

# LVI Linear Discriminant Analyses: Beta Version

---

April 12, 2013

Prepared for:

Forest Analysis and Inventory Branch

BC Ministry of Forests, Lands and Natural Resource Operations

By

Ian Moss, PhD, RPF

Tesera Systems Inc.

Victoria, BC

# LVI Linear Discriminant Analyses: Beta Version

---

## Table of Contents

1.	Introduction .....	1
1.1.	kNN: An overview .....	1
1.2.	LVI: An overview .....	2
1.3.	The integration of kNN with LVI: Issues to be aware of. ....	4
1.4.	About this manual.....	7
2.	Installation .....	9
2.1.	Create an R Working Directory .....	9
2.2.	Installations.....	9
2.3.	Create R Working Directory & Load R packages .....	9
2.4.	Python Modules .....	10
3.	Fuzzy C-Means Classification .....	11
4.	Start with Variable Selection from a Large Number of Variables .....	12
5.	Run Discriminant Analysis for a Single X-Variable Set .....	15
6.	Run Linear Discriminant Analysis for Multiple X-Variable Sets.....	17
7.	Produce Unique X-Variable Subset Correlation Matrix .....	21
8.	Produce Original Classification Unique X- or Y-Variable Subset Box and Scatter Plots.....	23
9.	Combine Evaluation Datasets .....	25
10.	Alternative Variable Selection Procedure (Under Development).....	26
11.	Calculate Z-Scores, Nearest Neighbours, and Root Mean Squared Errors. ....	27
12.	Calculate Nearest Neighbors, and Root Mean Squared Errors Using Selected X-Variables .....	29
13.	Calculate Nearest Neighbors, and Root Mean Squared Errors Using Selected Y-Variables .....	30
14.	Identify k-Nearest Neighbours in Target Dataset .....	30
15.	Compile the kNN Target Dataset Y-Variable Statistics.....	30
16.	Convert TNNSTATS file VARNAMES into Column Format.....	31
17.	References .....	31

## [Appendices](#)

Appendix I: A listing of Python Modules and R Scripts and the R Working Directory Structure

Appendix II: The Standard LVI kNN Analysis Process

# LVI Linear Discriminant Analyses: Beta Version

---

## 1. Introduction

The goal of manual is to provide a guide to the use of some software tools for improving estimation of the British Columbia Ministry of Forests, Lands and Natural Resource Operations (MoFL) Landscape Vegetation Inventory (LVI) polygon attributes. LVI is based on deriving polygons and associated attributes from Landsat Thematic Mapper (TM). A small portion of the polygons have manual interpretations of vegetation inventory attributes as observed in aerial photographs. The objective is to estimate vegetation inventory attributes for those polygons that do not have assigned vegetation inventory attributes (referred to as the target dataset) by way of association with those polygons that are of similar characteristics and do have such attributes (referred to as the reference dataset). One method to achieve this is referred to as the k-Nearest Neighbour (kNN) concept. The introduction that follows provides a high level overview of nearest neighbor concepts with reference to process of constructing a LVI.

### 1.1. kNN: An overview

kNN is based on the notion that there are two datasets, one that includes a complete set of variables, and one that does not. The first dataset is called the reference (or training) set and the latter is called the target set. The missing variables or data in the target set are referred to as the Y-variable dataset and the variables that are held in common across both datasets are referred to as the X-variables. For each observation in the target set, one or more nearest neighbours are found in the reference set based on the magnitudes of differences in the X-variables. If the magnitude of differences between a target observation and a given reference observation, summed across all X-variables, is found to be a minimum when compared with differences in relation to each of the the remaining, alternative observations in the reference set, then that particular reference observation is identified as the nearest neighbor. This process may be extended from the nearest neighbor, to the next nearest neighbor, and so on, up to k-nearest neighbours as defined by users of this technique. Once the nearest neighbours have been identified, the missing Y-variables can then be estimated as the mean value associated with k nearest neighbours and those values are then assigned to the target observation as a best estimate of the missing data.

The above mentioned process is a form of classification. At the extreme, when  $k = 1$ , each observation represents a single class. As  $k$  increases ( $k > 1$ ), the number of classes remains the same (equal to the number of observations) but the classes are based on some degree of overlapping sets of observations (O) in the reference data calculated as follows:

$$O = (k-1)/Nr$$

## LVI Linear Discriminant Analyses: Beta Version

where,

Nr is the total number of observations in the reference dataset.

The concept of kNN can be contrasted with more traditional types of classification where each observation belongs to one, and only one group (e.g. k-Means classification), or where each observation may have membership's in different groups, but the attributes that define each group remains constant (i.e. Fuzzy C-Means classification; Bezdek 1981).

### 1.2. LVI: An overview

LVI begins with three sources of data: (1) recent remote sensing from Landsat TM, (2) ancillary data, including for example the previous Vegetation Resource Inventory (VRI) and Predictive Ecosystem Mapping (PEM), and (3) photo sample plot data (Figure 1). The LVI process starts with segmentation of Landsat TM imagery (at a 30 m pixel resolution) to derive inventory polygons. The segmentation process relies on joining neighbouring pixels according to the strength of differences in 7 spectral bands (blue-green, green, red, 2 near infrared bands, thermal infrared, and shortwave infrared). More specifically (Wulder et al. 2005),

“Image segmentation is the partitioning of a digital image into a set of jointly exhaustive and mutually disjoint regions (uniformity being evaluated by a dissimilarity measure with which the partition itself is constructed). “

While there are hundreds of different merging techniques, for the purpose of forest inventory only a few methods have been applied, mostly based on “region merging” (Wulder et al. 2005).

Traditional (digital camera) aerial photography is then used to establish aerial photo-sample plots across the area of interest. These plots are used to estimate the Y-variables of interest that form an essential part of the reference set. Typically each plot is relatively small in ground area representation (e.g. 30 x 30 m), but the technique is not limited to this scale of observation. Using manual photo interpretation, each of the plots are described according to a standard set of Vegetation Resource Inventory (VRI) attributes, including for example, tree species composition, basal area, and dominant tree height. The Landsat segments with corresponding photo-plot data are then used as a reference data set. Landsat indices are derived from the 7 spectral bandwidths referred to above, and then selected for use with different weights assigned for the purpose of best estimating a selected set of photo plot inventory attributes (Y-variables). More specifically, the selected X-variables and their associated weights are designed to either maximize correlations with selected Y-variables (e.g. Most Similar Neighbour; Mouer and Stage 1995), or minimize root mean squared errors for estimating these same Y-variables (e.g. Seemingly Unrelated Regression; LeMay et al. 2012), or perhaps with some other criteria in mind. The procedures in this manual make use of traditional methods of classification to describe the Y-variable set (Fuzzy C-Means classification), and Multiple Discriminant Analysis (MDA) to complete variable selection and apply weights. The objective of MDA is to maximize the ratio of the between to within group sum of squares by applying weights to the X-variable dataset (Dillon and Goldstein 1984).

## Generalized Inventory Process

February-16-13

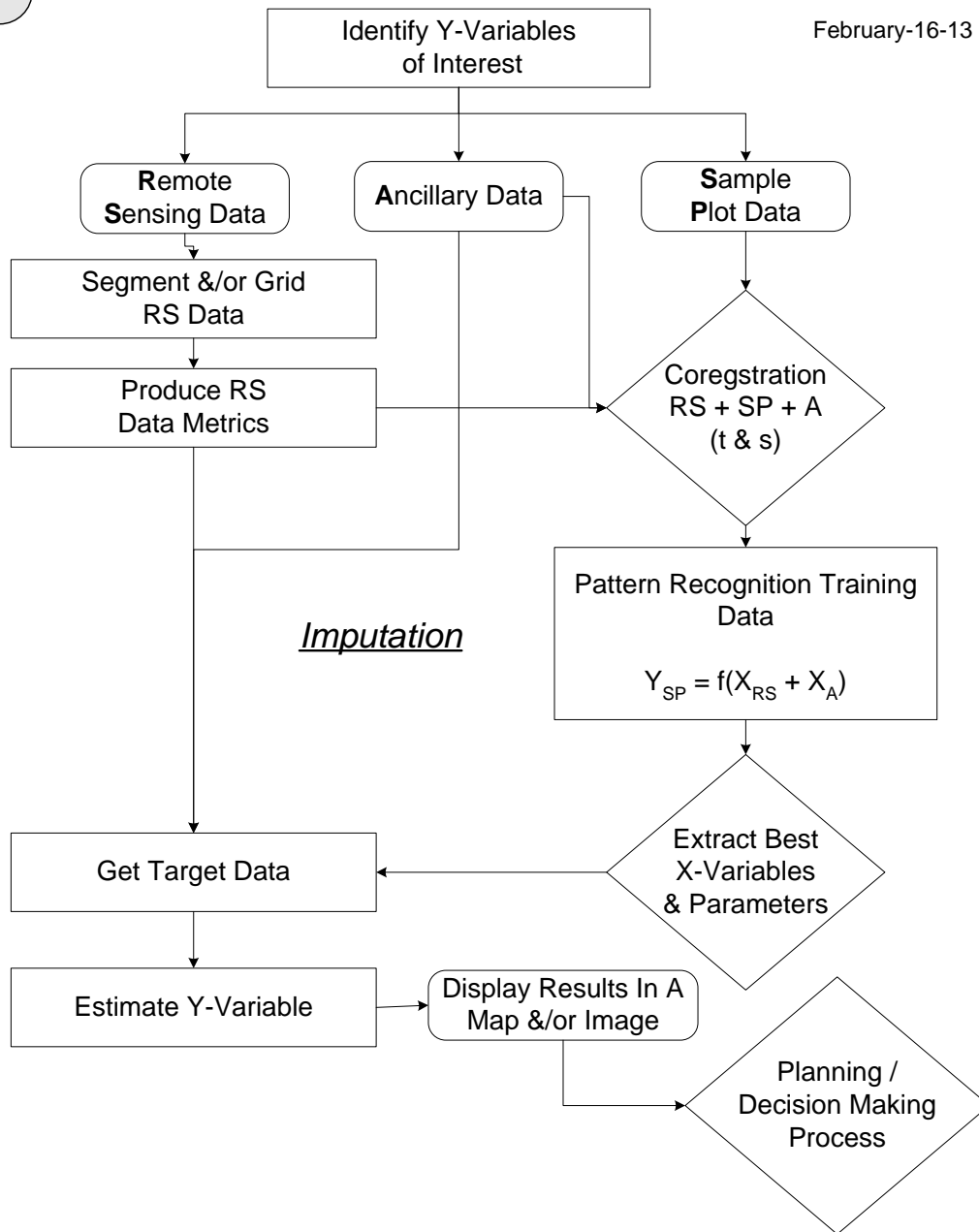


Figure 1. A generalized process for implanting automated inventories using remote sensing data, ancillary data, sample photo and/or ground plot data.

## LVI Linear Discriminant Analyses: Beta Version

In the final step, Landsat TM segments that are without any photo plot information (these are referred to as the target dataset) are associated with one or more nearest neighbour, reference segments based on the selected X-variables and their associated weights. Average Y-variable characteristics are then derived from the k-nearest neighbours associated with each segment.

### 1.3. The integration of kNN with LVI: Issues to be aware of.

The process referred to above assumes that there is a set of Y-variables that have been observed in a sample dataset that are of primary interest, and a set of X-variables that can be used to estimate the Y-variables. The X-variables provide census data for all of the pixels, grid cells, or segments within an area. It is important that the Y-variable sample dataset is representative of the range of conditions observed within a particular geographic area, and it may also be advisable to have a sample that is established for the purpose of being representative of the population as a whole (e.g. grid based sampling with a random start and orientation; pers. comm. Petteri Packalén). The argument in favour of the latter decision is to ensure that the distribution of samples is proportional to the stand type frequency distribution within any given population; this is to avoid biases that might occur as a result of model based sampling. However the risk is that certain extreme values may be missed in the sampling process. The best solution may be a combination of population and model based sample designs.

Another issue of concern is that the size and scale of each sample correspond as closely as possible with the size and scale of delineated units used to represent the X-variables. In particular when sample units are small in size and number relative to the size of each segment, the level of sampling error may be large, particularly when the within segment sample size (cluster based sampling) approaches 1. In addition the locations of the sample units should be known with a small degree of error to avoid registration error and the sample units should be established entirely within delineated segments without crossing boundaries.

Spatial autocorrelation is yet another issue to be addressed. This issue applies to data analysis when it is commonly assumed that the samples are independently and identically distributed. Tests are available to evaluate whether or not this is the case. The plots should be reasonably well separated to avoid this problem. Spatial autocorrelation will also influence be a factor in the preparation of a final map or set of maps. Estimates of the reliability of maps must take this into account.

A key aspect of this work involves the use of robust procedures for both variable selection and assignment of variable weights as a basis for identifying nearest neighbours. This becomes particularly important where there are a large number of X-variables to choose from. One commonly used procedure includes either backward stepwise, forward stepwise, or (combined forward and backward) stepwise procedures for selecting eligible variables to be used in a prediction equation; a p-value or F-value threshold is used in this process for removal (addition) of a variable from (into) a predicted equation. With large datasets a bootstrapping procedure may be used to develop observation subsamples (usually with replacement) for model calibration purposes. The remaining data not initially included in the subsample can then be used for model testing. Another common procedure applied for the purpose of model testing involves the use of jackknifing where each single observation is removed from the dataset in turn, the remaining portion of the dataset is used for model calibration, and then

## LVI Linear Discriminant Analyses: Beta Version

the nearest neighbor Y variable set is compared with the actual set associated with the removed observation as a basis for estimating root mean squared errors. The partitioning of the reference dataset into two, one for model calibration and one for model testing can provide information on parameter uncertainty, and can produce estimates of precision, bias and classification success rates that better reflect final results with respect to the target dataset.

In conjunction with the bootstrap or jackknife procedures, random subsets of variables may also be selected as being eligible for calibrating different combinations of variables. Evaluation of the results can be done internally (with respect to the calibration data), and externally (with respect to subsets of observation data not included in each of the subsamples). Individual variables can then be selected for eligibility in a final modeling according to: a) how frequently they appear in a final solution (when compared with their availability), b) the ranking of each solution (RMSE, correlations of predicted versus actual outcomes), and c) their average ranking relative to the influence of the other variables in each equation (e.g. change in RMSE or  $r_{adj}^2$  if the variable is removed).

Finally there is one other significant consideration in nearest neighbour types of analyses: What to do with outliers? Certain reference observations can often be identified as making a proportionately large contribution to errors of estimation of the Y-variables in the final outcome. These observations should be removed from the reference dataset. More generally, once a model has been developed and tested, certain levels or errors or inconsistencies with respect to the observed versus imputed nearest neighbour will be present. Often it is not known whether the errors are due to: measurement error, registration error, observer error, sample error, sensor error, the need for additional remote sensing or ancillary data (i.e. data from other sources), or the need for additional remote sensing indices, etc. Precision can be improved by ensuring that the predicted versus actual nearest neighbour s in the calibration data set are the same within some kind of error tolerance limits, recognizing that there is a trade-off between obtaining increased precision and reducing the amount of data available for calibration. In general, and with a reasonably large sample dataset, as tolerance limits are reduced, RMSE is expected to decrease down to a minimum when removing observations that contribute substantial amounts of error in the process, and then it is expected to start increasing due to the loss of calibration data. This suggests an optimal level of precision may be found iteratively, first by exclusion of data calibration points contributing the most error, and then by re-evaluation of nearest neighbour outcomes. The variable selection process may be inserted in the middle of this filter, albeit this would add considerable complexity to the process.

Many different refinements and alterations have been applied to the general procedures outlined in the previous paragraphs. This includes for example retaining a number of different models, and using all of them in a process for determining an expected outcome (i.e. bagging, a term derived from “bootstrap aggregation” where each dataset is used to train a component classifier and the final decision is based on the vote of each classifier. Such a process is referred to as a “multiclassifier”. This concept also bares similarities to the use of neural networks (see Duda et al. 2001 for a more complete discussion). Duda et al.(2001, p 499) summarize the situation as follows:

- “The No Free Lunch Theorem states that in the absence of prior information about the problem there are no reasons to prefer one learning algorithm or classifier model over another.”



## LVI Linear Discriminant Analyses: Beta Version

- “Given that a finite set of feature values are used to distinguish the patterns under consideration, the Ugly Duckling Theorem states that the number of predicates shared by any two different patterns is constant and does not depend on the choice of the two objects.”
- Therefore, “There is no problem-independent “best” learning or pattern recognition system, nor feature representation.”
- “The bias-variance dilemma states that learning procedures with increased flexibility to adapt to the training data (e.g. have more free parameters) tend to have lower bias but higher variance.”

There is no perfect system but for those who are interested in routinely applying kNN concepts it is desirable to have a system for:

1. Identifying the key Y-variables of interest, and key, candidate X-variables that are likely to be useful in terms of explaining the similarities and differences amongst the observed combinations of Y-variables.
2. Acquiring or producing the data identified in step 1.
3. Importing the data from various sources into a computing environment.
4. Connecting the various sources of data based on their specific locations and extents.
5. Using the reference dataset to determine the best set(s) of X-Variables for explaining similarities and differences amongst the Y-variables, and so too for identifying appropriate variable weights.
6. Selecting a distance metric (Euclidean, Mahalanobis, Absolute Difference) to be applied to the selected X-variables, either before or after having applied the variable weights.
7. Evaluating various combinations of X- variables, variable weights, and distance metrics to determine the best combination for estimating the Y-variables.
8. Using the selected variables, variable weights, and distance metrics to match between 1 and k nearest neighbours from the reference set with each observation in the target set.
9. Use from 1 to k nearest neighbours to generate the mean Y-variable statistics associated with each target observation.
10. Providing GIS database products, along with standard reports providing the information on the testing used to determine the finally adopted procedures, a summary of the results, and a summary of the analyses carried out to determine reliability of the final results.

### 1.4. About this manual

The system described above is specified according to the kinds of software used to implement it, the set of procedures invoked by the software, and the human intervention required to run the software, implement the procedures, and evaluate the final results. The focus of this manual is to document the system, and thereby enable users in running the various routines and interpreting the various outputs. This manual is a first approximation that was developed in parallel with development of various scripts in R and modules in Python. As such it represents a work in progress rather than a final product. While an effort has been made to ensure that the manual is complete the tools developed below continue to be expanded, modified, and reconfigured to extend the applications, improve performance, and to make the whole process easier to install and use. As a result some of the material presented herein is identified as being “Under Development”. Other utilities and functions may have already been added to the latest version and not yet included in the manual. This manual is based on the assumption that user’s are competent in the use of computers and have some experience in programming; some familiarity with the use of Python and R is will be helpful but ultimately not required. The scripts developed in R are generally short and easy to interpret; running these scripts may be a useful way to start learning R if users are not already familiar with it.

The documentation herein pertains to the use of Microsoft WINDOWS 7 operating system. The R scripts were developed and tested using R version 2.9.1., but have also been run successfully in version 2.15.3. The Python code was developed in Python2.6 and has run successfully in Python 2.7. The Rscripts and Python modules are all located in an “R working directory” or “Rwd.” They operate on “.csv” files that are in the top of that directory.

The Manual itself covers installation, and a then a set of instructions for specific types of analyses (e.g. variable selection using a particular R routine), in particular:

- Fuzzy C-Means Classification.
- Variable Selection (using the R subselect package).
- Discriminant analysis using a single X-variable set (using the R MASS package).
- Discriminant analyses using multiple X-variable sets (R MASS package).
- Identifying correlations (Pearson’s  $r^2$ ) amongst X- or Y-variables.
- Producing box-plots and scatter grams for the purpose of visualizing data.
- Combining several output files into a single file for evaluation of results.
- Using the results of discriminant analyses to produce Z-Scores, identify nearest neighbor, and generating Root Mean Squared Errors (RMSE’s) for k-Nearest Neighbours, all based on use of the reference dataset only.
- Using the results of discriminant analyses to define the X-variables, and then use those variables to identify nearest neighbor, and generating Root Mean Squared Errors (RMSE’s) for k-Nearest Neighbours, all based on use of the reference dataset only.
- Using the Y-variables to identify nearest neighbor, and generating Root Mean Squared Errors (RMSE’s) for k-Nearest Neighbours, all based on use of the reference dataset only.
- Assigning up to k-Nearest Neighbours from the reference dataset to each of the target observations, based on the use of discriminate function Z-Scores.

## LVI Linear Discriminant Analyses: Beta Version

- Generating Y-variable characteristics for  $k = 1, 2, 3, \dots, n$ , nearest neighbours associated with the target set.
- Reformatting the target kNN Y-variable dataset for ease of use.

Each of these procedures start with loading a dataset, selecting X and/or Y variables, doing an analysis, reading the results into a format suitable for writing to a file, and finally writing the results to one or more files. Standard file naming conventions are used to allow progression from one step to the next without requiring additional programming or additional user defined inputs. However, user defined inputs are required at various stages: (1) occasionally by altering a script or program, (2) more frequently by changing parameters within a “.csv” file, or (3) through interaction with the program when prompted in an interpreter.

Many, but not all of the procedures referred to above have been organized into a standard, step-by-step set of kNN LVI procedures, starting with Fuzzy C-Means classification, and ending with nearest neighbor assignments from a reference dataset to a target dataset, and including production of associated Y-variable statistics (Appendix II). Users can follow these steps that involve referring to the more detailed procedures as outlined in each of the key sections in the manual. This is somewhat cumbersome, but functional. To start with it is recommended that users go through the Appendix II procedures once, and in their entirety, using the demonstration dataset. This dataset has been reduced in size so that users can progress through the steps reasonably quickly.

There are a number of features in this software that need upgrading. Perhaps the greatest limitation is in processing speed, particularly when completing some of the nearest neighbor analysis including production of nearest neighbor statistics using a reference dataset; these programs can take days to run depending on the size of the dataset, the processing speed of your computer, and memory. This issue can be dealt with in the future by invoking multicore or multiprocessor processing. In addition, once one is familiar with the steps, the manual use of individual scripts is cumbersome and could be consolidated to reduce the amount of effort required by users. These and other upgrades will have to wait until completion of the next version.

## 2. Installation

The following instructions pertain to a Windows 7 machine.

### 2.1. Create an R Working Directory

1. Copy or unzip the [\\Rwd\\](#) directory into the top of your C drive.

### 2.2. Installations

1. R should be installed on your computer. If not run the R-2.15.3-win.exe in the [\\Rwd\\Python\\Intsallers\\](#) directory.
2. Python2.7 should be loaded in the top of your C: drive. If not the run the python-2.7.3.msi (32 bit) in the [\\Rwd\\Python\\Intsallers\\](#) directory.
3. Numpy should also be installed in the [\\Python27\\Lib\\site-packages\\](#) directory in the numpy folder. If not run the numpy-1.6.1-win32-superpack-python2.7.exe ) in the [\\Rwd\\Python\\Intsallers\\](#) directory.
4. Scipy should be installed in the : [\\Python27\\Lib\\site-packages\\](#) directory in the scipy folder. If not run the numpy-1.6.1-win32-superpack-python2.7.exe ) in the [\\Rwd\\Python\\Intsallers\\](#) directory.

### 2.3. Create R Working Directory & Load R packages

Open the R interpreter under by clicking the R icon under the R program file in your start menu.

Click *File*, and then *Open script* at the top RHS of the R interpreter and navigate to the [\\Rwd\\Rscript\\](#) directory.

1. Double click on the *SetRwd.R* script file. This will open the file in the interpreter.
2. Check the statement `setwd("C:\\Rwd")` to ensure that it corresponds with the location of the Rwd directory on your computer.
3. Click *Edit, Run all*.
4. A new file will now be stored in your Rwd directory called *OpenSession.RData*.
5. Close the R session by clicking on the x button in the top right hand corner; when asked to "Save workspace image?" press "No".
6. Go to the C:\\Rwd\\ directory, and double click on *OpenSession.RData* ... a new R session will open up in the interpreter.
7. Now if you once again click on *File, Open script* you will see that the interpreter automatically opens up in the Rwd directory. Once again close the R interpreter and don't save the workspace image (this is the routine procedure for ending a session).
8. Open (double click on) the R-script called *installPackages.R* – it should once again open up in the interpreter. In the RGui interface click on *Edit, Run all* ... and a box will open up asking you

## LVI Linear Discriminant Analyses: Beta Version

to select a CRAN mirror site near to your location (e.g. Canada(BC)) – after clicking on the location, press *OK* at the bottom of the box. A series of downloads will then occur, ensuring you have all of the necessary packages to run the routines identified below.

9. This script loads a number of R-packages into R – making various kinds of processes and analysis available to the user.

### 2.4. Python Modules

1. Copy the dbfpy file and the routineLviApplications.py module in [\\Rwd\\Python\\CopyToSitePackages\\](#) to the [\\Python27\\Lib\\site-packages\\](#) directory.
2. Open the Python27 Interpreter (IDLE) in the Windows start menu under *All Programs Python2.7* by clicking on *IDLE*.
3. Go to *File, Open* and navigate through the [\\Lib\\site-packages\\](#) to *routineLviApplications.py*. Doubleclick on that file to open it up. Near the top of the file you should see something like this:

```
#Identify admin code directory
adminPath = 'C:\\Rwd\\Python\\Admin\\'
filePath = 'C:\\Rwd\\'
dataFilePath = 'C:\\Rwd\\'
dictFilePath = 'C:\\Rwd\\Python\\DATDICT\\'
fileExtension = '.csv'
readErrorFilePath = 'C:\\Rwd\\Python\\PyReadError\\'
```

If your [\\Rwd\\](#) directory is not in the top of your C: directory then you will need to change these addresses. Rwd. These paths are used to direct the reading of certain Python modules and so too the reading and writing of .csv files. If you have a different location for the [\\Rwd\\](#) directory then correctly identify it by changing the Python code. To save it: click on *Run Check Module*. If there is an error then Python will tell you, otherwise you will find yourself back at the command line, `>>>`, in the interpreter (Python Shell).

4. Note that as a result of some legacy coding – if [\\Rwd\\](#) directory structure is not correct then the following files in the [\\Rwd\\Python\\](#) directory (with associated line numbers in parenthesis) need to also be changed to reflect the proper location of the Rwd directory (note the line numbers are indicated in the bottom right hand corner of the Python interpreter; after you are finished correcting the paths click on *Run* then *Check Module* in each case and as indicated previously):

- 4.1. COHENS\_KHAT.py (line numbers: 9, 10, 11; 24)
- 4.2. COMBINE\_EVALUATION\_DATASETS.py (8, 9, 10, and 11)
- 4.3. DbfFileRecovery.py (14, 15, 16, 17 ... this program is included but not referred to again in this document).
- 4.4. EXTRACT\_RVARIABLE\_COMBOS.py (15, 16, 17, 18)

### 3. Fuzzy C-Means Classification

1. Develop the data input file LVINEW and place in top of \\Rwd\\ directory. An example file is in the [\\Rwd\\Rdata\\Archived\\LVI\\Quesnel\\BaseData\\](#) directory. There are a few constraints on labeling files – all of the scripts are tied to certain file name conventions. The first one is that the reference data is always found in a file called LVINEW.csv. Each line in the file is a unique observation identified as a number under the key variable or column name of LVI\_FCoid.
2. Ensure that the data dictionary (PyRDataDict.csv in \\Rwd\\Python\\DATDICT\\ directory) is correctly filled in. There is a sample file in the \\DATDICT\\Sample\\ subdirectory that was developed for the Quesnel dataset. This dictionary is used by Python as part of the process for bringing data into memory and casting each variable according to a certain type as follows (the types listed in parenthesis are defined for user purposes but in terms of type conversions in Python they refer to only one of the three dominant types: integer, float, and string):
  - a. Integer (count, dummy, nominal)
  - b. float (continuous, proportion)
  - c. string

For the Quesnel LVINEW dataset the types are listed in the Sample dictionary starting in line 36 (TVID = 37). TVID is an id that identifies a unique combination of data Table names (listed under the TABLENAME column) and variable name within a given data table (listed under the VARNAME column). The variable names may be changed if so desired by filling in the “NEWVARNAME”. These variables will be used in the analysis and are the ones used in any new files that may be produced.

3. Update XVARSELV1 with list of variables in LVINEW (and LVINORM). XVARSELV always consist of 3 columns, TVID (this does not correspond with the TVID's in LVINEW), with some or all of the unique variable name in LVINEW. This table (XVARSELV1) is used to indicate whether each variable is selected as a candidate X-variable (XVARSEL = X), a Y-variable to be used in the Fuzzy C-Means clustering (XVARSEL = Y), or none of the above (XVARSEL = N). If the names in XVARSELV1 are not correctly identified, as specified in LVINEW then the programs will crash. Both LVINEW and XVARSELV1 are used when running Python and R scripts. Note also that XVARSELV1 does not have to include all of the variables in LVINEW – only those ones that are to be selected as eligible X-Variables or as Y-variables. Y-variables are used primarily to develop a system of classification and X-Variables are then selected to best identify the differences between classes based on the use of Discriminant analysis (maximize the ratio of between to within class differences).
4. Open FUZZYC\_INITIALIZATION.csv to set the fuzzy c-means classification parameters.

In particular select the desired classification routine (e.g. Euclidean distance by inputting FCM\_E in row 4, FID = 3, ROUTINE, under column C, VARVALUE). Also enter the range of classifications to be developed using the Y-variable dataset (specified in step 3) by entering the lower limit

## LVI Linear Discriminant Analyses: Beta Version

number of classes (LLNCLASS, row 5, FID = 4; enter integer in row under column C, VARVALUE) and upper limit number of classes (ULNCLASS, row 6, FID = 5; enter integer in row under column C, VARVALUE). Note that ULNCLASS must be  $\geq$  LLNCLASS. LLNCLASS must be  $\geq 2$ . If ULNCLASS == LLNCLASS then 1 system of classification will be produced with the number of classes equal to LLNCLASS.

5. Open the Python interpreter (IDLE).
6. Go to *File ... Open* and navigate to the `\\Rwd\\Python\\` directory; click on `LVI_CLASSIFY.py`; Go to *Run ... Run Module (F5)* to run the classification routine.
7. For each classification with N classes, the class assigned to each observation are written to `FCLASS.csv` (the file is overwritten) and the centroids associated with each of the selected Y-variables for each class in each system of classification are written to `FCENTROID.csv`.
8. The classes assigned to each of the observations should be manually transferred to `LVINEW` along with the corresponding labels (e.g. `CLASS5` indicating a classification with 5 classes) and a corresponding update of variables in `LVINEW` to the data dictionary (`PyRDataDict.csv`) referred to in step 2 above. Updates are not required to be made in `XVARSELV1`, but one may do so. If you look in the `LVINEW.csv` in the [\\Rwd\\Rdata\\Archived\\LVI\\Quesnel\\BaseData\\](#) directory you will see that a set of variables (column names) have been added as follows: `CLASS1`, `CLASS2`, ... `CLASS25`. These are the class assignments using `FCM_E` for 1 to 25 classes. Note also that these are based on the Y-variable dataset identified in `XVARSELV1` in that same directory.

## 4. Start with Variable Selection from a Large Number of Variables

1. *LoadDatasetAndAttachVariableNames.R*  
Load dataset and attach variable names.  
`DATA` in `LVINEW.csv`
2. *SelectObservationSubset.R*  
Optional  
Select observation subset.

Note that this is currently set to ensure that the `bec` zone (recorded in `LVINEW` under the variable name `LVI_BEZ`) is equal to a particular `BEC_ZONE`. In the Quesnel dataset this may be set to equal `'SBPS'`, `'SBS'` or `'MS'`. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

## LVI Linear Discriminant Analyses: Beta Version

### 3. *DeclareClassificationVariableAsFactor.R*

Declare (numerical) classification (Y-) variable as a “Factor.”  
Modify script to identify the factor, e.g. “CLASS6”

Note that this is an opportunity to change the classification Y-variable dataset – e.g. in the code where:

```
CLASSIFICATION = factor(CLASS5)
```

Change CLASS5 to CLASS510 (i.e. one of the optional classifications in LVINEW) to indicate which system of classification you wish to analyze.

### 4. *SelectXVariableSubset\_v1.R*

Select (independent) X variable subset.  
DATA in XVARSELV1.csv

Note that it is easy to confuse SelectXVariableSubset\_v1 with SelectYVariableSubset\_v1.

### 5. *Loadsubselect-R-Package.R*

Load R library package, subselect.

(Note if you wish to view the data and make changes to it you can run the following script:

```
ViewLviNewDataset.R
```

This will open up a new screen and allow you to view the data. However, if this is deployed using a large dataset it may cause the interpreter to become non-responsive since it tries to write all of the data into the interpreter once the data editor is closed.

### 6. *RunLinearDiscriminantAnalysis\_subselect\_IdaHmat.R*

Run IdaHmat in subselect package.

### 7. *Run-IdaHmat-VariableSelection-Improve.R*

Select chosen criteria and run IdaHmat variable selection routine.

Note that for variable selection you can select a range in the number of variables to be used in the model by changing the minimum (minNvar) and the maximum (maxNvar). You can also control the number of solutions that you would like to investigate for each number of variables (between minNvar and maxNvar, inclusive). Finally you can set different criteria for variable selection as follows:

- Roy’s first root statistic (“ccr12”)
- Wilks’ Lamda (“Wilkes”)
- Chi squared (“x12”)
- And the Zeta 2 coefficient (“zeta2”)

Currently of the alternatives are listed in the script file. To deselect a choice put a # in front of the line (R recognizes these as comments). To select a criterion, remove the # sign from the



## LVI Linear Discriminant Analyses: Beta Version

front of the line. Note that if two of the criteria are selected, the last one listed (toward the bottom of the script) will be the one used.

Note that if there are attributes (columns) with many zero's this may cause the variable selection algorithm to fail.

Note also that in the process of testing this algorithm, the following error statement was encountered when the number of classes exceeded 16:

**Error in if (maxabssym > tolsym) { :**

**missing value where TRUE/FALSE needed**

As a result the program does not complete properly. There may be an upper limit in the number of classes that can be handled using the subselect package improve function.

8. *ExtractVariableNameSubsets.R*

Extract all variable subsets derived from step 6 and put in data frame called SOLSUM (solution summary).

9. *WriteDataframeToCsvFile.R*

Write SOLSUM from step 7 to VARSELECT.csv in R working directory.

This creates a file called VARSELECT.csv. In the file are the following variables:

- 9.1.1. UID: A unique ID identifying each row.
- 9.1.2. MODELID: A model ID representing each model.
- 9.1.3. SOLTYPE: Solution type ... referring to the number of variables associated with each solution.
- 9.1.4. SOLNUM: Solution number associated with the number of solutions to be explored at each iteration. So for example if up to 5 solutions were to be identified (decision made when running script in step 7 above) , then this number will range from 1 to 5 for each solution type.
- 9.1.5. KVAR: This is the variable number, so for the selection of 1 variable, this is equal to 1, for two variables, variables numbered 1 and 2 are identified, and so on.
- 9.1.6. VARNUM: This refers to the variable number, or column number in the X-dataset (determined according to step 4 above)
- 9.1.7. VARNAME: This is the actual variable name associated with the variable number.

### 10. *ExtractRvariableCombos.py*

Python code

Using a Python module create a reformatted list of all of the unique combinations of variables and print it to a file called XVARSELV in the LVI directory; Run the following routine:

OUTPUT :

“.csv” comma delimited files

XVARSELV      contains the list of unique combinations of variables developed from running the variable selection routine *Ida.Hmat*.

UNIQUEVAR    contains a unique list of variable names compiled from all variable sets produced in *Ida.Hmat*.

Note that the output is assigned to the following directory: “E\\Rwd\\”

## 5. Run Discriminant Analysis for a Single X-Variable Set

### 1. *LoadDatasetAndAttachVariableNames.R*

Load Dataset

DATA in LVINEW.txt

### 2. *SelectObservationSubset.R*

Optional

Select observation subset

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI\_BEZ) is equal to a particular BEC\_ZONE. In the Quesnel dataset this may be set to equal ‘SBPS’, ‘SBS’ or ‘MS’. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

### 3. *DeclareClassificationVariableAsFactor.R*

Select and declare (numerical) classification variable as a “Factor.”

Modify script to identify the factor, e.g. “CLASS6”

Note that this is an opportunity to change the classification Y-variable if there are a number of different classifications you would like to investigate.

### 4. Set prior classification distribution as being uniform or as per sample:

#### 4.1. *ComputeUniformPriorClassificationProbabilityDistribution.R*

Uniform distribution

## LVI Linear Discriminant Analyses: Beta Version

### 4.2. *ComputeSamplePriorClassProbabilityDistribution.R*

Distribution according to sample

## 5. Select (independent) X variable subset

### 5.1. *SelecXVariableSubset\_v1*

DATA in XVARSELV1.csv

### 5.2. *SelectXVariableSubset\_v2*

DATA in XVARSELV2.csv

These are loaded in the `//Rwd//`

Note that the second version (v2) is designed to accommodate multiple variable selection; however, when used in this procedure only header and the first row may be entered, otherwise only the last variable set in the list will be used in the next step.

## 6. *LoadMASS-R-Package.R*

Load Discriminant Analysis R-package

## 7. Run Linear Discriminant Analysis:

### 7.1. *RunLinearDiscriminantAnalysis\_MASS\_Ida.R*

### 7.2. *RunLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R*

Notes:

In *RunLinearDiscriminantAnalysis\_MASS\_Ida.R* the Take-One-Leave-One option is disabled (CV = FALSE). As a result the following output is available following the discriminant analysis (note by typing the command, indicated in bold, the results can be printed out in the interpreter:

**lvi.Ida\$prior** this produces the prior probability distribution (established in step 3 above) used to represent the distribution of observations amongst the classes (CLASSIFICATION).

**lvi.Ida\$counts** the number of observations by class.

**lvi.Ida\$means** the mean for each X-variable by class

**lvi.Ida\$scaling** the discriminant functions are scaled so that the mean z-score for each function is 0. Note that this is equivalent to subtracting observed X-variable value from the mean for each of the variables and then multiplying by the discriminant functions.

**lvi.Ida\$svd** the ratio's of between to within-group standard deviations in the linear discriminant variables. These are also referred to as eigenvalues. The squares

## LVI Linear Discriminant Analyses: Beta Version

of these figures are the canonical F-statistics. When the squares of these figures are converted into proportions of the total – this is equivalent to the proportion of the total (Between-to-within) variance explained by each discriminant function.

**lvi.Ida\$N** is the number of observations contained in the dataset.

Within this same routine the following additional output is also available:

**class.pred\$class** (the predict function) produces the class assignments to each observation (with all observations used in the discriminant analysis); also at the bottom of this output, the unique class names (or numbers) are listed.

**class.pred\$posterior** the estimated posterior probability distributions based on the prior distribution calculated as follows (see Hora and Wilcox 1982, Dillon and Goldstein 1984 pp. 392 – 393).

**class.pred\$x** the scores for each if the test cases associated with each variate (function)

**class.table** the (table function produces a) classification contingency table with the original class distribution in rows and the predicted class distribution in columns.

In *RunLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R* the Take-One-Leave-One option is enabled (CV =TRUE). The following output may be obtained:

**lvi.Ida\$class** the class assigned to each observation

**lvi.Ida\$posterior** the posterior probabilities developed using Take-One-Leave-One; these are superior to those not involving the Take-One-Leave-One process (Bates and Wilcox 1982; Dillon and Goldstein 1984, pp. 406-409).

**class.table** the (table function produces a) classification contingency table with the original class distribution in rows and the predicted class distribution in columns.

## 6. Run Linear Discriminant Analysis for Multiple X-Variable Sets

### 1. *LoadDatasetAndAttachVariableNames.R*

Load dataset  
DATA in LVINEW.csv

### 2. *SelectObservationSubset.R*

Optional  
Select observation subset.

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI\_BECZ) is equal to a particular BEC\_ZONE. In the Quesnel dataset this may be

## LVI Linear Discriminant Analyses: Beta Version

set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

### 3. *DeclareClassificationVariableAsFactor.R*

Declare (numerical) classification variable as a "Factor."

Modify script to identify the factor, e.g. "CLASS6"

Note that this is an opportunity to change the classification Y-variable if there are a number of different classifications you would like to investigate. If this process is used after having run process number 4 (Start with Variable Selection from a Large Number of Variables) then make sure that the correct (same) class label is selected as before.

### 4. Compute prior classification distribution as being uniform or as per sample

#### 4.1. *ComputeUniformPriorClassificationProbabilityDistribution.R*

Uniform distribution

#### 4.2. *ComputeSamplePriorClassProbabilityDistribution.R*

Distribution according to sample

### 5. *WritePriorDistributionToFile.R*

Write prior distribution to file

OUTPUT (.csv files)

PRIOR: Contains a list of classes (CLASS) and associated prior probabilities (PROIRD) of occurrence.

### 6. *LoadMASS-R-Package.R*

Load Discriminant Analysis R-package

### 7. *SelectXVariableSubset\_v2.1*

Select (independent) X variable subset

DATA in XVARSELV.csv

### 8. *RunMultipleLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R*

Run (Multiple) Linear Discriminant Analysis for Multiple Sets of Variables – Take One Leave One

WARNING this is easily confused with another script that will not work in this context (missing the "Multiple"):

- *RunLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R*

Note also that this routine uses the Take-One-Leave-One routine for the purpose of rating the quality of the variable sets in terms of their accuracies in classification based on producing

## LVI Linear Discriminant Analyses: Beta Version

contingency tables as 1 output, and in terms of the posterior estimation of error as another output.

Note that the posterior error of estimation is calculated following the procedures of Hora and Wilcox (1982; Equation 9):

$$\hat{e} = 1 - N^{-1} \sum_{i=1}^N \max [P(Y | X_i)] \quad \text{Eq. 1}$$

Where,

$\hat{e}$  is the estimated (posterior) error

$N$  Is the total number of observations

$P(Y | X_i)$  is the probability of class Y, where Y is equal to 1 to m classes, given a set of variables,  $X_i$ , where i equals 1 to n observations.

### 9. *WriteMultipleLinearDiscriminantAnalysis\_MASS\_Ida\_TOLO\_File.R*

Write (Multiple Linear) Results from Step 6 to Files

OUTPUT (.csv files)

CTABULATION: contains the contingency table data from which the Cohen's (1960) Coefficient of Agreement can be calculated. VARSET refers to the variable sets in XVARSELV.

REFCLASS: refers to the reference or actual class. PREDCLASS is the predicted class and CTAB is a cross tabulation indicating the number of observations in the associated combination of REFCLASS and PREDCLASS.

POSTERROR: contains the errors of estimation (UERROR) using Eq. 1 above for each variable set (VARSET) with an associated number of variables (NVAR).

### 10. *RunMultipleLinearDiscriminantAnalysis\_Mass\_Ida.R*

Run (Multiple) Linear Discriminant Analysis for Multiple Sets of Variables

All observations included

Note this can be confused with *RunLinearDiscriminantAnalysis\_Mass\_Ida.R* ... missing the "Multiple."

### 11. *WriteMultipleLinearDiscriminantAnalysis\_MASS\_Ida.R*

Write Multiple Linear results from step 10 to files.

OUTPUT (.csv files)

CTABALL: contains the contingency table data from which the Cohen's (1960) Coefficient of Agreement can be calculated. VARSET refers to the variable sets in XVARSELV. REFCLASS refers to the reference or actual class. PREDCLASS is the

## LVI Linear Discriminant Analyses: Beta Version

predicted class and CTAB is a cross tabulation indicating the number of observations in the associated combination of REFCLASS and PREDCLASS.

- VARMEANS:** contains the mean values for each variable by class. VARSET2 refers to VARSET in XVARSELV. CLASS2 is the reference or actual class. VARNAMES2 refers to the variable names in the original dataset and as selected for inclusion in the X-variable set. MEANS2 refers to the mean value associated with each variable (VARNAMES2) and class (CLASS2) combination. Note that the same variable may occur in several variable sets – producing redundancies.
- DFUNCT:** contains the discriminant functions for each axis and combination of variables. VARSET3 refers to VARSET in XVARSELV. VARNAMES3 refers to the variable names in the original dataset and as selected for inclusion in the X-variable set. FUNCLABEL3 refers to the discriminant functions in order of priority (ranked from highest to lowest eigenvalue as follows: LN1, LN2, ... up to n-1 functions where n is equal to number of classes or the number of variables selected for inclusion in an equation, whichever is the lesser of the two.
- BWRATIO:** contains the between-to-within variance ratio (BTWTWCR4; i.e. eigenvalues) of the differences in class Z-statistics associated with each discriminant function. VARSET4 refers to VARSET in XVARSELV. FUNCLABEL4 is identical to FUNCLABEL3 in DFUNCT.

### 12. COHENS\_KHAT.py

Python

Compile Take-One-Leave-One CTABULATION classification accuracy statistics.

Data in CTABULATION.csv

Note that the input file was produced using step 7 in these procedures.

OUTPUT (.csv files in Rwd directory)

**CTABSUM** provides statistics as indicated in Table 2. VARSET refers to VARSET in XVARSELV.

Table 2. A description of variables included in the output file: CTABSUM.

Variable	Name
VARSET	Variable Set
OA	Overall Accuracy
KHAT	Coefficient of Agreement
MINPA	Minimum Producer Accuracy
MAXPA	Maximum Producer Accuracy
MINUA	Minimum User Accuracy
MAXUA	Maximum User Accuracy

## LVI Linear Discriminant Analyses: Beta Version

A variable set is associated with a given combination of different kinds and numbers of variables that as developed in steps 5, 6 and 7 above. The overall accuracy indicates the proportion of observations that were correctly classified according to the original (reference) classification. Cohen's coefficient of agreement is an indicator of the overall success rate after having removed the potential for a certain level of agreement to occur by chance.

The minimum producer accuracy indicates the minimum number of correctly classified observations given the total number of observations assigned to any given class, amongst all classes by way of discriminant analysis in this case, and expressed as a proportion. The maximum producer accuracy is similarly derived but with respect to the maximum. These figures are also related to the maximum and errors of omission amongst all of the classes (e.g.  $1 - \text{MINPA}$ , and  $1 - \text{MAXPA}$ ).

The minimum user accuracy indicates the minimum number of correctly classified observations given the total number of observations as originally assigned to any given class, amongst all classes by way of discriminant analysis in this case, and expressed as a proportion. The maximum user accuracy is similarly derived but with respect to the maximum. These figures are also related to the maximum and errors of commission amongst all of the classes (e.g.  $1 - \text{MINPA}$ , and  $1 - \text{MAXPA}$ ).

## 7. Produce Unique X-Variable Subset Correlation Matrix

Note that as a guideline you may wish to exclude any one variable in pairs with correlations  $> 0.8$  (or  $< -0.8$ ) a priori. The step of removing correlated variables may best be done before starting data processing. The reason it is suggested that it be done after variable selection is to reduce the number of variable combinations that must be considered in terms of Pearson correlation coefficients.

1. *LoadDatasetAndAttachVariableNames.R*

Load dataset  
DATA in LVINEW.csv

2. *SelectObservationSubset.R*

Optional  
Select observation subset

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI\_BECZ) is equal to a particular BEC\_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. *SelectUniqueXVariableSubset.R*

Select unique X variable subset  
DATA in UNIQUEVAR.csv



## LVI Linear Discriminant Analyses: Beta Version

4. *CompileUniqueXVariableCorrelationMatrixSubset.R*  
Compile Unique Variable Correlation Matrix
5. *CreateUniqueVarCorrelationMatrixFileForPrinting.R*  
Create a unique variable correlation matrix file for printing
6. *SelectXVariableSubset\_v2.1.R*  
Select new X variable subset version 2.1  
This must be included prior to next step  
DATA in XVARSELV21.csv
7. *AddVariableSubsetIndicatorsToCorrelationMatrix.R*  
Add variable subset indicators to correlation matrix file if option 5.1 selected

Note that this routine labels each variable indicator set as I1, I2, I3 ... in the order that they are produced in step 5.

8. *WriteUniqueVarCorrelationMatrix.R*  
Write unique variable correlation matrix to a file

OUTUPT (.csv file)

UCORCOEF: This produces a table of correlation coefficients and indicator variables for each variable subset with 1's assigned to variable pairs that exist in the subset, and 0's assigned to all other variable pairs. VARNAME1 and VARNAME2 refer to the variable names in the original dataset and in the X-dataset. CORCOEF refers to the Pearson correlation representing the correlations between the VARNAME1 and VARNAME2 pairs. The I1, I2, ... variables are indicators for the variable combinations included (1) or excluded (0) from any given variable set. This provides the user with an opportunity to look at specific variables sets (VARSET in VARSELV).

MINMAXCOR: This is a compilation of the maximum (MAXCOR) and minimum correlations (MINCOR) within each variable set across all variable pairs (excluding identical pairs for which the correlations are 1) within each variable subset (VARSET) in XVARSELV. This provides the opportunity to look through the variable sets to see if there are any tow variables within each of those sets that may have excessively high (positive) or low (negative) correlations such that one of the two variables should be eliminated from inclusion in the dataset.

## 8. Produce Original Classification Unique X- or Y-Variable Subset Box and Scatter Plots

1. *LoadDatasetAndAttachVariableNames.R*

Load dataset  
DATA in LVINEW.csv

2. *SelectObservationSubset.R*

Optional  
Select observation subset

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI\_BECZ) is equal to a particular BEC\_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. *DeclareClassificationVariableAsFactor.R*

Declare (numerical) classification (Y-) variable as a "Factor"  
Modify script to identify the factor, e.g. "CLASS6"

4. Select unique X or Y variable subset

4.1. *SelectUniqueXVariableSubset.R*

DATA in UNIQUEVAR.csv

4.2. *SelectXVariableSubset\_v1.R*

DATA in XVARSELV1.csv

4.3. *SelectYVariableSubset\_v1.R*

DATA in XVARSELV1.csv

5. *LoadGraphics-R-Package.R*

Load R package – graphics

6. Run the box plot script

6.1. *CreateUniqueVariableClassificationBoxPlots.R*

If X variables have been selected.

6.2. *CreateYVariableClassificationBoxPlots.R*

If Y-variables were selected.

Note that you must right-click on each graph to proceed to the next graph. The first graph will be blank.

## LVI Linear Discriminant Analyses: Beta Version

### 7. *CreateUniqueVariableScatterPlots.R*

Run the scatter plot script

Note that this can produce a lot of graphs. Specifically if there a total of K unique variables, then  $(k*(K-1))/2$  graphs will be produced. For 20 variables that is equal to 190 graphs. Having said that you can end the graphics session at any time, by closing the graphics box (right clicking on the “X” button in the top right hand corner of the window).

The figures illustrate the scatter of observations for y-axis versus x-axis variables. A regression line (red, y vs. x) and a lowess line (blue; a locally weighted polynomial regression line – similar to a moving average) are also shown on each figure to facilitate interpretation of the trends.

### 8. Produce X vs. Y variable scattergrams

#### 8.1. *SelectYVariableSubset\_v1.R*

X-variables have already been selected.

Select Y-variables

DATA in XVARSELV1

#### 8.2. *SelectUniqueXVariableSubset.R*

Y-variables have already been selected.

Unique X-variable set is desired.

DATA in UNIQUEVAR.csv

#### 8.3. *SelectXVariableSubset\_v1.R*

Y-variables have already been selected.

Larger X-variable subset desired.

DATA in XVARSELV1.csv

### 9. *CreateYvXVariableScatterPlot.R*

Produce Y versus X scatter plots

## 9. Combine Evaluation Datasets

### 1. *COMBINE\_EVALUATION\_DATASETS.py*

Combine the following datasets for purposes of overall assessment

New file created: ASSESS.csv

Table 3. A list of files combined into one file: ASSESS.csv.

Input	Description
PyRDataDict.csv	This contains the data types (e.g. string, float, integer) associated with each of the input tables, except XVARSELV
MINMAXCOR	This contains the minimum and maximum correlation coefficients amongst all pairs of variables contained within a variable set.
CTABSUM	See Table 2.
POSTERIOR	See Eq. 1 above.
XVARSELV	The actual variable sets produced during the variable selection process.

This summary includes the following information:

VARSET	This is the variable set number in VARSELV.
VARNO	The number of variables associated with a given variable set.
MAXCOR	The maximum Pearson correlation coefficient amongst all pairs of variables in a VARSET.
MINCOR	The minimum Pearson correlation coefficient amongst all pairs of variables in a VARSET.
OA	The overall accuracy (The percent of corresponding predicted vs. actual class assignments).
KHAT	Cohen's Coefficient of Agreement (Equal to the overall accuracy minus the probability that such correspondences could have occurred by chance).
MINPA	The minimum Producer accuracy (i.e. the minimum in the number of predicted values that are correct divided by the total number of observations predicted to be within a given class; across all classes).
MAXPA	The maximum Producer accuracy (i.e. the maximum in the number of predicted values that are correct divided by the total number of observations predicted to be within a given class; across all classes).
MINUA	The minimum User accuracy (i.e. the minimum in the number of predicted values that are correct divided by the total number of observations observed to be within a given class; across all classes).
MAXU	The maximum User accuracy (i.e. the maximum in the number of predicted values that are correct divided by the total number of observations observed to be within a given class; across all classes).
NVAR	The number of variables in the variable set.
UERROR	The estimated error using equation 1 above.

## LVI Linear Discriminant Analyses: Beta Version

MODELID	This refers to the MODELID in VARSELECT.csv. Note that not all MODELID's are included in the list since some of the models in VARSELECT.csv are not unique in their variable selections and associated discriminant functions.
NMODELS	The number of models with the same variable selections.
XVAR1 ...	The names of variables associated with a given VARSET.

### 10. Alternative Variable Selection Procedure (Under Development)

1. *LoadDatasetAndAttachVariableNames.R*

Load dataset

DATA in LVINEW.csv

2. *SelectObservationSubset.R*

Optional

Select observation subset.

Note that this is currently set to ensure that the bec zone (recorded in LVINEW under the variable name LVI\_BECZ) is equal to a particular BEC\_ZONE. In the Quesnel dataset this may be set to equal 'SBPS', 'SBS' or 'MS'. Other variables and variable names may be applied. The script also reduces the original dataset down to a particular set of observations. If a different set of observations are desired then the process must be started from the beginning. Also issues may arise where certain zones or subzones do not have enough observations to support.

3. *DeclareClassificationVariableAsFactor.R*

Declare (numerical) classification (Y-) variable as a "Factor"

Modify script to identify the factor, e.g. "CLASS6"

4. Compute prior classification distribution as being uniform or as per sample

4.1. *ComputeUniformPriorClassificationProbabilityDistribution.R*

Uniform distribution

4.2. *ComputeSamplePriorClassProbabilityDistribution.R*

Distribution according to sample

5. *SelectXVariableSubset\_v1.R*

Select X variable subset (v1)

6. *LoadklaR-R-Package.R*

Load klaR Package

7. *LoadCombinat-R-Package.R*

Load combinat Package

8. *RunLinearDiscriminantAnalysis\_klaR\_pvs.R*  
Run pairwise class variable selection

Note that this routine uses linear discriminant analysis (or alternatives such as quadratic or reduced discriminant procedures – making it more flexible than the subselect package which only provides for the standard linear discriminant analysis procedure). The procedure then compares each possible pair of classes in turn (using the pvs command in klaR) and selects the best variable sets according to certain criteria. The basic criteria are “stepclass” (forward, backward, or both), “ks.test” (Kolmogorov-Smirnov test), or “greedy.wilks” (Wilks’ lambda). This procedure could be applied within a bootstrap procedure to generate multiple variable sets for further testing.

## 11. Calculate Z-Scores, Nearest Neighbours, and Root Mean Squared Errors.

1. *NN\_ZSCORE.py*  
Python  
Apply the discriminant functions to the reference dataset.  
Input files:
  - DATA in LVINEW.csv
  - DFUNCT.csv
  - BWRATIO.csv

Note that this routine can take a long time to run. A summary of analysis steps carried out in this routine is as follows:

- 1.1. For each variable set, compute the Z-scores for each observation for each of the associated discriminant functions. Note that these functions were derived from the reference dataset; bootstrapping has not been invoked; nor a take-one-leave-one strategy.
- 1.2. Normalize the Z-scores (subtract from the mean and divide by the standard deviation for each function).
- 1.3. An option is then provided to weight the results based on the proportion of between-to-within variance explained by each discriminant function associated with a variable set. This option is embedded in the program for identifying nearest neighbours. However preliminary tests of this option indicate no differences in the results when using Mahalanobis distances primarily because this distance metric immediately restores the weight assigned in proportion to the inverse of the covariance matrix. Furthermore the Mahalanobis distances tended to consistently produce the lowest root mean squared errors associated with nearest neighbours when compared with the use of absolute or Euclidean distances. In this case all of the observations were used to derive the discriminant function but each observation was then, in turn, excluded from having itself identified as being the nearest neighbor. As a consequence the current default is set so that the Z-values assigned by the discriminant function are given equal weight in determining the nearest neighbor.

## LVI Linear Discriminant Analyses: Beta Version

Under this scenario Euclidean and Mahalanobis distances produce equivalent results; absolute differences are generally associated with higher nearest neighbor root mean squared errors. The option is always there to test this further with other datasets.\

- 1.4. Identify nearest neighbours for each reference observation (excluding identification of self) and calculate root mean squared errors for the associated Y-variable set.

### OUTPUT (.csv files)

ZSCORE	This file contains the ZSCORES derived from the discriminant functions in DFUNCT.csv for each variable set (VARSET3 equivalent to VARSET in VARSELV.csv) and each observation (LVI_FCoid) in LVINEW.csv ... the initial dataset used as input to the entire set of procedures. LN1, LN2 ... represent the discriminant functions in DFUNCT.csv; the estimated ZSCORES associated with each of these functions are listed in the columns below each of these headings.
ZNSCORE	This file contains the same information as in ZSCORE but the ZSCORES in ZSCORE.csv have been normalized (subtracting the mean z-score for all observations within a VARSET from each observation (LVI_FCoid) value and then dividing by the standard deviation.
ZSTAT	This file contains the estimated means (MEAN) and standard deviations (SDEV) associated with a given variable set (VARSET3) and discriminant function (FUNCLABEL3). These data are used in normalizing the figures in ZSCORE to produce ZNSCORE.
ZNNA	For each variable set (VARSET3): Using the normalized ZNSCORES the k nearest reference neighbours (NNOBS; where k is user defined as part of running NN_ZSCORE.py) are identified in relation to a given reference observation (REFOBS; a given reference observation cannot be identified as a nearest neighbor to itself). The distance of each nearest neighbor (DIST) is also indicated by summing up the absolute differences in the ZNSCORES across all discriminant functions, for each variable set -reference observation-nearest neighbor combination.
ZNNE	This is analogous to ZNNA, but in this case Euclidean distances are used instead.
ZNNM	This is analogous to ZNNA but in this case Mahalanobis distances are used instead. Note that because ZNSCORE is used (already adjusted for the mean and standard deviation) this will produce results very similar to Euclidean distances unless there is significant covariance in ZNSCORES across the discriminant functions.
ZARMSE	For each variable set (VARSET) and number of nearest neighbours (KNN = 1, 2 ... k nearest neighbours where the maximum number of neighbors to be evaluated is user defined when running NN_ZSCORE.py) and associated Y-variable name (VARNAME; as indicated in VARSELV1 by manually entering a "Y" next to the

## LVI Linear Discriminant Analyses: Beta Version

variable name; these entries are usually kept consistent with those used to develop the classes in the first place but users may choose to change them if they wish) the following statistics are reported:

RMSE	For each VARSET, KNN and VARNAME: This is the sum of: squared difference between the Y-values and the mean Y-value across all k-nearest neighbor for each reference observation and then divided by the number of nearest neighbors minus 1, summed across all observations and then divided by the total number of observations. By definition, for $k = 1$ this value is equal to 0.
BIAS	For each VARSET - KNN – Y-variable: This is the difference between the mean estimated Y-value given k nearest neighbours minus the actual value associated with each reference observation, summed across all reference observations and then divided by the total number of reference observations. A positive number indicates that on average a given variable is over-estimated by a certain amount (in the same units as the original Y-variable). A negative number indicates that on average a variable is under-estimated.
BRMSE	For each VARSET, KNN and VARNAME: This is the sum of: squared difference between the Y-values and the actual reference Y-value across all k-nearest neighbors for each reference observation, and then divided by the number of nearest neighbours minus 1, before being summed across all observations and then divided by the total number of observations. This figure varies from RMSE because it includes the effect of BIAS as part of the estimation of BRMSE.

## 12. Calculate Nearest Neighbors, and Root Mean Squared Errors Using Selected X-Variables (without Z-Scores).

### 1. *NN\_XDIS.py* Python

This program produces a set of outputs completely analogous to those described under section 11 above. The difference is that instead of using normalized Z-scores associated with each variable set and associated discriminant function, it uses normalized X-variables to determine the nearest neighbours. As before the measures of similarity and differences amongst prospective nearest neighbours are based on absolute values of the difference, Euclidean distances, and Mahalanobis distances. All of the same files are printed out as outlined in section 11 above, except the file names are prefaced by “X” instead of “Z”. For example the file called ZSCORE ( a .csv file) is called XSCORE using the X-variable names to label the columns instead of the discriminant function labels. The purpose of this analysis is to determine whether or not for example, Euclidean distances amongst the X-variables provides a better assessment of nearest neighbors when compared with using Z-Scores.



### 13. Calculate Nearest Neighbors, and Root Mean Squared Errors Using Selected Y-Variables

1. *NN\_YDIST.py*  
Python

This follows the same pattern as that described in sections 12 and 13, except that the Y-variables remain constant across all variable sets. This established a lower bound in terms of the Root Mean Squared Errors that can be obtained based on the sample population and therefore can be used as a benchmark against the results. It represents the best results that can potentially be obtained given the sample population.

### 14. Identify k-Nearest Neighbours in Target Dataset

1. *NN\_TARG\_REF.py*  
Python

This routine identifies k-nearest neighbours where k is specified as user input. It will also prompt the user to select one or more variable sets (VARSET) as identified in VARSELV.csv. The program prints out one or more files: NNTARGETxx.csv where “xx” refers to the variable set number(s) used to generate each file. The file lists the Target observation numbers (TARGOBS), the nearest neighbours in order of priority starting from the nearest neighbor, e.g. (NN1, NN2 .. NNk) and the associated distances based on the normalized Z-Scores (DIST1, DIST2, ... DISTk).

### 15. Compile the kNN Target Dataset Y-Variable Statistics

1. *COMPILE\_NN\_TARGET\_YSTATS.py*  
Python

This routine compiles the Y-variable statistics for the Y-variables listed in XVARSELV1. The user is prompted to input the nearest neighbour filename number, associated with one of the output files (NNTARGETxx) described in section 14, before it begins processing. A “.csv” output file is created as follows: “TNNSTATS64”. This file list contains information as follows:

TARGOBS	The target observation key numbers.
VARNAME	The Y-variable names associated with each TARGOBS
KNN	The number of k-nearest neighbours associated with a given TARGOBS and VARNAME.
MEAN	The mean value assigned to each VARNAME associated with a given TARGOBS and KNN.

SDEV            The standard deviation assigned to each VARNAME associated with a given TARGOBS and KNN.

### 16. Convert TNNSTATS file VARNAMEs into Column Format

1. *PIVOT\_TABLE.py*  
Python

This routine converts the VARNAMEs that are listed in one column in NNTARGETxx uses each of the names as columns instead in a '.csv' file called TNNSTATSPxx. This is for convenience only. Users are asked to input the file number as per step 15, and this same number is appended to TNNSTATSP. Two different TNNSTATP files are produced:

TNNSTATxx\_MEAN      This contains all of the mean values for each selected Y-variable and for k is equal to 1, 2, 3 ... n. The "xx" represents a selected nearest neighbor model number (i.e. variable set) used to determine the nearest neighbours.

TNNSTATxx\_SDEV      This contains all of the standard deviations for each selected Y-variable and for k is equal to 1, 2, 3 ... n. The "xx" represents a selected nearest neighbor model number (i.e. variable set) used to determine the nearest neighbours.

### 17. References

Bezdek, J.C. 1981. Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York. Pp. 256.

Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement XX(1): 37 – 46.

Duda, R.O., Hart, P.E., and Stork, D.G. 2001. Pattern recognition. Second Edition. John Wiley & Sons Inc., New York. Pp. 654.

Dillon, W.R., and Goldstein, M. 1984. *Multivariate analysis*. Methods and applications. John Wiley & Sons, New York, NY, US.

Hora, S.C. and Wilcox, W.B. 1982. Estimation of error rates in several-population discriminant analysis. *Journal of Marketing Research* 19(1):57-61.

Wulder, M.A., White, J.C., Hay, G.J., and Castilla, G. 2008. Pixels to objects to information: Spatial context to aid in forest characterization with remote sensing. In: Blashke, T., Lang, S., Hay, G.J. (Eds.). Object-based image analysis – spatial concepts for knowledge driven remote sensing applications. Springer Verlag. Pp 19.

### Additional References

Birkal, D. 2006. Regularized Discriminant Analysis. Lecture Notes. <http://www.uni-leipzig.de/~strimmer/lab/courses/ss06/seminar/slides/daniela-2x4.pdf> [accessed Dec 19 2012]

Davis, H.Z., Mesznik, R. and Lee, J.Y. 2009. Finding an internal optimum in the classification of management accounting information: The role of fuzzy sets. *Management Accounting* 17:203-216.

Liu, H. and Yu, L. 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering* 17(4):491-502.

## Appendix I: A listing of Python Modules and R Scripts and the R Working Directory Structure

### *The Rwd Working Directory Structure*

This is a dictionary created to maintain LVI data management processes. It is created separately from the location of the R interface, packages, etc.

1. E:
  - 1.1. Rwd
    - 1.1.2. Python
      - 1.1.2.1. Admin
        - 1.1.2.1.1. addToDataMachette20121026.py
        - 1.1.2.1.2. BASE.py
        - 1.1.2.1.3. DbfFileRecovery.py
        - 1.1.2.1.4. dictionaryDBUtilities.py
        - 1.1.2.1.5. fileUtilities.py
        - 1.1.2.1.6. fuzzyC\_v2.py
        - 1.1.2.1.7. innerJoinDictDBs.py
        - 1.1.2.1.8. makeDD.py
        - 1.1.2.1.9. MAN.py
        - 1.1.2.1.10. MATRIX\_MAN.py
        - 1.1.2.1.11. PRINTv1.py
        - 1.1.2.1.12. readCSV.py
        - 1.1.2.1.13. READv1.py
        - 1.1.2.1.14. typeDataset.py
      - 1.1.2.2. CopyToSitePackages
        - 1.1.2.2.1. dbfpy (file folder)
        - 1.1.2.2.2. routineLviApplications.py
      - 1.1.2.3. DATDICT
        - 1.1.2.3.1. DemoDict
        - 1.1.2.3.2. QuesnelDictionary
        - 1.1.2.3.3. PyRDataDict.csv (initially a copy of the one in DemoDict)
      - 1.1.2.4. Installers
        - 1.1.2.4.1. Matplotlib-1.2.1rc1.win32-py2.7.exe
        - 1.1.2.4.2. Numpy-1.6.1-win32-superpack-python2.7.exe
        - 1.1.2.4.3. Python-2.7.3.msi
        - 1.1.2.4.4. R-2.15.3-win.exe
        - 1.1.2.4.5. Scipy-0.9.0-win32-superpack-python2.7.exe
      - 1.1.2.5. Other
        - 1.1.2.5.1. INTERACTIVE\_DATA\_LOADER.py
      - 1.1.2.6. PyReadError (initially empty)
      - 1.1.2.7. COHENS\_KHAT.py
      - 1.1.2.8. COMBINE\_EVALUATION\_DATASETS.py
      - 1.1.2.9. COMBINE\_NN\_TAREGET\_YSTATS.py

## LVI Linear Discriminant Analyses: Beta Version

- 1.1.2.10. CONVERT\_DBF\_TO\_CSV\_FILE.py
- 1.1.2.11. EXTRACT\_RVARIABLE\_COMBOS.py
- 1.1.2.12. GET\_SPECIES\_REPORT.py
- 1.1.2.13. LVI\_CLASSIFY.py
- 1.1.2.14. NN\_TARG\_REF.py
- 1.1.2.15. NN\_XDIST.py
- 1.1.2.16. NN\_YDIST.py
- 1.1.2.17. NN\_ZSCORE.py
- 1.1.2.18. PIVOT\_TABLE.py
- 1.1.3. Rdata
  - 1.1.3.1. Archived
    - 1.1.3.1.1. LVI
      - 1.1.3.1.1.1. Demo
        - 1.1.3.1.1.1.1. InputFiles
        - 1.1.3.1.1.1.2. OutputFiles
      - 1.1.3.1.1.2. Quesnel
        - 1.1.3.1.1.2.1. BaseData
        - 1.1.3.1.1.2.2. ReferenceOriginal
        - 1.1.3.1.1.2.3. ReferenceResults
        - 1.1.3.1.1.2.4. TargetOriginal
        - 1.1.3.1.1.2.5. TargetResults
      - 1.1.3.1.1.3. WLake
- 1.1.4. Rdocs
  - 1.1.4.1. DataAnalysis
    - 1.1.4.1.1. LOGISTIC
    - 1.1.4.1.2. LVI\_KNN\_MANUAL
    - 1.1.4.1.3. MANOVA
    - 1.1.4.1.4. MDA
    - 1.1.4.1.5. MISC
    - 1.1.4.1.6. MULTICOLLINEARITY
  - 1.1.4.2. Packages
  - 1.1.4.3. ReplacementFiles
  - 1.1.4.4. R-Language
- 1.1.5. RScript
  - 1.1.5.1. Additional AddVariableSubsetIndicatorsToCorrelationMatrix.R
  - 1.1.5.2. CompileUniqueVariableCorrelationMatrixSubset.R
  - 1.1.5.3. ComputeSamplePriorClassProbabilityDistribution.R
  - 1.1.5.4. ComputeUniformPriorClassificationProbabilityDistribution.R
  - 1.1.5.5. CreateUniqueVarCorrelationMatrixFileForPrinting.R
  - 1.1.5.6. CreateUniqueVariableClassificationBoxPlots.R
  - 1.1.5.7. CreateUniqueVariableScatterPlots.R
  - 1.1.5.8. CreateYVariableClassificationBoxPlots.R
  - 1.1.5.9. CreateYvXVariableScatterPlots.R
  - 1.1.5.10. DeclareClassificationVariableAsFactor.R
  - 1.1.5.11. ExtractVariableNameSubsets.R
  - 1.1.5.12. installPackages.R

## LVI Linear Discriminant Analyses: Beta Version

- 1.1.5.13. LoadCombinat-R-Package.R
- 1.1.5.14. LoadDatasetAndAttachVariableNames.R
- 1.1.5.15. LoadGraphics-R-Package.R
- 1.1.5.16. LoadklaR-R-Package.R
- 1.1.5.17. LoadMASS-R-Package.R
- 1.1.5.18. LoadR-Packages.R
- 1.1.5.19. Loadsubselect-R-Package.R
- 1.1.5.20. Run-IdaHmat-VariableSelection-Improve.R
- 1.1.5.21. RunLinearDiscriminantAnalysis\_klaR\_pvs.R
- 1.1.5.22. RunLinearDiscriminantAnalysis\_MASS\_Ida.R
- 1.1.5.23. RunLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R
- 1.1.5.24. RunLinearDiscriminantAnalysis\_subselect\_IdaHmat.R
- 1.1.5.25. RunMultipleLinearDiscriminantAnalysis\_Mass\_Ida.R
- 1.1.5.26. RunMultipleLinearDiscriminantAnalysis\_MASS\_Ida\_TakeOneLeaveOne.R
- 1.1.5.27. RunQuadraticDiscriminantAnalysis\_klaR\_qda.R
- 1.1.5.28. SelectUniqueXVariableSubset.R
- 1.1.5.29. SelectObservationSubset.R
- 1.1.5.30. SelectXVariableSubset\_v2.1.R
- 1.1.5.31. SelectXVariableSubset\_v2.R
- 1.1.5.32. SelectXVariableSubset\_v1.R
- 1.1.5.33. SelectYVariableSubset\_v1.R
- 1.1.5.34. SetRwd.R
- 1.1.5.35. ViewLviNewDataset.R
- 1.1.5.36. WriteDataframeToCsvFile.R
- 1.1.5.37. WriteMultipleLinearDiscriminantAnalysis\_MASS\_Ida\_to\_File.R
- 1.1.5.38. WriteMultipleLinearDiscriminantAnalysis\_MASS\_Ida\_TOLO\_to\_File.R
- 1.1.5.39. WritePriorDistributionToFile.R
- 1.1.5.40. WriteUniqueVarCorrelationMatrix.R
- 1.1.6. FUZZYC\_INITIALIZATION.csv (demo version)
- 1.1.7. LVI\_NEW.csv (demo version; reference dataset)
- 1.1.8. OpenSession.RData (To be recreated at time of installation)
- 1.1.9. QTARGET.csv (demo version; target dataset)
- 1.1.10. XVARSELV1.csv (demo version; used to select X and Y Variables)

*R-packages included for installation in installPackages.R (1.1.5.12 above)*

1. MASS
2. subselect
3. klaR
4. combinat
5. ggplot2

## LVI Linear Discriminant Analyses: Beta Version

Table A1.1. The input (User) and output (Python and R) data (.csv) files.

ID	File	Source	Description
1	ASSESS	Python	File with summary statistics for assessment of results from modeling
2	BWRATIO	R	Ratios: Square root of between to within variance for each discriminant function
3	CTABALL	R	Cross tabulation of results using all data in model calibration.
4	CTABSUM	Python	Cross tabulation statistic summary derived from CTABULATION
5	CTABULATION	R	Cross tabulation results using Take-OneLeave-One
6	DFUNCT	R	Discriminant functions
7	FCENTROID	Python	Centroids associated with each classification and set of selected variables
8	FCLASS	Python	Class assignments to each observation in LVINEW
9	FUZZYC_INITIALIZATION	User	User defined inputs to developing classification
10	LVINEW	User	User defined base data including X, Y and CLASSIFICATION variables
10	MINMAXCORR	R	Minimum and Maximum Correlations within each X variable set
11	NNTARGETxx	Python	This contains the list of k-nearest neighbours and associated distances to target observation using a set of chosen set of X-variables, where xx = 1,2, ... p variables. There may be more than one of these files created in the Rwd.
12	POSTERIOR	R	Error of estimation using posterior probabilities (see Hora and Wilcox 1982)
13	PRIOR	R	Prior probability selected by user
14	PyDataDict	Python	This is a special file in the \\Rwd\\Python\\DATDICT\\ used to control how data is brought into the Python environment and transformed for further analyses (i.e. declared as a string, integer, or float value).
15	TNNSTATxx_MEAN	Python	This contains the target nearest neighbor Y-variable (in separate columns) MEAN statistics for each TARGOBS, where xx represents a model number.
16	TNNSTATxx_SDEV	Python	This contains the target nearest neighbor Y-variable (in separate columns) standard deviation (SDEV) statistics for each TARGOBS, where xx represents a model number.
17	UCORCOEF	R	Correlation Coefficients amongst variables in UNIQUEVAR
18	UNIQUEVAR	Python	Unique variable set associated with variables listed in VARSELECT
19	VARMEANS	R	Variable means by original CLASS assignments
20	VARSELECT	R	String table describing variable sets used to derive XVARSELV

## LVI Linear Discriminant Analyses: Beta Version

ID	File	Source	Description
21	XARMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on absolute distances using the normalized X-Variables.
22	XCOV	Python	This file contains the covariances for each pair of variables across all the X-variables selected in the variable selection process.
23	XERMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Euclidean distances using the normalized X-Variables.
24	XMRMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Mahalanobis distances using the normalized X-Variables.
25	XNNA	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on absolute distances and using the normalized X-Variables.
26	XNNE	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on Euclidean distances and using the normalized X-Variables.
27	XNNM	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on Mahalanobis distances and using the normalized X-Variables.
28	XNSCORE	Python	These are the normalized X-Variable values assigned to each reference observation (LVI_FCoid) and variable set combination (VARSET3).
29	XSCORE	Python	These are the non-normalized (original) X-Variable values assigned to each reference observation (LVI_FCoid) and variable set combination (VARSET3).
30	XSTAT	Python	This is the list of means (MEAN) and standard deviations (SDEV) for the X-Variables (VARNAME3) associated with each variable set (VARSET3) and used to normalize the XSCORES to produce XNSCORES.
31	XVARSELV	Python	This is the list of unique combinations of X-variables sets used as further input into discriminant analysis. It indicates the variable set (VARSET), the MODELID (relates to MODELID in VARSELECT), the number of models (NMODELS) in VARSELECT that used the same list variables, the number of variables used in each model (NVAR), and a list of the associated variable names (under column headings identified as NVAR1, NVAR2 ... "N" indicates that there were no variables beyond a certain number).
32	YARMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on absolute distances using the normalized Y-Variables.
33	YCOV	Python	This file contains the covariances for each pair of variables across all the Y-variables selected in the variable selection process.
34	YERMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Euclidean distances using the normalized Y-Variables.



## LVI Linear Discriminant Analyses: Beta Version

ID	File	Source	Description
35	YMRMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Mahalanobis distances using the normalized Y-Variables.
36	YNNA	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on absolute distances and using the normalized Y-Variables.
37	YNNE	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on Euclidean distances and using the normalized Y-Variables.
38	YNNM	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on Mahalanobis distances and using the normalized Y-Variables.
39	ZARMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on absolute distances using the normalized Z-Scores.
40	ZCOV	Python	Contains the covariance matrix amongst the discriminant scores for each function associated with normalized Z-scores for each variable set; the inverse of the covariance matrix is used in calculating Mahalanobis distances. Note that this tends to be a diagonal matrix (i.e. 1's on the diagonal and 0's in the off diagonal).
41	ZERMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Euclidean distances using the normalized Z-Scores.
42	ZMRMSE	Python	This produces root mean squared errors (RMSE, BRMSE) and levels of bias (BIAS) for each VARSET-KNN-VARNAME combination based on Mahalanobis distances using the normalized Z-Scores.
43	ZNNA	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on absolute distances and using the normalized Z-Scores.
44	ZNNE	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and associated distances based on Euclidean distances and using the normalized Z-Scores.
45	ZNNM	Python	This contains the list of reference observation (REFOBS) nearest nearest neighbours (NNOBS) for each variable set (VARSET3) and Mahalanobis distances based on absolute distances and using the normalized Z-Scores.
46	ZNSCORE	Python	Normalized discriminant scores (subtracted from the mean and divided by the standard deviation for each discriminant function).
47	ZSCORE	Python	Un-normalized discriminant scores for each observation associated with a variable set – discriminant function combination.
48	ZSTAT	Python	Mean and standard deviation in Z-values for each variable set – discriminant function combination.

### Appendix II: The Standard LVI kNN Analysis Process

#### *Demonstration Package*

It is recommended that if you are a first time user you start with the demonstration package as follows:

1. Copy demonstration input files into the top of the Rwd directory (note these will already be loaded into that directory when you first put the Rwd file into your (C:) directory.
  - 1.1. The following files are located in the `\\Rwd\\Rdata\\Archived\\LVI\\Demo \\ InputFiles\\` directory:
    - FUZZYC\_INITIALIZATION.csv: This file contains information used to run the Fuzzy C-means algorithm.
    - LVINEW.csv: This file contains the “reference” data used to select a subset of X-Variables (that are also contained in the “target” dataset for the purpose of estimating Y-Variables that are of interest (that are not contained in the “target” dataset.
    - QTARGET.csv: This file contains the “target” dataset. Note that the X-Variable data should be the same in both the reference and target datasets. Best practice suggests that both the reference and the target datasets be created at the same time, with the same information (except the Y-variables) to ensure that this is the case. In this particular demo this rule is violated.
    - XVARSELV1.csv: This file contains a listing of variables to be selected as “candidate” X (ideally in both the reference and target datasets) and preferred or desired Y-variables (from the reference data) that are to be estimated for each observation in the target dataset. Warning: In this demonstration package the X-variables contained in the reference data are not the same as those in the target data. Care was taken in setting up this file to ensure that the variables selected as potential for candidates were limited to those that were in both datasets.

These files are to be copied and then pasted into the `\\Rwd\\` directory.

You will learn more about these files as you work through the process below. The input files developed for the demonstration package are referred to in the first 5-steps of the process outline below. Finally the names of these files correspond with those used originally to develop these procedures and associated Python modules and R scripts; however, the contents have been reduced in size to reduce processing time.

- 1.2. The following file is located in the `\\Rwd\\Python\\DATDICT\\DemoDict\\` directory:
  - PyRDataDict.csv: This file is central to importing new data into many of the Python modules. This table has the following column names:

## LVI Linear Discriminant Analyses: Beta Version

- TVID: This is a unique ID (integer) identifying each row consisting of an input TABLENAME and variable names (VARNAME) within that table.
- TABLENAME: This refers to an actual file name , where the file name consists of the indicated TABLENAME plus a “.csv” extension and is to be found in the top of [\\Rwd\\](#) directory. “LVINEW” is the name of the “.csv” file that is recognized in the Python scripts as containing the “reference” data. In the PyRDataDict file, “LVINEW” is noted as a TABLENAME without a “.csv” extension in rows 37 (TVID = 36) to 215 (TVID = 214). Pending further changes to the program users must refer to the reference dataset TABLENAME as LVINEW and must have a “.csv file of that name (i.e. LVINEW.csv) in the [\\Rwd\\](#) directory. The following table names and associated attributes should always be maintained within the dictionary and should not be changed (this applies to all of the associated data in adjacent columns, except the TVID’s that can be changed but should be ordered from 1 to n in steps of 1):
  - VARSELECT (new file generated in process)
  - XVARSELV1 (user defined)
  - CTABULATION (new file)
  - MINMAXCOR (new file)
  - POSTERIOR (new file)
  - CTABSUM (new file)
  - DFUNCT (new file)
  - BWRATIO (new file)
  - LVINORM (new file? identical to LVINEW but contains normalized variables; I believe that this can be eliminated)
  - QTARGET (user defined – Unique ID’s must have NEWVARNAME equal to TFCOID. All of the remaining variable names are user defined. The target data should always be contained within a “.csv” file named QTARGET).
- VARNAME: These are the variable names referred to within each TABLENAME located in the top row of the “.csv” file and sometimes referred to as the “header”. The first column should contain the unique ID’s associated with each observation. That column should always be named “LVI\_FCOID” . All of the remaining variable names and associated numbers of columns may be changed in reference to LVINEW (but be sure to update the TVID’s when you do this). “LVI\_FCOID” should also be entered into associated NEWVARNAME column as a standard practice. With the exception of LVI\_FCOID as the indicated unique ID VARNAME associated with LVINEW (and LVINORM) all of the remaining variables names associated with this TABLENAME, and all of the VARNAMEs associated with QTARGET can be user defined.

## LVI Linear Discriminant Analyses: Beta Version

- NEWVARNAME: With the exception of LVI\_FCOID as the unique ID associated with LVINEW (and LVINORM) TABLENAME and TFCOID associated with QTARGET TABLENAME, the remaining NEWVARNAME's are user defined.

This file has been unchanged from the original that was used to process the Quesnel Data. Now that you are familiar with the contents of demonstration files, and have copied them into the appropriate directories (or are using them for the first time after installing the Rwd directory), you are ready to work through the process as described below. Note that examples of all of the output files are in the //Rwd//Rdata//Archived//LVI//Demo//OutputFiles// directory. One way to go through the process is to read the descriptions involving each step, look at the output files and try to understand what has happened without running any of the scripts to begin with.

### *From classification to nearest neighbours: Stepwise procedures for the analysis process*

The following is a complete description of the process used to develop a map of forest types using nearest neighbor techniques:

1. Prepare LVINEW dataset and update PyRDtataDict.csv with new attributes; ensure that key variable name is labeled 'LVI\_FCOID'. Prepare the QTARGET dataset with the key NEWVARNAME identified as TFCOID.
2. Make sure PyRDataDict.csv is in the \\Rwd\\Python\\DATDICT\\ directory and that the variable names associated with LVINEW and QTARGET are correctly identified.
3. Put the list of X and Y variables (minus the Key Variable) in LVINEW into VARSELV1.csv. Select the candidate X variables by entering an X next to those selected variables. Select the desired Y variables (variables of interest that are to be estimated in some way using the X-variable set) by entering a Y next to those variables. Enter N beside any remaining variables that are not to be included in the analysis. LVI\_FCOID may or may not be included; if it is included it must have an N entered next to it indicating that it is not a Y or X variable of interest.
4. Check to make sure the following files are in the Rwd Directory:
  - FUZZY\_C\_INITIALIZATION.csv
  - LVINEW.csv
  - OpenSession.RData
  - XVARSELV1.csv with indication of selected Y-variables and selected X-Variable candidates.
  - QTARGET.csv

## LVI Linear Discriminant Analyses: Beta Version

### 5. Start with process number 3: Fuzzy C-means Classification

5.1. Open FUZZY\_C\_INITIALIZATION .csv and set initial parameters, particularly the range of classifications to be produced in terms of number of classes (The Demo has the following settings:

FID	VARNAME	VARVALUE	DESCRIPTION
1	M_VALUE	2	This controls the degree of fuzziness where 1 is classification with hard boundaries – generally recommended to be set equal to 2
2	MIN_ERROR	0.01	This controls the minimum amount of gain (reduction in within group total sums of squares in the distance metric) before the algorithm stops – smaller numbers result in more iterations before a solution is reached. A suggested value is 0.01
3	ROUTINE	FCM_E	This refers to the distance metric. FCM_E is the Euclidean distance and produces the most stable outcomes. FCM_M is Mahalanobis distance across all classes.
4	LLNCLASS	2	This determines the numbers of classifications to be produced with the number of classes being equal to LLNCLASS, LLNCLASS +1, LLNCLASS+2 ... UNCLASS. For each classification in return. The different classifications are defined in the output as CLASSx where x is equal to the number of classes included in the classification.
5	ULNCLASS	25	See description for LLNCLASS.

5.2. Run LVI\_CLASSIFY.py (Procedure number 3, i.e. section 3 in the main document entitled, “Fuzzy C-Means Classification”; see main text for the step-by-step description of how to implement the procedure)

- The following files will be added to the Rwd directory:
  - FCLASS.csv
  - FCENTROID.csv

5.3. Add the classifications and associated labels identified in FCLASS.csv to the LVINEW.csv dataset, including the classification labels. Make sure that the assignments are based on matching LVI\_FCOID (This has already been done in the demonstration package; you can check that this is so by looking at the classifications FCLASS.csv and checking that these are also in LVINEW) .

5.4. Update the classification filed names associated with LVINEW in the PySDataDict.csv file (You can also check this when running through the demonstration).

## LVI Linear Discriminant Analyses: Beta Version

### 6. Run Procedure Number 4: Start with Variable Selection from a Large Number of Variables

#### 6.1. Open R-session

##### 6.1.1. OpenSession.RData

Click on this in the Rwd directory

#### 6.2. Implement procedure number 4 (Start with Variable Selection from a Large Number of Variables) above

##### 6.2.1. Use XVARSELV1.csv in the process

- At the end of the R part of the session the following file will be added to the Rwd directory
  - OpenSession.RData
- At the end of the Python part of the session the following files will be added to the Rwd directory:
  - XVARSELV.csv
  - UNIQUEVAR.csv

#### 6.3. Close R session – do not save the results

### 7. Run Procedure Number 7: Produce Unique X-Variable Subset Correlation Matrix

#### 7.1. Open R Session

##### 7.1.1. OpenSession.RData

- At the end of the R session the following files will have been added to the Rwd directory:
  - MINMAXCOR
  - UCORCOEF

Check UCORCOEF for variables with correlations  $\leq -0.8$  or  $\geq 0.8$ . Select the preferred variable associated with these pairs of variables to be used in the analysis. Use process number 8 (Produce Original Classification Unique X- or Y-Variable Subset Box and Scatter Plots) to help with this by comparing variable pairs and selecting the one that seems produce the greatest differentiation amongst the classes, while excluding the others. Start by selecting the best x- variable and then eliminate any variables highly correlated with that variable. Then select the next best available variable and repeat this process until all of the high correlations are removed. For those variables that are to be removed go back into XVARSELV1 and change the corresponding variable names with XVARSEL = X to XVARSEL = N instead. Then repeat stem 7 in this overall procedure.

## LVI Linear Discriminant Analyses: Beta Version

### 8. Run Procedure Number 6: Run Linear Discriminant Analysis for Multiple X-Variable Sets

#### 8.1. Open R Session

##### 8.1.1. OpenSession.RData

- At the end of the R part of the session the following files will have been added to the Rwd directory:
  - BWRATIO.csv
  - CTABALL.csv
  - CTABULATION.csv
  - DFUNCT.csv
  - POSTERIOR.csv
  - PRIOR.csv
  - VARMEANS.csv
- At the end of running the Python part of the session the following files have been added to the Rwd directory:
  - CTABSUM.csv

### 9. Run Procedure Number 9: Combine Evaluation Datasets

#### 9.1. Python Session

- At the end of running the Python part of the session the following files have been added to the Rwd directory:
  - ASSESS.csv

### 10. Run process number 11: Calculate Z-scores, Nearest Neighbours, ...

#### 10.1. Python Session

- At the end of running the Python part of the session the following files have been added to the Rwd directory:
  - ZSCORE.csv
  - ZNSCORE.csv
  - ZSTAT.csv
  - ZNNA.csv
  - ZNNE.csv
  - ZNNM.csv
  - ZARMSE.csv

### 11. Run Procedure Number 12: Calculate Nearest Neighbours, ... , Using Selected X-Variables.

## LVI Linear Discriminant Analyses: Beta Version

12. Run Procedure Number 13: Calculate Nearest Neighbours ... , Using Selected Y- Variables.
13. Run Procedure Number 14: Identify k-Nearest Neighboursd in Target Dataset.
14. Run Procedure Number 15: Compile the kNN Target Dataset Y-Variable Statistics.
15. Run Procedure Number 16: Convert TNNSTATS file VARNAMES into Column Format.