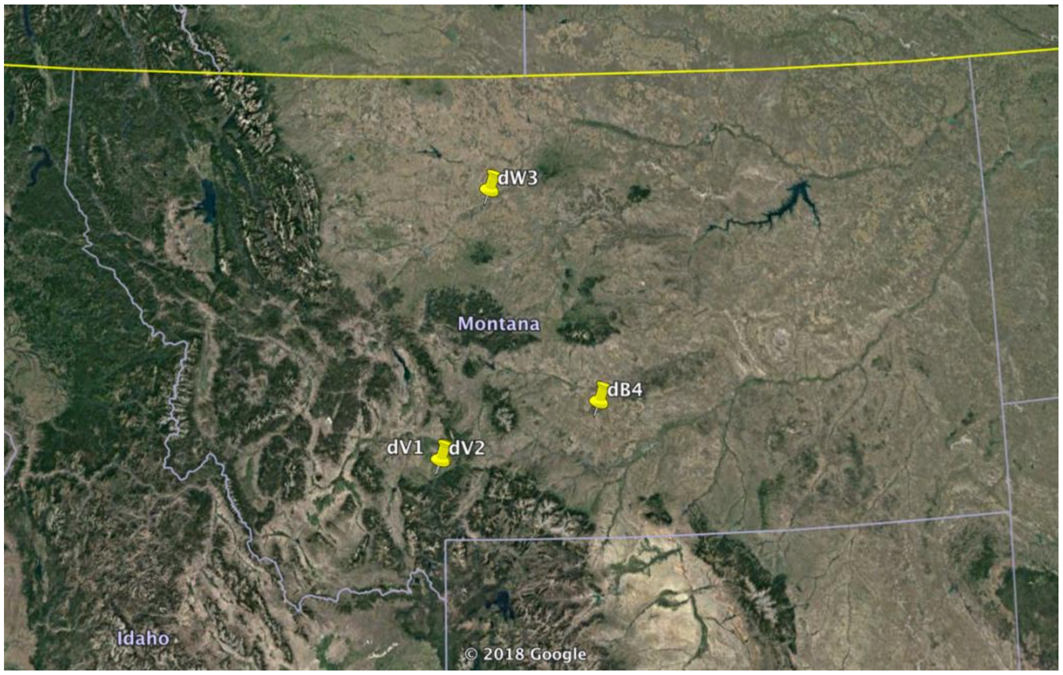

Figure 1. Map visualizing the location of the four fields

Validation

Field	Method	Metric		
		RMSE	*ME*	*R2*
dB4	UK	1.46	-0.184	0.171
	CK	5.82	-3.33	0.004
dV1	UK	1.14	-0.134	0.298
	CK	5.14	-2.534	3.45E-07
dV2	UK	1.086	0.175	0.238
	CK	7.273	-3.992	4E-04
dW3	UK	1.381	-0.076	0.284
	CK	11.391	-1.55	2.7E-04

Table 1. Results of three statistical measures from comparison of kriged values to validation dataset for each field

Cross Validate

Field	Method	Metric		
		RMSE	*ME*	*MSDR*
dB4	UK	1.377	-0.007	0.972
	CK	1.374	0.025	0.969
dV1	UK	0.979	0.002	0.992
	CK	1.005	-9E-04	1.047
dV2	UK	1.126	0.004	1.056
	CK	4.824	0.367	1.149
dW3	UK	1.248	5E-04	1.034
	CK	9.434	-2.059	57.791

Table 2. Cross validation results of three statistical measures for both kriging types for each field

Model Results	AIC		
Field	*NK*	*UK*	*CK*
dB4	613.529	368.515	1048.368
dV1	234.260	56.556	420.263
dV2	287.994	119.868	610.508
dW3	572.320	255.333	1218.519

Table 3. AIC results from predictive model, NK = no kriging (just validation dataset), UK = universal kriging, CK = co kriging

References

Babak, O. 2013. Inverse distance interpolation for facies modeling. Stoch Environ Res Risk Assess.

Babak, O. and C.V. Deutsch. 2009. Statistical approach to inverse distance interpolation. Stoch Environ Res Risk Assess. 23:543–553

Bailey, T.C. and Gatrell, A.C. (1995) Interactive Spatial Data Analysis.
Addison Wesley Longman, Harlow, Essex.

Chiles, J.P., and P. Delfiner. 1999. Geostatistics—modeling spatial uncertainty. Wiley, New York

Delcourt, H., and J. De Baerdemaeker. 1994. Soil nutrient mapping implications using GPS. Computers and Electronics in Agric. 11(1): 37-51.

Eldeiry, A.A., and L.A. Garcia. 2010. Comparison of ordinary kriging, regression kriging, and cokriging techniques to estimate soil salinity using LANDSAT images. Journal of Irrigation and Drainage Engineering. 146(6):355-364.

Gaetan, C., and X. Guyon. 2010. Spatial statistics and modeling. Springer, New York

Goovaerts P, 1998. Ordinary cokriging revisited. Mathematical Geology, 30(1): 21–42.

Gomez-Gil, J., Lopez-Lopez, L. J., Navas-Gracia, L.M., and G. Ruiz-Ruiz. 2011. The spatial low-pass filtering as an alternative to interpolation methods in the generation of combine harvester yield maps. Applied Engineering in Agriculture. 27(6): 1087-1097.

Gotway, C.A., Ferguson, R.B., Hergert, G.W., and T.A. Peterson. 1996. Comparison of kriging and inverse-distance methods for mapping soil parameters. Soil Science Society of America. 60:1237-1247.

Haining, R. 1990. Spatial data analysis in the social and environmental sciences. Cambridge University Press. 409.

Houlong, J., Daibin, W., Chen, X., Shuduan, L., Hongfeng, W., Chao, Y., Najia, L., Yiyin, C. and G. Lina. Comparison of kriging interpolation precision between grid sampling scheme and simple random sampling scheme for precision agriculture. Eurasian Journal of Soil Science. 5(1):62-73.

Hwang, Y., Clark, M., Rajagopalan, B., and G. Leavesley .2012. Spatial interpolation schemes of daily precipitation for hydrologic modeling. Stoch Environ Res Risk Assess. 26:295–320

Kleijnen, J.P.C. 2009. Kriging metamodeling in simulation: a review. European Journal of Operational Research. 192:707-716.

Kleijnen, J.P.C., and E. Mehdad. 2014. Multivariate versus univariate kriging metamodels for multi-response simulation models. European Journal of Operation Research. 236: 573-582.

Lawrence, P.G. 2015. Resilience of Montana's agroecosystems to economic and climatic change. Dissertation. Montana State University. LRES.

Li, X., Tong, L. Kang, S. and F. Li. 2011. Comparison of spatial interpolation methods for yield response factor of winter wheat and its spatial distribution in Haihe basin of north China. Irrig. Sci. 29:455-468.

Liu, S., Anh, V., McGree, J., Kozan, E., and R.C. Wolff. A new approach to spatial data interpolation using higher order statistics. Stoch Environ Res Risk Assess. 29:1679-1690.
Lowenberg-DeBoer, J., Swinton, S.M. 1997. Economics of site specific management in agronomic crops. In: Pierce, F.J., Sadler, E.J. (Eds.), The State of Site-specific Management for Agriculture. ASA, CSSA, SSSA. 369–396.

Miller, J., Franklin, J., and R. Aspinali. 2007. Incorporating spatial dependence in predictive vegetation models. Ecological Modeling. 202(3): 225-242.

Remy, N., Boucher, A., and J. Wu. 2009. Applied geostatistics with SGeMs: a users's guide. Cambridge University Press. Cambridge

Robinson, M., and S. Dietrich. 2016. An introduction to spatial autocorrelation and kriging. University of Alberta. *RenR 690:* 13-20.

Robinson, T. P., and G. Metternicht. 2006. Testing the performance of spatial interpolation techniques for mapping soil properties. Computers and Electronics in Agric. 50(2): 97-108.

Robinson, T. P., and G. Metternicht. 2003. A Comparison of Inverse Distance Weighting and Ordinary Kriging for Characterising within-paddock Spatial Variability of Soil Properties in Western Australia, Cartography, 32(1): 11-24.

Rojas-Avellaneda, D., and J.L. Silvan-Cardenas. 2006. Performance of geostatistical interpolation methods for modeling sampled data with non-stationary mean. Stoch Environ Res Risk Assess. 20:455–467

Rossiter, D. G. 2012. Technical Note: Co-kriging with gstat of the R environment for statistical computing. University of Twente. 74pp.

Saito, H., McKenna, S.A., Zimmerman, D.A., and T.C. Coburn. 2005. Geostatistical interpolation of object counts collected from multiple strip transects: ordinary kriging versus finite domain kriging. Stoch Environ Res Risk Assess. 19:71–85

Simpson, T.W., Peplinski, J.D., Koch, P.N., and J.K. Allen. 2001. Metamodels for computer-based engineering design: survey and recommendations. Engineering with Computers. 17: 129-150.

Stafford, J. V., B. Ambler, R. M. Lark, and J. Catt. 1996. Mapping and interpreting the yield variation in cereal crops. Computer and Electronics in Agric. 14(2-3): 101-119

Tobler, W.R.. 1970. A computer movie simulating urban growth in the Detroit region. Economic Geography. 46: 234-240.

Vauclin, M., Viera, S.R., Vachaud, G., and D.R. Nielson. 1983. The use of Cokriging with limited field soil observations. Soil Science Society of America Journal. 47(2):175-184.

Xiao, M., Zhang, G., Breitkopf, P., Villon, P., and W. Zhange. 2017. Extended Co-Kriging interpolation method based on multi-fidelity data. Applied Mathematics and Computation. 323: 120-131.

Wang, Z., and W. Shi. 2017. Mapping soil particle-size fractions: A comparison of compositional kriging and log-ratio kriging. Journal of Hydrology. 546: 526-541.

Wang K, Zhang C R, Li W D, 2013. Predictive mapping of soil total nitrogen at a regional scale: a comparison between geographically weighted regression and cokriging. Applied Geography. 42: 73–85.

Warrick, A.W., Zhang, R., El-Harris, M.K., and D.E. Myers. 1988. Direct comparisons between kriging and other interpolators. p. 505-515. In Wierenga, P.J., and D. Bachelet. Proc. Validation of flow and transport models for the unsaturated zone. New Mexico State University.

Weber, D.D., and E.J. Englung. 1994. Evaluation and comparison of spatial interpolators. Mathematical Geology. 24:589-602.

Wood, C., and B. Miller. 2016. Comparing simple and ordinary kriging methods for 2015 Iowa precipitation. Iowa State University.

Wopereis, M.C.S., Stein, A., Kropff, M.J., and Bouma, J. Spatial interpolation of soil hydraulic properties and simulated rice yield. Soil Use and Management. 12:158-166.

Yufeng, G.E., Thomasson, J.A., Sui, R., and J. Wooten. Regression-kriging for characterizing soils with remote sensing data. Front. Earth Sci. 5(3):239-244.

Appendix

```
#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
## LRES 535
## Project Script
## Author: Paul Hegedus
## Date: April 2018

## Script runs ordinary, co-kriging, and residual kriging methods on field and their
## performance in a predictive yield model are compared using AIC.
#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

#************************************ set up  *******************************************
#*************************************************************************************
## install packages
library(sp)
library(gstat)
library(spatstat)
library(maptools)
library(rgdal)
library(GISTools)
library(raster)
library(dplyr)
library(ggplot2)
library(ggpubr)

## field info (for reference)
fields <- data.frame(farmer=c("broyles","vandyke","vandyke","wood"),
                fields=c("sec35west","portnell1south","davidsonmidwest2","henrys"),
                code=c("dB4","dV1","dV2","dW3"),
                year=c("2017","2016","2016","2016"))

#============================== bring in field data ===================================
## Broyles sec35west
d <- read.csv("broyles_sec35west_2017_yl_aggreg_20171107.csv")
d <- d[!(d$n17_lbs_ac > 1000), ]        # get rid of outliers
d <- na.omit(d)                      # get rid of NAs
dp <- read.csv("broyles_sec35west_Pro_wYld.csv")

colnames(d)[6] <- "Yld"
colnames(d)[7] <- "Nrate"
dB4 <- list(d=d,dp=dp)

## VanDyke Portnell 1 south
d <- read.csv("vandyke_portnellsouth1_data_aggreg_yl_pnts_20160920.csv")
d <- d[!(d$n_lbs_ac > 1000), ]        # get rid of outliers
d <- na.omit(d)                      # get rid of NAs
dp <- read.csv("vandyke_portnellsouth1_Pro_wYld.csv")

colnames(d)[7] <- "Yld"
colnames(d)[8] <- "Nrate"
dV1 <- list(d=d,dp=dp)

## VanDyke davidsonmidwest2
d <- read.csv("vandyke_davidsonmidwest2_2016_yl_aggreg_20160920.csv")
d <- d[!(d$n_lbs_ac > 1000), ]        # get rid of outliers
```

```
d <- na.omit(d)                    # get rid of NAs
dp <- read.csv("vandyke_davidsonmidwest2_Pro_wYld.csv")

colnames(d)[7] <- "Yld"
colnames(d)[8] <- "Nrate"
dV2 <- list(d=d,dp=dp)

## Wood henrys
d <- read.csv("wood_henrys_2016_yl_aggreg_20161202.csv")
d <- d[!(d$n_lbs_ac > 1000), ]        # get rid of outliers
d <- na.omit(d)                    # get rid of NAs
dp <- read.csv("wood_henrys_Pro_wYld.csv")

colnames(d)[6] <- "Yld"
colnames(d)[7] <- "Nrate"
dW3 <- list(d=d,dp=dp)


## compile all fields data into big list (for functions below)
mastDatList <- list(dB4=dB4,
              dV1=dV1,dV2=dV2,
              dW3=dW3,
              dM1=dM1)

#============================== add yield to protein data ================================
## NOTE: I wrote and ran this section to add yield data into the protein data so that I
##      could look at the correlation between protein and yield later. For the cor() function,
##      data needed to be the same length and because yield is more heavily sampled it
##      was more appropriate to add that to the smaller dataset. Because this function takes
##      so long to run, the protein files read in the above section (^) included the result
##      of the process below. This is why this section  is commented out.

## function to associate the closest yield value to each protein point
# addYld <- function(df){
#   x <- df[[2]]
#   y <- df[[1]]
#
#   x$Yld <- 0
#   for(nd in 1:length(x[,1])){
#     dst <- matrix(0,length(y[,1]),1)
#     for(np in 1:length(y[,1])){
#       dst[np] = sqrt((x[,"x"][nd]-y[,"x"][np])^2 +
#                    (x[,"y"][nd]-y[,"y"][np])^2) #calculate the distance between points
#       #                                   in each map and put in vector
#     }
#     x$Yld[nd]<-y[,"Yld"][which.min(dst)]
#   }
#   return(x)
# }

# pc <- proc.time()
# proYld <- lapply(mastDatList,addYld)
# (proc.time()-pc)/60 ## 225.5304 minutes to run

## write new files to csv
# writeCSV <- function(d){
#   name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")
#   write.csv(d,file=paste(name,"_Pro_wYld.csv",sep=""),row.names=F)
#   return("saved")
# }
# lapply(proYld,writeCSV)

#====================== write point shapefile for each field ============================
## NOTE:  This section is also commented out because this is a function that wrote shapefiles
##       of the points for each field from yield data to draw polygons around in qGIS and bring
##       back in to clip kriged results to later. I have attached the shapefiles for the polygons
##       that are brought in later sections of this code so running this section and doing the
##       external GIS processing is not required.

## write point shapefiles for each field
# writeSHP <- function(dl){
#   d <- dl[[1]]
#   name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")
#   coordinates(d)=~x+y
#   proj4string(d)=CRS("++proj=utm +zone=12") # set it to UTM
#   d.df <- SpatialPointsDataFrame(d, data.frame(id=1:length(d)))
#   dCoor<-spTransform(d.df,CRS("+proj=longlat"))
#   dCoor.df=SpatialPointsDataFrame(dCoor, data.frame(id=1:length(dCoor)))
#   writeOGR(dCoor.df, dsn="~/Graduate/Spring_2018/LRES535/Project/R_ScriptsDirectories" ,layer=paste(name),driver="ESRI Shapefile")
#   writeSpatialShape(dCoor.df, paste(name))
#   return("done")
# }
# lapply(mastDatList,writeSHP)

#*****************************************************************************************
#*****************************************************************************************



#************************************* analysis *****************************************
#*****************************************************************************************
#==================================== set up ============================================
## make list to fill with lists of future results from each field
resultsList <- as.list(rep(0,10)) # empty list to hold future results
names(resultsList) <- as.vector(fields$code) # named with code for each field

## make list to fill with datasets for each field
bigDatList <- as.list(rep(0,10)) # empty list to hold gstat objects and kriged grids for co kriging
names(bigDatList) <- as.vector(fields$code)

## make list to fill with gstat objects for co kriging
ckList <- as.list(rep(0,10)) # empty list to hold gstat objects and kriged grids for co kriging
names(ckList) <- as.vector(fields$code)

#======================================== dB4 ===========================================
#------------------------ analysis & kriging set up ------------------------------------
## get data from master list
d <- mastDatList$dB4$d
dp <- mastDatList$dB4$dp

## id of farmer and field (for plots etc)
name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")

## take out validation datasets
```

```
dVal <- d[sample(nrow(d),.25*nrow(d)),] # take 1/4 of data for validation
dTrn <- anti_join(d, dVal)
dpVal <- dp[sample(nrow(dp),.25*nrow(dp)),]
dpTrn <- anti_join(dp,dpVal)

## put data into list
datList <- list(d,dp,dTrn,dpTrn,dVal,dpVal)
names(datList) <- c("d","dp","dTrn","dpTrn","dVal","dpVal")

## put data into spatial coordinate class
coords <- function(x){
  coordinates(x)= ~x+y
  return(x)
}
datList <- lapply(datList,coords)

## make grid to krige on
grd <- makegrid(datList$dp,n=nrow(datList$d)) # makes a grid of as many points as yield points
#                                               because every field is a different size and
#                                               a grid size for one would not be appropriate
#                                               for another
names(grd) <- c("x","y")
coordinates(grd) <- ~x+y
gridded(grd) <- T
fullgrid(grd) <- T
proj4string(grd) <- proj4string(datList$dp)

#------------------------------------ universal kriging --------------------------------
## look at trends in data
dplm <- lm(Pro~1,data=datList$dpTrn)
dpXYlm <- lm(Pro~x+y,data=datList$dpTrn)
dpXlm <- lm(Pro~x,data=datList$dpTrn) # only x is significant
#summary(dpXYlm)

## fit variogram
dpVgm <- variogram(Pro~x,datList$dpTrn)
dpVgmFit <- fit.variogram(dpVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                          "Exc","Ste","Lin", "Bes", "Per","Wav",
                          "Hol","Log","Spl"))) # fit model trying all variograms
#                                                ignore warnings...
#plot(dpVgm,dpVgmFit)

## universal kriging
dpKrig <- krige(Pro~x,datList$dpTrn,grd,dpVgmFit)
#plot(dpKrig)

## clip to polygon
sa <- readShapeSpatial(paste(name,"_poly.shp",sep = "")) # read in polygon shapefile
sasp <- as(sa,'SpatialPolygons') # change shapefile class
dpKrigR <- raster(dpKrig) # rasterize the kriging results
dpKrigSa <- mask(dpKrigR,sasp)
#plot(dpT.krig.sa, add = T) # map kriged results

#--------------------------------- UK results ----------------------------------------
## compare to validation dataset
krigVal <- krige(Pro~x, datList$dpTrn, datList$dpVal, dpVgmFit) # predict by kriging the values at the
#                                                         locations of our validation dataset
#                                                  to compare to observed values at those
#                                                  locations

## summarize errors
diff <- krigVal$var1.pred - datList$dpVal$Pro # difference between predictions and observed
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                          sum(diff)/length(diff),
                          median(dpVal$Pro),
                          rsq(krigVal$var1.pred,datList$dpVal$Pro)),
                    row.names = c("RMSE", # precision
                              "MeanError", # bias
                              "MedianError", #
                              "R2")) # how well variance is explained

## add UK kriged (predicted) points to validation data
datList$dpVal$UKpredPro <- krigVal$var1.pred

## cross validate
crossVal <- krige.cv(Pro~x, datList$dpTrn, model=dpVgmFit, nfold=nrow(datList$dpTrn)) # Cross validation holds out
#                                                                   each point and predicts one by one

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                         mean(res),
                         mean(res^2/as.data.frame(crossVal)$var1.var)),
                   row.names = c("RMSE", # precision
                             "MeanError", # bias
                             "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ukResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## add UK results to list of results
dB4Results <- list(UK=ukResults,CK=0,NK=0) # adds lists for ck and no kriging results

#-------------------------------- co kriging set up ---------------------------------------
## look at correlations b/w variables
cor(datList$dpTrn$Pro, datList$dpTrn$Yld)

## will use Nrate from d as covariable
dXYlm <- lm(Yld~x+y,data=datList$d) # look at 1st order trends
#summary(dXYlm)

## fit variogram to covariable
dCoVgm <- variogram(Yld~x+y,datList$d)
dCoVgmFit <- fit.variogram(dCoVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                              "Exc","Ste","Lin", "Bes", "Per","Wav",
                              "Hol","Log","Spl"))) # fit model trying all variograms
#                                                ignore warnings...
#plot(dCoVgm,dCoVgmFit)

## specify variograms computed above
g <- gstat(NULL, id = "Pro", form = Pro~x, data=datList$dpTrn, model=dpVgmFit,set = list(nocheck = 1)) # makes gstat object
g <- gstat(g,"Yld",Yld~x+y,datList$d,nmax = 1,model=dCoVgmFit,merge = c("Pro","Yld"),set = list(nocheck = 1))
```

```
crossV <- variogram(g,cross="ONLY") # computes only cross variogram to fit model to
cvVgmFit <- fit.variogram(crossV,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                    "Exc","Ste","Lin", "Bes", "Per","Wav",
                                    "Hol","Log","Spl"))) # fit model trying all variograms
#                                                       ignore warnings...
#plot(crossV,cvVgmFit)

g <- gstat(g,c("Pro","Yld"),model=cvVgmFit,set = list(nocheck = 1))

## add objects to ckList for co kriging
ckList$dB4 <- list(g=g,grd=grd)

## co kriging
dpCKdB4 <- predict(ckList$dB4$g,ckList$dB4$grd)
ck <- dpCKdB4

ck <- as.data.frame(ck)
ck <- na.omit(ck)
ck <- ck[!(ck$Pro.pred > 100), ]          # get rid of outliers
ck <- ck[!(ck$Pro.pred < 0), ]          # get rid of outliers
coordinates(ck) = ~x+y

summary(ck$Pro.pred); summary(ck$Pro.var)

## validation
kdB4 <- predict(ckList$dB4$g,datList$dpVal,set = list(nocheck = 1))
krigVal <- kdB4
krigVal <- as.data.frame(krigVal)
krigVal[is.na(krigVal)] <- 0 # set NA to 0 to compare predictions to observed
for(i in 1:nrow(krigVal)){
  if(krigVal$Pro.pred[i] >= 100){
    krigVal$Pro.pred[i]=0
  }
  if(krigVal$Pro.pred[i] < 0){
    krigVal$Pro.pred[i]=0
  }
}   # get rid of outliers over 100 % or below 0 % protein
coordinates(krigVal) = ~x+y

diff <- krigVal$Pro.pred - datList$dpVal$Pro
summary(diff)
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                                 sum(diff)/length(diff),
                                 median(dpVal$Pro),
                                 rsq(krigVal$Pro.pred,datList$dpVal$Pro)),
                         row.names = c("RMSE", # precision
                                       "MeanError", # bias
                                       "MedianError", #
                                       "R2")) # how well variance is explained

## add CK kriged (predicted) points to validation data
datList$dpVal$CKpredPro <- krigVal$Pro.pred

## cross validation
crossVal <- gstat.cv(g)
crossVal <- as.data.frame(crossVal)
```

```
crossVal[is.na(crossVal)] <- 0 # set NA to 0
crossVal <- crossVal[!(crossVal$Pro.pred > 100), ]          # get rid of outliers
crossVal <- crossVal[!(crossVal$Pro.pred < 0), ]          # get rid of outliers
coordinates(crossVal) = ~x+y

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                                 mean(res),
                                 mean(res^2/as.data.frame(crossVal)$Pro.var)),
                         row.names = c("RMSE", # precision
                                       "MeanError", # bias
                                       "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ckResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## store UK and CK results from field
resultsList$dB4 <- list(ukResults=ukResults,
                        ckResults=ckResults)

## store datasets created for this field
bigDatList$dB4 <- datList # put into big list (if you want to come back to rerun something with same random samples)
#                         also has the kriged predictions of validation set (for model results)

#==========================================================================================
#==========================================================================================


#======================================== dV1 ==========================================
#------------------------- analysis & kriging set up ------------------------------------
## get data from master list
d <- mastDatList$dV1$d
dp <- mastDatList$dV1$dp

## id of farmer and field (for plots etc)
name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")

## take out validation datasets
dVal <- d[sample(nrow(d),.25*nrow(d)),] # take 1/4 of data for validation
dTrn <- anti_join(d, dVal)
dpVal <- dp[sample(nrow(dp),.25*nrow(dp)),]
dpTrn <- anti_join(dp,dpVal)

## put data into list
datList <- list(d,dp,dTrn,dpTrn,dVal,dpVal)
names(datList) <- c("d","dp","dTrn","dpTrn","dVal","dpVal")

## put data into spatial coordinate class
coords <- function(x){
  coordinates(x)= ~x+y
  return(x)
}
datList <- lapply(datList,coords)

## make grid to krige on
grd <- makegrid(datList$dp,n=nrow(datList$d)) # makes a grid of as many points as yield points
```

```
#                                        because every field is a different size and
#                                        a grid size for one would not be appropriate
#                                        for another
names(grd) <- c("x","y")
coordinates(grd) <- ~x+y
gridded(grd) <- T
fullgrid(grd) <- T
proj4string(grd) <- proj4string(datList$dp)

#------------------------------------ universal kriging --------------------------------
## look at trends in data
dplm <- lm(Pro~1,data=datList$dpTrn)
dpXYlm <- lm(Pro~x+y,data=datList$dpTrn)
dpXlm <- lm(Pro~x,data=datList$dpTrn) # only x is significant
#summary(dpXYlm)

## fit variogram
dpVgm <- variogram(Pro~x,datList$dpTrn)
dpVgmFit <- fit.variogram(dpVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                               "Exc","Ste","Lin", "Bes", "Per","Wav",
                               "Hol","Log","Spl"))) # fit model trying all variograms
#                                                   ignore warnings...
#plot(dpVgm,dpVgmFit)

## universal kriging
dpKrig <- krige(Pro~x,datList$dpTrn,grd,dpVgmFit)
#plot(dpKrig)

## clip to polygon
sa <- readShapeSpatial(paste(name,"_poly.shp",sep = "")) # read in polygon shapefile
sasp <- as(sa,'SpatialPolygons') # change shapefile class
dpKrigR <- raster(dpKrig) # rasterize the kriging results
dpKrigSa <- mask(dpKrigR,sasp)
#plot(dpT.krig.sa, add = T) # map kriged results

#--------------------------------- UK results ----------------------------------------
## compare to validation dataset
krigVal <- krige(Pro~y, datList$dpTrn, datList$dpVal, dpVgmFit) # predict by kriging the values at the
#                                                      locations of our validation dataset
#                                                      to compare to observed values at those
#                                                      locations

## summarize errors
diff <- krigVal$var1.pred - datList$dpVal$Pro # difference between predictions and observed
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                            sum(diff)/length(diff),
                            median(dpVal$Pro),
                            rsq(krigVal$var1.pred,datList$dpVal$Pro)),
                      row.names = c("RMSE", # precision
                                  "MeanError", # bias
                                  "MedianError", #
                                  "R2")) # how well variance is explained

## add UK kriged (predicted) points to validation data
datList$dpVal$UKpredPro <- krigVal$var1.pred
```

```
## cross validate
crossVal <- krige.cv(Pro~y, datList$dpTrn, model=dpVgmFit, nfold=nrow(datList$dpTrn)) # Cross validation holds out
#                                                                      each point and predicts one by one

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                          mean(res),
                          mean(res^2/as.data.frame(crossVal)$var1.var)),
                    row.names = c("RMSE", # precision
                              "MeanError", # bias
                              "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ukResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## add UK results to list of results
dV1Results <- list(UK=ukResults,CK=0,NK=0) # adds lists for ck and no kriging results

#--------------------------------- co kriging set up ----------------------------------------
## look at correlations b/w variables
cor(datList$dpTrn$Pro, datList$dpTrn$Yld)

## will use Nrate from d as covariable
dXYlm <- lm(Yld~x+y,data=datList$d) # look at 1st order trends
#summary(dXYlm)

## fit variogram to covariable
dCoVgm <- variogram(Yld~x+y,datList$d)
dCoVgmFit <- fit.variogram(dCoVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                               "Exc","Ste","Lin", "Bes", "Per","Wav",
                               "Hol","Log","Spl"))) # fit model trying all variograms
#                                                  ignore warnings...
#plot(dCoVgm,dCoVgmFit)

## specify variograms computed above
g <- gstat(NULL, id = "Pro", form = Pro~y, data=datList$dpTrn, model=dpVgmFit,set = list(nocheck = 1)) # makes gstat object
g <- gstat(g,"Yld",Yld~x+y,datList$d,nmax = 1,model=dCoVgmFit,merge = c("Pro","Yld"),set = list(nocheck = 1))

crossV <- variogram(g,cross="ONLY") # computes only cross variogram to fit model to
cvVgmFit <- fit.variogram(crossV,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                              "Exc","Ste","Lin", "Bes", "Per","Wav",
                              "Hol","Log","Spl"))) # fit model trying all variograms
#                                                 ignore warnings...
#plot(crossV,cvVgmFit)

g <- gstat(g,c("Pro","Yld"),model=cvVgmFit,set = list(nocheck = 1))

## add objects to ckList for co kriging
ckList$dV1 <- list(g=g,grd=grd)

## co kriging
dpCKdV1 <- predict(ckList$dV1$g,ckList$dV1$grd)
ck <- dpCKdV1

ck <- as.data.frame(ck)
ck <- na.omit(ck)
ck <- ck[!(ck$Pro.pred > 100), ]          # get rid of outliers
ck <- ck[!(ck$Pro.pred < 0), ]          # get rid of outliers
coordinates(ck) = ~x+y

summary(ck$Pro.pred); summary(ck$Pro.var)

## validation
kdV1 <- predict(ckList$dV1$g,datList$dpVal,set = list(nocheck = 1))
krigVal <- kdV1
krigVal <- as.data.frame(krigVal)
krigVal[is.na(krigVal)] <- 0 # set NA to 0
for(i in 1:nrow(krigVal)){
  if(krigVal$Pro.pred[i] >= 100){
    krigVal$Pro.pred[i]=0
  }
  if(krigVal$Pro.pred[i] < 0){
    krigVal$Pro.pred[i]=0
  }
}   # get rid of outliers over 100 % or below 0 % protein
coordinates(krigVal) = ~x+y

diff <- krigVal$Pro.pred - datList$dpVal$Pro
summary(diff)
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                                  sum(diff)/length(diff),
                                  median(dpVal$Pro),
                                  rsq(krigVal$Pro.pred,datList$dpVal$Pro)),
                          row.names = c("RMSE", # precision
                                        "MeanError", # bias
                                        "MedianError", #
                                        "R2")) # how well variance is explained

## add CK kriged (predicted) points to validation data
datList$dpVal$CKpredPro <- krigVal$Pro.pred

## cross validation
crossVal <- gstat.cv(g)
crossVal <- as.data.frame(crossVal)
crossVal[is.na(crossVal)] <- 0 # set NA to 0
crossVal <- crossVal[!(crossVal$Pro.pred > 100), ]
crossVal <- crossVal[!(crossVal$Pro.pred < 0), ]
coordinates(crossVal) = ~x+y

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                                 mean(res),
                                 mean(res^2/as.data.frame(crossVal)$Pro.var)),
                         row.names = c("RMSE", # precision
                                       "MeanError", # bias
                                       "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ckResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## store UK and CK results from field
```

```
resultsList$dV1 <- list(ukResults=ukResults,
                        ckResults=ckResults)

## store datasets created for this field
bigDatList$dV1 <- datList

#==========================================================================================
#==========================================================================================

#========================================== dV2 ==========================================
#------------------------- analysis & kriging set up ------------------------------------
## get data from master list
d <- mastDatList$dV2$d
dp <- mastDatList$dV2$dp

## id of farmer and field (for plots etc)
name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")

## take out validation datasets
dVal <- d[sample(nrow(d),.25*nrow(d)),] # take 1/4 of data for validation
dTrn <- anti_join(d, dVal)
dpVal <- dp[sample(nrow(dp),.25*nrow(dp)),]
dpTrn <- anti_join(dp,dpVal)

## put data into list
datList <- list(d,dp,dTrn,dpTrn,dVal,dpVal)
names(datList) <- c("d","dp","dTrn","dpTrn","dVal","dpVal")

## put data into spatial coordinate class
coords <- function(x){
  coordinates(x)= ~x+y
  return(x)
}
datList <- lapply(datList,coords)

## make grid to krige on
grd <- makegrid(datList$dp,n=nrow(datList$d)) # makes a grid of as many points as yield points
#                                               because every field is a different size and
#                                               a grid size for one would not be appropriate
#                                               for another
names(grd) <- c("x","y")
coordinates(grd) <- ~x+y
gridded(grd) <- T
fullgrid(grd) <- T
proj4string(grd) <- proj4string(datList$dp)

#------------------------------------ universal kriging --------------------------------
## look at trends in data
dplm <- lm(Pro~1,data=datList$dpTrn)
dpXYlm <- lm(Pro~x+y,data=datList$dpTrn)
dpYlm <- lm(Pro~y,data=datList$dpTrn) # only y is significant
#summary(dpXYlm)

## fit variogram
dpVgm <- variogram(Pro~y,datList$dpTrn)
dpVgmFit <- fit.variogram(dpVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                  "Exc","Ste","Lin", "Bes", "Per","Wav",
                                "Hol","Log","Spl"))) # fit model trying all variograms
#                                                   ignore warnings...
#plot(dpVgm,dpVgmFit)

## universal kriging
dpKrig <- krige(Pro~y,datList$dpTrn,grd,dpVgmFit)
#plot(dpKrig)

## clip to polygon
sa <- readShapeSpatial(paste(name,"_poly.shp",sep = "")) # read in polygon shapefile
sasp <- as(sa,'SpatialPolygons') # change shapefile class
dpKrigR <- raster(dpKrig) # rasterize the kriging results
dpKrigSa <- mask(dpKrigR,sasp)
#plot(dpT.krig.sa, add = T) # map kriged results

#--------------------------------- UK results ----------------------------------------
## compare to validation dataset #!!!!!!!!!!!!!!!!!!
krigVal <- krige(Pro~y, datList$dpTrn, datList$dpVal, dpVgmFit) # predict by kriging the values at the
#                                                     locations of our validation dataset
#                                                     to compare to observed values at those
#                                                     locations

## summarize errors
diff <- krigVal$var1.pred - datList$dpVal$Pro # difference between predictions and observed
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                              sum(diff)/length(diff),
                              median(dpVal$Pro),
                              rsq(krigVal$var1.pred,datList$dpVal$Pro)),
                        row.names = c("RMSE", # precision
                                  "MeanError", # bias
                                  "MedianError", #
                                  "R2")) # how well variance is explained

## add UK kriged (predicted) points to validation data
datList$dpVal$UKpredPro <- krigVal$var1.pred

## cross validate
crossVal <- krige.cv(Pro~y, datList$dpTrn, model=dpVgmFit, nfold=nrow(datList$dpTrn)) # Cross validation holds out
#                                                                 each point and predicts one by one

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                             mean(res),
                             mean(res^2/as.data.frame(crossVal)$var1.var)),
                       row.names = c("RMSE", # precision
                                 "MeanError", # bias
                                 "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ukResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## add UK results to list of results for
dV2Results <- list(UK=ukResults,CK=0,NK=0) # adds lists for ck and no kriging results

#--------------------------------- co kriging set up ----------------------------------------
```

```
## look at correlations b/w variables
cor(datList$dpTrn$Pro, datList$dpTrn$Yld)

## will use Nrate from d as covariable
dXYlm <- lm(Yld~x+y,data=datList$d) # look at 1st order trends
#summary(dXYlm)

## fit variogram to covariable
dCoVgm <- variogram(Yld~x+y,datList$d)
dCoVgmFit <- fit.variogram(dCoVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                    "Exc","Ste","Lin", "Bes", "Per","Wav",
                                    "Hol","Log","Spl"))) # fit model trying all variograms
#                                                       ignore warnings...
#plot(dCoVgm,dCoVgmFit)

## specify variograms computed above
g <- gstat(NULL, id = "Pro", form = Pro~y, data=datList$dpTrn, model=dpVgmFit,set = list(nocheck = 1)) # makes gstat object
g <- gstat(g,"Yld",Yld~x+y,datList$d,nmax = 1,model=dCoVgmFit,merge = c("Pro","Yld"),set = list(nocheck = 1))

crossV <- variogram(g,cross="ONLY") # computes only cross variogram to fit model to
cvVgmFit <- fit.variogram(crossV,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                   "Exc","Ste","Lin", "Bes", "Per","Wav",
                                   "Hol","Log","Spl"))) # fit model trying all variograms
#                                                      ignore warnings...
#plot(crossV,cvVgmFit)

g <- gstat(g,c("Pro","Yld"),model=cvVgmFit,set = list(nocheck = 1))

## add objects to ckList for co kriging
ckList$dV2 <- list(g=g,grd=grd)

## co kriging
dpCKdV2 <- predict(ckList$dV2$g,ckList$dV2$grd)
ck <- dpCKdV2

ck <- as.data.frame(ck)
ck <- na.omit(ck)
ck <- ck[!(ck$Pro.pred > 100), ]
ck <- ck[!(ck$Pro.pred < 0), ]
coordinates(ck) = ~x+y

summary(ck$Pro.pred); summary(ck$Pro.var)

## validation
kdV2 <- predict(ckList$dV2$g,datList$dpVal,set = list(nocheck = 1))
krigVal <- kdV2
krigVal <- as.data.frame(krigVal)
krigVal[is.na(krigVal)] <- 0
for(i in 1:nrow(krigVal)){
  if(krigVal$Pro.pred[i] >= 100){
    krigVal$Pro.pred[i]=0
  }
  if(krigVal$Pro.pred[i] < 0){
    krigVal$Pro.pred[i]=0
  }
}   # get rid of outliers over 100 % or below 0 % protein
coordinates(krigVal) = ~x+y
```

```
diff <- krigVal$Pro.pred - datList$dpVal$Pro
summary(diff)
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                                sum(diff)/length(diff),
                                median(dpVal$Pro),
                                rsq(krigVal$Pro.pred,datList$dpVal$Pro)),
                        row.names = c("RMSE", # precision
                                      "MeanError", # bias
                                      "MedianError", #
                                      "R2")) # how well variance is explained

## add CK kriged (predicted) points to validation data
datList$dpVal$CKpredPro <- krigVal$Pro.pred

## cross validation
crossVal <- gstat.cv(g)
crossVal <- as.data.frame(crossVal)
crossVal[is.na(crossVal)] <- 0 # set NA to 0
crossVal <- crossVal[!(crossVal$Pro.pred > 100), ]         # get rid of outliers
crossVal <- crossVal[!(crossVal$Pro.pred < 0), ]         # get rid of outliers
coordinates(crossVal) = ~x+y

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                               mean(res),
                               mean(res^2/as.data.frame(crossVal)$Pro.var)),
                       row.names = c("RMSE", # precision
                                     "MeanError", # bias
                                     "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ckResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## store UK and CK results from field
resultsList$dV2 <- list(ukResults=ukResults,
                      ckResults=ckResults)

## store datasets created for this field
bigDatList$dV2 <- datList # put into big list (if you want to come back to rerun something with same random samples)
#                       also has the kriged predictions of validation set (for model results)

#==========================================================================================
#==========================================================================================


#======================================== dW3 ==========================================
#------------------------ analysis & kriging set up -----------------------------------
## get data from master list
d <- mastDatList$dW3$d
dp <- mastDatList$dW3$dp

## id of farmer and field (for plots etc)
name <- paste(as.character(d$farmer[1]),"_",as.character(d$field[1]),sep = "")
```

```
## take out validation datasets
dVal <- d[sample(nrow(d),.25*nrow(d)),] # take 1/4 of data for validation
dTrn <- anti_join(d, dVal)
dpVal <- dp[sample(nrow(dp),.25*nrow(dp)),]
dpTrn <- anti_join(dp,dpVal)

## put data into list
datList <- list(d,dp,dTrn,dpTrn,dVal,dpVal)
names(datList) <- c("d","dp","dTrn","dpTrn","dVal","dpVal")

## put data into spatial coordinate class
coords <- function(x){
  coordinates(x)= ~x+y
  return(x)
}
datList <- lapply(datList,coords)

## make grid to krige on
grd <- makegrid(datList$dp,n=nrow(datList$d)) # makes a grid of as many points as yield points
#                                     because every field is a different size and
#                                     a grid size for one would not be appropriate
#                                     for another
names(grd) <- c("x","y")
coordinates(grd) <- ~x+y
gridded(grd) <- T
fullgrid(grd) <- T
proj4string(grd) <- proj4string(datList$dp)

#------------------------------------ universal kriging --------------------------------
## look at trends in data
dplm <- lm(Pro~1,data=datList$dpTrn)
dpXYlm <- lm(Pro~x+y,data=datList$dpTrn)
#summary(dpXYlm)

## fit variogram
dpVgm <- variogram(Pro~x+y,datList$dpTrn)
dpVgmFit <- fit.variogram(dpVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                "Exc","Ste","Lin", "Bes", "Per","Wav",
                                "Hol","Log","Spl"))) # fit model trying all variograms
#                                                   ignore warnings...
#plot(dpVgm,dpVgmFit)

## universal kriging
dpKrig <- krige(Pro~x+y,datList$dpTrn,grd,dpVgmFit)
#plot(dpKrig)

## clip to polygon
sa <- readShapeSpatial(paste(name,"_poly.shp",sep = "")) # read in polygon shapefile
sasp <- as(sa,'SpatialPolygons') # change shapefile class
dpKrigR <- raster(dpKrig) # rasterize the kriging results
dpKrigSa <- mask(dpKrigR,sasp)
#plot(dpT.krig.sa, add = T) # map kriged results

#--------------------------------- UK results ----------------------------------------
## compare to validation dataset
krigVal <- krige(Pro~x+y, datList$dpTrn, datList$dpVal, dpVgmFit) # predict by kriging the values at the
#                                                      locations of our validation dataset
#                                                   to compare to observed values at those
#                                                   locations

## summarize errors
diff <- krigVal$var1.pred - datList$dpVal$Pro # difference between predictions and observed
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                          sum(diff)/length(diff),
                          median(dpVal$Pro),
                          rsq(krigVal$var1.pred,datList$dpVal$Pro)),
                    row.names = c("RMSE", # precision
                               "MeanError", # bias
                               "MedianError", #
                               "R2")) # how well variance is explained

## add UK kriged (predicted) points to validation data
datList$dpVal$UKpredPro <- krigVal$var1.pred

## cross validate
crossVal <- krige.cv(Pro~x+y, datList$dpTrn, model=dpVgmFit, nfold=nrow(datList$dpTrn)) # Cross validation holds out
#                                                                   each point and predicts one by one

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                          mean(res),
                          mean(res^2/as.data.frame(crossVal)$var1.var)),
                    row.names = c("RMSE", # precision
                               "MeanError", # bias
                               "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ukResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## add UK results to list of results for dB1
dW3Results <- list(UK=ukResults,CK=0,NK=0) # adds lists for ck and no kriging results


#--------------------------------- co kriging set up ----------------------------------------
## look at correlations b/w variables
cor(datList$dpTrn$Pro, datList$dpTrn$Yld)

## will use Nrate from d as covariable
dXYlm <- lm(Yld~x+y,data=datList$d) # look at 1st order trends
#summary(dXYlm)

## fit variogram to covariable
dCoVgm <- variogram(Yld~x+y,datList$d) #!!!!!!!!!!!!!!!
dCoVgmFit <- fit.variogram(dCoVgm,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                               "Exc","Ste","Lin", "Bes", "Per","Wav",
                               "Hol","Log","Spl"))) # fit model trying all variograms
#                                                 ignore warnings...
#plot(dCoVgm,dCoVgmFit)

## compare variogram structure to target variable
dCoVgmFit$range[2]; dpVgmFit$range[2]
round(dCoVgmFit$psill[1]/sum(dCoVgmFit$psill),2)
round(dpVgmFit$psill[1]/sum(dpVgmFit$psill),2)

## specify variograms computed above
g <- gstat(NULL, id = "Pro", form = Pro~x+y, data=datList$dpTrn, model=dpVgmFit,set = list(nocheck = 1)) # makes gstat object
g <- gstat(g,"Yld",Yld~x+y,datList$d,nmax = 1,model=dCoVgmFit,merge = c("Pro","Yld"),set = list(nocheck = 1))

crossV <- variogram(g,cross="ONLY") # computes only cross variogram to fit model to
cvVgmFit <- fit.variogram(crossV,vgm(c("Cir","Sph","Pen","Mat","Nug","Exp","Gau",
                                   "Exc","Ste","Lin", "Bes", "Per","Wav",
                                   "Hol","Log","Spl"))) # fit model trying all variograms
#                                                    ignore warnings...
#plot(crossV,cvVgmFit)

g <- gstat(g,c("Pro","Yld"),model=cvVgmFit,set = list(nocheck = 1))

## add objects to ckList for co kriging
ckList$dW3 <- list(g=g,grd=grd)

## co kriging
dpCKdW3 <- predict(ckList$dW3$g,ckList$dW3$grd)
ck <- dpCKdW3

ck <- as.data.frame(ck)
ck <- na.omit(ck)
ck <- ck[!(ck$Pro.pred > 100), ]        # get rid of outliers
ck <- ck[!(ck$Pro.pred < 0), ]          # get rid of outliers
coordinates(ck) = ~x+y

summary(ck$Pro.pred); summary(ck$Pro.var)

## validation
kdW3 <- predict(ckList$dW3$g,datList$dpVal,set = list(nocheck = 1))
krigVal <- kdW3
krigVal <- as.data.frame(krigVal)
krigVal[is.na(krigVal)] <- 0 # set NA to 0
for(i in 1:nrow(krigVal)){
  if(krigVal$Pro.pred[i] >= 100){
    krigVal$Pro.pred[i]=0
  }
  if(krigVal$Pro.pred[i] < 0){
    krigVal$Pro.pred[i]=0
  }
}   # get rid of outliers over 100 % or below 0 % protein
coordinates(krigVal) = ~x+y

diff <- krigVal$Pro.pred - datList$dpVal$Pro
summary(diff)
rsq <- function(x,y){cor(x,y)^2} # calculates r squared
diffResults <- data.frame(Value=c(sqrt(sum(diff^2)/length(diff)),
                                 sum(diff)/length(diff),
                                 median(dpVal$Pro),
                                 rsq(krigVal$Pro.pred,datList$dpVal$Pro)),
                          row.names = c("RMSE", # precision
                                        "MeanError", # bias
                                        "MedianError", #
                                        "R2")) # how well variance is explained
```

```
## add CK kriged (predicted) points to validation data
datList$dpVal$CKpredPro <- krigVal$Pro.pred

## cross validation
crossVal <- gstat.cv(g)
crossVal <- as.data.frame(crossVal)
crossVal[is.na(crossVal)] <- 0 # set NA to 0
crossVal <- crossVal[!(crossVal$Pro.pred > 100), ]
crossVal <- crossVal[!(crossVal$Pro.pred < 0), ]
coordinates(crossVal) = ~x+y

## summarize results of cross validation
res <- as.data.frame(crossVal)$residual
resResults <- data.frame(Value=c(sqrt(mean(res^2)),
                                 mean(res),
                                 mean(res^2/as.data.frame(crossVal)$Pro.var)),
                         row.names = c("RMSE", # precision
                                       "MeanError", # bias
                                       "MSDR")) # Mean Squared Deviation Ratio

## combine results from universal kriging
ckResults <- list(ComparisonValidation=diffResults,CrossValidation=resResults)

## store UK and CK results from field
resultsList$dW3 <- list(ukResults=ukResults,
                        ckResults=ckResults)

## store datasets created for this field
bigDatList$dW3 <- datList # put into big list (if you want to come back to rerun something with same random samples)
#                             also has the kriged predictions of validation set (for model results)

#****************************************************************************************
#****************************************************************************************



#************************************** modelling ****************************************
#****************************************************************************************
#================================== set up ============================================
## make list to fill with AIC results of each method
resultsList2 <- as.list(rep(0,10)) # empty list to hold future results
names(resultsList2) <- as.vector(fields$code) # named with code for each field

## function for predicting yield
hyperbolic <- function(alpha, beta, gamma, N){
  y <- alpha + ((beta - alpha) * N)/((1.0/gamma) + N)
  return(y)
}

#====================================== dB4 ==========================================
## data
dp <- as.data.frame(bigDatList$dB4$dpVal)

## results table
rslts <- data.frame(AIC=rep(0,3),row.names = c("NK","UK","CK"))

#------------------------------------- no kriging -------------------------------------
```

```
## non linear least squares model
gamma = 0.02
modNK <- nls(Pro ~ hyperbolic(a0 +
                              a1*aspect_cos +
                              a2*aspect_sin +
                              a3*slope_deg +
                              a4*elev_m +
                              a5*tpi30,
                            b0 +
                              b1*ndvi_2017 +
                              b2*ndvi_2016 +
                              b3*ndvi_2015,
                            gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                       b0=1, b1=2, b2=.1, b3=1)) #
## NOTE: not refining the model because the validation sets are so small that almost all parameters
##       would be taken out. Comparing all models with all parameters because AIC also takes
##       number of parameters into account

rslts[1,1] <- AIC(modNK)

#-------------------------------------- universal kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modUK <- nls(UKpredPro ~ hyperbolic(a0 +
                                    a1*aspect_cos +
                                    a2*aspect_sin +
                                    a3*slope_deg +
                                    a4*elev_m +
                                    a5*tpi30,
                                  b0 +
                                    b1*ndvi_2017 +
                                    b2*ndvi_2016 +
                                    b3*ndvi_2015,
                                  gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                       b0=1, b1=2, b2=.1, b3=1)) #
rslts[2,1] <- AIC(modUK)

#-------------------------------------- co kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modCK <- nls(CKpredPro ~ hyperbolic(a0 +
                                    a1*aspect_cos +
                                    a2*aspect_sin +
                                    a3*slope_deg +
                                    a4*elev_m +
                                    a5*tpi30,
                                  b0 +
                                    b1*ndvi_2017 +
                                    b2*ndvi_2016 +
                                    b3*ndvi_2015,
                                  gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                     b0=1, b1=2, b2=.1, b3=1)) #
rslts[3,1] <- AIC(modCK)

## add rslts to bigger results list
resultsList2$dB4 <- rslts


#======================================== dV1 ==========================================
## data
dp <- as.data.frame(bigDatList$dV1$dpVal)

## results table
rslts <- data.frame(AIC=rep(0,3),row.names = c("NK","UK","CK"))

#-------------------------------------- no kriging --------------------------------------
## non linear least squares model
gamma = 0.02
modNK <- nls(Pro ~ hyperbolic(a0 +
                             a1*aspect_cos +
                             a2*aspect_sin +
                             a3*slope_deg +
                             a4*elev_m +
                             a5*tpi30,
                           b0 +
                             b1*ndvi_2016 +
                             b2*ndvi_2015,
                           gamma, Nrate),
           data=dp, nls.control(maxiter = 500, minFactor=1e-10),
           start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                      b0=1, b1=2, b2=.1)) #
## NOTE: not refining the model because the validation sets are so small that almost all parameters
##       would be taken out. Comparing all models with all parameters because AIC also takes
##       number of parameters into account

rslts[1,1] <- AIC(modNK)

#-------------------------------------- universal kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modUK <- nls(UKpredPro ~ hyperbolic(a0 +
                                   a1*aspect_cos +
                                   a2*aspect_sin +
                                   a3*slope_deg +
                                   a4*elev_m +
                                   a5*tpi30,
                                 b0 +
                                   b1*ndvi_2016 +
                                   b2*ndvi_2015,
                                 gamma, Nrate),
           data=dp, nls.control(maxiter = 500, minFactor=1e-10),
           start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                      b0=1, b1=2, b2=.1)) #
rslts[2,1] <- AIC(modUK)

#-------------------------------------- co kriging ---------------------------------
## non linear least squares model
gamma = 0.02
```

```
modCK <- nls(CKpredPro ~ hyperbolic(a0 +
                                 a1*aspect_cos +
                                 a2*aspect_sin +
                                 a3*slope_deg +
                                 a4*elev_m +
                                 a5*tpi30,
                               b0 +
                                 b1*ndvi_2016 +
                                 b2*ndvi_2015,
                               gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                       b0=1, b1=2, b2=.1)) #
rslts[3,1] <- AIC(modCK)

## add rslts to bigger results list
resultsList2$dV1 <- rslts



#======================================= dV2 ==========================================
## data
dp <- as.data.frame(bigDatList$dV2$dpVal)

## results table
rslts <- data.frame(AIC=rep(0,3),row.names = c("NK","UK","CK"))

#-------------------------------------- no kriging --------------------------------------
## non linear least squares model
gamma = 0.02
modNK <- nls(Pro ~ hyperbolic(a0 +
                          a1*aspect_cos +
                          a2*aspect_sin +
                          a3*slope_deg +
                          a4*elev_m +
                          a5*tpi30,
                        b0 +
                          b1*ndvi_2016 +
                          b2*ndvi_2015,
                        gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                       b0=1, b1=2, b2=.1)) #
## NOTE: not refining the model because the validation sets are so small that almost all parameters
##       would be taken out. Comparing all models with all parameters because AIC also takes
##       number of parameters into account

rslts[1,1] <- AIC(modNK)

#-------------------------------------- universal kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modUK <- nls(UKpredPro ~ hyperbolic(a0 +
                                 a1*aspect_cos +
                                 a2*aspect_sin +
                                 a3*slope_deg +
                                 a4*elev_m +
                                  a5*tpi30,
                                b0 +
                                  b1*ndvi_2016 +
                                  b2*ndvi_2015,
                                gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                      b0=1, b1=2, b2=.1)) #
rslts[2,1] <- AIC(modUK)

#-------------------------------------- co kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modCK <- nls(CKpredPro ~ hyperbolic(a0 +
                                  a1*aspect_cos +
                                  a2*aspect_sin +
                                  a3*slope_deg +
                                  a4*elev_m +
                                  a5*tpi30,
                                b0 +
                                  b1*ndvi_2016 +
                                  b2*ndvi_2015,
                                gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                      b0=1, b1=2, b2=.1)) #
rslts[3,1] <- AIC(modCK)

## add rslts to bigger results list
resultsList2$dV2 <- rslts



#======================================== dW3 ==========================================
## data
dp <- as.data.frame(bigDatList$dW3$dpVal)

## results table
rslts <- data.frame(AIC=rep(0,3),row.names = c("NK","UK","CK"))

#-------------------------------------- no kriging --------------------------------------
## non linear least squares model
gamma = 0.02
modNK <- nls(Pro ~ hyperbolic(a0 +
                            a1*aspect_cos +
                            a2*aspect_sin +
                            a3*slope_deg +
                            a4*elev_m +
                            a5*tpi30,
                          b0 +
                            b1*ndvi_2016 +
                            b2*ndvi_2015,
                          gamma, Nrate),
            data=dp, nls.control(maxiter = 500, minFactor=1e-10),
            start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                      b0=1, b1=2, b2=.1)) #
## NOTE: not refining the model because the validation sets are so small that almost all parameters
##      would be taken out. Comparing all models with all parameters because AIC also takes
##      number of parameters into account

rslts[1,1] <- AIC(modNK)

#-------------------------------------- universal kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modUK <- nls(UKpredPro ~ hyperbolic(a0 +
                                  a1*aspect_cos +
                                  a2*aspect_sin +
                                  a3*slope_deg +
                                  a4*elev_m +
                                  a5*tpi30,
                                b0 +
                                  b1*ndvi_2016 +
                                  b2*ndvi_2015,
                                gamma, Nrate),
         data=dp, nls.control(maxiter = 500, minFactor=1e-10),
         start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                    b0=1, b1=2, b2=.1)) #
rslts[2,1] <- AIC(modUK)

#-------------------------------------- co kriging ---------------------------------
## non linear least squares model
gamma = 0.02
modCK <- nls(CKpredPro ~ hyperbolic(a0 +
                                  a1*aspect_cos +
                                  a2*aspect_sin +
                                  a3*slope_deg +
                                  a4*elev_m +
                                  a5*tpi30,
                                b0 +
                                  b1*ndvi_2016 +
                                  b2*ndvi_2015,
                                gamma, Nrate),
         data=dp, nls.control(maxiter = 500, minFactor=1e-10),
         start=list(a0=-200, a1=5, a2=-30, a3=10, a4=15,a5=3,
                    b0=1, b1=2, b2=.1)) #
rslts[3,1] <- AIC(modCK)

## add rslts to bigger results list
resultsList2$dW3 <- rslts

#*****************************************************************************************
#*****************************************************************************************
```