PRRFECT User Manual

Chapter 1

What does PRRFECT do?

PRRFECT can be used to provide rapid pre-processing and classification results for many types of spectroscopy. There are several pre-processing options split into 4 main categories. These are binning, smoothing, normalisation, and baseline correction.

1.1 Binning

Binning is used when changing the overall resolution of the spectra is desired. It works by taking an average of the intensity value of every n points. It also effectively increases the wavenumber spacing of the spectra. This practice can sometimes increase the signal-to-noise ratio of spectra. In the implementation here, the parameter **bin_factor** is set in the user defined variables file. This factor dictates how many consecutive datapoints are averaged. For example, a **bin_factor** of 4 would provide a processed spectrum with 4 times fewer wavenumber columns and datapoints.

1.2 Smoothing

The choices for smoothing (set via the **smooth_choice** variable value) are fourfold; none (0), Savitzky-Golay filtering (1), wavelet denoising (2), and local polynomial fitting with Gaussian weighting (3).
These options also have parameters which can be set via the **smooth_par** value. This **smooth_par** becomes the parameter for the smoothing method set by **smooth_choice**. In the case of **smooth_choice** option 1 (S-G filter), the **smooth_par** value sets the filter order. For option 2 (wavelet denoising), **smooth_par** sets the length of the wavelet filter. If **smooth_choice** option 3 (local polynomial with Gaussian weighting) is selected, **smooth_par** sets the bandwidth of the Gaussians. See Table 5.1 for a clearer explanation of how this

works.

1.3 Normalisation

Normalisation choice is set by the **norm_choice** value. There are four options for normalisation in PRRFECT. These are none (0), min/max scaling (1), vector normalisation (2), and normalisation to Amide I band (3).

1.4 Baseline Correction

There are five options for baseline/background correction offered. These are set by the **bg_choice** value. The options are none (0), first derivative (1), second derivative (2), rubberband (3), and polynomial baseline correction (4). Options 3 and 4 have additional parameters which must be set. These are set by altering the value of the integer variable **bg_par**. If **bg_choice** is set to 3, the **bg_par** value controls the noise cut-off level used by the rubberband algorithm. An additional parameter **RBp** is provided to further control the noise-cutoff level for the rubberband method only. This **RBp** parameter acts as a multiplier for the **bg_par** value for rubberband. This is to allow for easier grid searching. When **bg_choice** is set to option 4 (polynomial baseline correction), **bg_par** controls the degree of the polynomial used.

Chapter 2

Prerequisites and Data Format

The user should have a recent version of R, with the packages: "randomForest", "xtable", "caret", "prospectr", "signal", "wrt", "KernSmooth", "Peaks", and "hyperSpec" installed.

PRRFECT also expects the input spectroscopic data to be in the CSV (comma separated values) format. This is available in all modern spreadsheet programs. All of the data should be in one file, with **spectra in rows**. The labels for the data, eg **class & ID information, should be in columns at the front of the file**.
The column headings for the class and ID information can be text or numeric. The column headings for the actual intensity data should be numbers, (eg wavenumbers, wavelength etc.) but with "X_" appended. See table 2.1 below for an example acceptable layout.

Table 2.1: Example dataset format

Cancer/non	Met/brain/non	Origin	HGG/LGG	subtype	Patient_ID	X_898.65	X_900.58	X_902.51	X_904.44	X_...
1	1	6	0	0	13673	0.0116	0.0116	0.0115	0.0115	...
1	1	6	0	0	13673	0.0115	0.0115	0.0115	0.0114	...
1	1	6	0	0	13673	0.0116	0.0116	0.0115	0.0115	...
1	1	6	0	0	13673	0.0114	0.0114	0.0113	0.0113	...
1	1	6	0	0	13673	0.0116	0.0115	0.0115	0.0114	...
1	1	6	0	0	13673	0.0117	0.0116	0.0116	0.0115	...
1	1	6	0	0	13673	0.0116	0.0116	0.0116	0.0115	...
1	1	6	0	0	13673	0.0117	0.0117	0.0116	0.0116	...
1	1	6	0	0	13673	0.0117	0.0117	0.0116	0.0116	...
1	1	3	0	0	13672	0.0116	0.0115	0.0115	0.0114	...
1	1	3	0	0	13672	0.0115	0.0115	0.0114	0.0113	...
1	1	3	0	0	13672	0.0116	0.0115	0.0115	0.0114	...
1	1	3	0	0	13672	0.0115	0.0115	0.0114	0.0113	...
1	1	3	0	0	13672	0.0115	0.0114	0.0113	0.0113	...
1	1	3	0	0	13672	0.0115	0.0114	0.0114	0.0113	...
1	1	3	0	0	13672	0.0117	0.0116	0.0115	0.0115	...
1	1	3	0	0	13672	0.0117	0.0116	0.0116	0.0115	...
1	1	3	0	0	13672	0.0119	0.0119	0.0118	0.0117	...
1	1	6	0	0	13671	0.0119	0.0118	0.0118	0.0117	...
1	1	6	0	0	13671	0.0121	0.0120	0.0119	0.0119	...
1	1	6	0	0	13671	0.0121	0.0120	0.0120	0.0119	...
...	...	...	...	...	...	...	...	...	...	...

In this case, there are several class columns and a patient ID column. The wavenumber columns are denoted by the X_ prefix. The content of the label and ID columns are numeric identifiers in this case, but they could be text also. The content of the X_ prefix columns are the recorded absorbencies at each wavenumber, forming a whole spectrum per row. If a column contains intensity/absorbance data, it **must be prefixed with X_ as shown in the table**. An easy way to do this is to use the CONCATENATE command in Excel. This command allows a user to add a prefix to every sell in a selected range.

Chapter 3

Setting up and splitting datasets

For the RF classifier, a splitting of the data into a training and test set is required. Cross validation is performed on the training set, and the learned patterns from this are used to predict the content of the test set. In the Github repository, there is a small R script provided called "select_training_and_test_sets.r". This script will split a spectroscopic dataset into two files, a training set and a test set. There are **four variables a user must set in order to used this script**. These are the name of the dataset, the column header denoting the ID column, the yvar column and the desired training set size.

Figure 3.1: Interface of the splitting script

```
#Read data - original full dataset
f<-read.table("coffee_BRS.csv", header=TRUE, sep=",")

#Variables
patients<-"sample_number"    #Name of column containin

#yvar sets which column to split data by - aiming to
yvar="coffee_type"
trprop <- 0.66
```

The first line should be changed to reflect the filename of the dataset under study. The second line should be changed to signify the column heading containing patient ID or sample number. This is to allow for datasets containing more than one spectrum per sample. It ensures that spectra are kept together, and that multi-spectrum samples are not split between the training and test

sets. yvar should be set to a class column. This is the column which the script "takes aim at" making sure an even split of classes occurs. trprop is the proportion of the data which ends up being in the training set. The rest goes to the test set. At the setting shown, a training set would be two thirds of the data, and the test set would be the remaining third.

After running this script, the original dataset will still be present and two new files will be created containing the training and test sets. By default, these are named "input_training_dataset.ssv" and "input_testing_dataset.ssv". These files are in the format of semicolon-separated values.

Data can of course be split manually, but care must be taken to ensure the two resulting files have the same number of columns and column headers.

Chapter 4

The user-defined input file

This file is how the user interacts with the program, and where the pre-processing and output choices are made.

The first two variables, **infile_tr & infile_te** are for telling the program the names of the training and testing input datasets.
The variable **k** sets the number of folds of the k-fold cross validation.
The logicals **se_and_ci & se_and_ci2** set whether standard errors and confidence intervals for each statistic are reported the the results; this is for text and tabular result files respectively.
The variable **bin_factor** sets the degree of wavenumber resolution decrease. A bin_factor setting of 2 would signify that the effective spectral resolution is being decreased by a factor of 2.
smooth_choice allows for the selection of the smoothing method. The variable **smooth_par** allows adjustment of the settings for certain smoothing methods. See the section "Available Pre-processing Options" for specific settings of both smooth_choice and smooth_par.
Normalisation choice is set by selecting the appropriate number for **norm_choice**.
Baseline/background correction methods and settings are chosen by adjusting **bg_choice & bg_par** respectively.
RBp is a setting for the rubberband baseline correction method. It specifies the level of the estimated noise floor of the spectrum. It is used in conjunction with **bg_par** when using the rubberband method. It is simply a multiplier of the **bg_par** only when rubberband baseline correction is used. We recommend that it is set to the same order of magnitude as the overall lowest y-axis point of the spectra. This allows a simplification of a grid search, allowing bg_par only to be adjusted.
Next comes the section of the input file which dictates the types of graphs desired. These are all set by logical variables. Graphs of spectra only, average spectra (per class), Gini importance & spectra, and average spectra & Gini importance can all be requested. The plot of Gini importance only is always plotted.

The **yvariables** input should contain the column headers with which to do the classification. This can be either one or multiple columns.

The options **min_wavenumber & max_wavenumber** choose the region of the spectrum to be used in the RF classification. This also affects the region should on the graphs.

The variable **patients** is where the name of the column containing patient IDs/sample numbers should be put.

The input file is read automatically upon running the main program.

Chapter 5

Available Pre-processing Options

Table 5.1: Key to results

Setting	Explanation	Parameter	Explanation
bin_factor	Binning factor		
1	No binning		
2	delta*2		
4	delta*4		
8	delta*8		
16	delta*16		
32	delta*32		
n	delta*n		
smooth_choice	Smoothing choice	smooth_par	Smoothing parameter
0	none	-	-
1	S-G filter	**1,2,3,4,5,6, ...**	Filter order
2	Wavelet denoising	2*(**2,3,4,5,6, ...**)	Filter length
3	Local polynomial fit with Gaussian	**1,2,3,4,5,6, ...**	Width of Gaussian
norm_choice	Normalisation choice		
0	none		
1	min/max to 0-1 scaling		
2	Vector normalisation		
3	Normalise to Amide I band		
bg_choice	Baseline/background Correction choice	bg_par	Baseline parameter
0	none	-	-
1	First derivative	-	-
2	Second derivative	-	-
3	Rubberband baseline correction	RBp*(**1,2,3,4,5,6, ...**)	Noise cut-off level
4	Polynomial baseline correction	**1,2,3,4,5,6, ...**	Polynomial degree

Figure 5.1: User Defined Variables

```
35 infile_tr<-"input_training_dataset.ssv"
36 infile_te<-"input_testing_dataset.ssv"
37 ########################################
38 ########################################
39 #       CLASSIFIER AND OUTPUT USER-DEFI
40 #       ################################
41 #
42
43 k<-5                                    
44 se_and_ci<-FALSE                        
45 se_and_ci2<-FALSE                       
46
47
48 bin_factor <- 2        #What factor shou
49 smooth_choice <- 0          #Smoothing s
50 smooth_par <- 0             #Parameter f
51 norm_choice <- 0            #Normalisati
52 bg_choice <- 1              #Baseline co
53 bg_par <- 0                 #ONLY for op
54 RBp <- 0.1                  #Rubberband 
55
56 spectra <- TRUE             # Plot all s
57 average_spectra <- FALSE     # Plot aver
58 imp_and_all_spectra <-FALSE # Plot impo
59 avg_and_gini <- FALSE
60
61
62 #Select columns to classify. This will 
63 yvariables<-c("coffee_type")
64
65 #Options
66 min_wavenumber<-800                   #M
67 max_wavenumber<-1900                    
68 patients<-"sample_number"          #Name
```

Chapter 6

Example Workflow - General Use

6.1 Input

An example workflow for a single run of the program is shown in this section. The user wishes to study a dataset called "olive_oil_BRS.csv" containing FTIR ATR spectra of various olive oil samples from different countries. The dataset contains 2 columns at the start of the file called "sample_number" and "country". The remainder of the file contains wavenumber columns with absorbency values, with the X_ prefix, as described above.
The user would first used the script "select_training_and_test_sets.r" and set desired parameters in that. In this case, **trprop** will be set to 0.75 for 75% of data as the training set. A simple R command to run the script is shown:

source "select_training_and_test_sets.r"

Screen captures of the script and the resulting log are shown below.

Figure 6.1: Example Workflow - general : Split script setup

```
#Read data - original full dataset
f<-read.table("olive_oil_BRS.csv",

#Variables
patients<-"sample_number"    #Name o

#yvar sets which column to split da
yvar="country"
trprop <- 0.75
```

Figure 6.2: Example Workflow - general : Split log

```
Class:  country
  Item:  Greece
    Training:  16
    Testing:  4
    Fraction (tr):  0.8
  Item:  Italy
    Training:  26
    Testing:  8
    Fraction (tr):  0.765
  Item:  Portugal
    Training:  12
    Testing:  4
    Fraction (tr):  0.75
  Item:  Spain
    Training:  38
    Testing:  12
    Fraction (tr):  0.76
--
```

After this step, 3 new files have been generated: **select_training_and_test_sets.log, input_training_dataset.ssv & input_testing_dataset.ssv**.

Now comes the inputting of user-defined variables in the **user_defined_input.R** file. In this case, the user wished to bin the data by a factor of 2, perform S-G filtering of order 2, vector normalisation and first derivative processing. A screen capture of these options is shown below. Other settings were a wavenumber range of 800 - 1800 cm^{-1}, setting all graphical output to TRUE, and specifying the relevant columns.

Figure 6.3: Example Workflow - general : User defined input

```
bin_factor <- 2        #What
smooth_choice <- 1
smooth_par <- 2
norm_choice <- 2
bg_choice <- 1
bg_par <- 0
RBp <- 0                    #0

spectra <- TRUE
average_spectra <- TRUE
imp_and_all_spectra <-TRUE
avg_and_gini <- TRUE


#Select columns to classify
yvariables<-c("country")

#Options
min_wavenumber<-800
max_wavenumber<-1800
patients<-"sample_number"
```

Then all that remains is to run the program. A command in R or Rstudio to do this is:

source "PRRFECTv1.r"

6.2 Output

Several files are generated by running the program. These are listed and described below.

country_random_forest_importance_MeanDecreaseGini.txt This contains two columns, wavenumber and Gini importance. It shows how important each wavenumber column was to the classification. It is formatted in order of importance, with the most important wavenumbers at the end of the file.

country_results_ALL.html This file contains a table of statistical results for the classification. It can be opened in a web browser for quick viewing.

country_results_ALL.txt Contains the same data as the html file, but in a standard text format suitable for viewing in spreadsheet programs etc.

seed.txt This contains the current random seed in R at the time of running the program. It is only useful if you wish to perform EXACTLY the same random forest classifier again in the future.

country_rf.RData This is captured image of the internal R workspace. It can be opened and browsed should one wish to see details of the internal process of the classification for example.

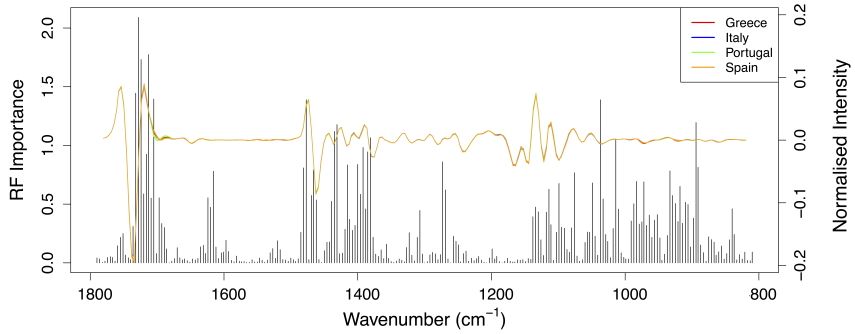
PDF files, *5 These contain the requested plots for viewing the spectra and/or the Gini importance.

Some files are shown as images below.

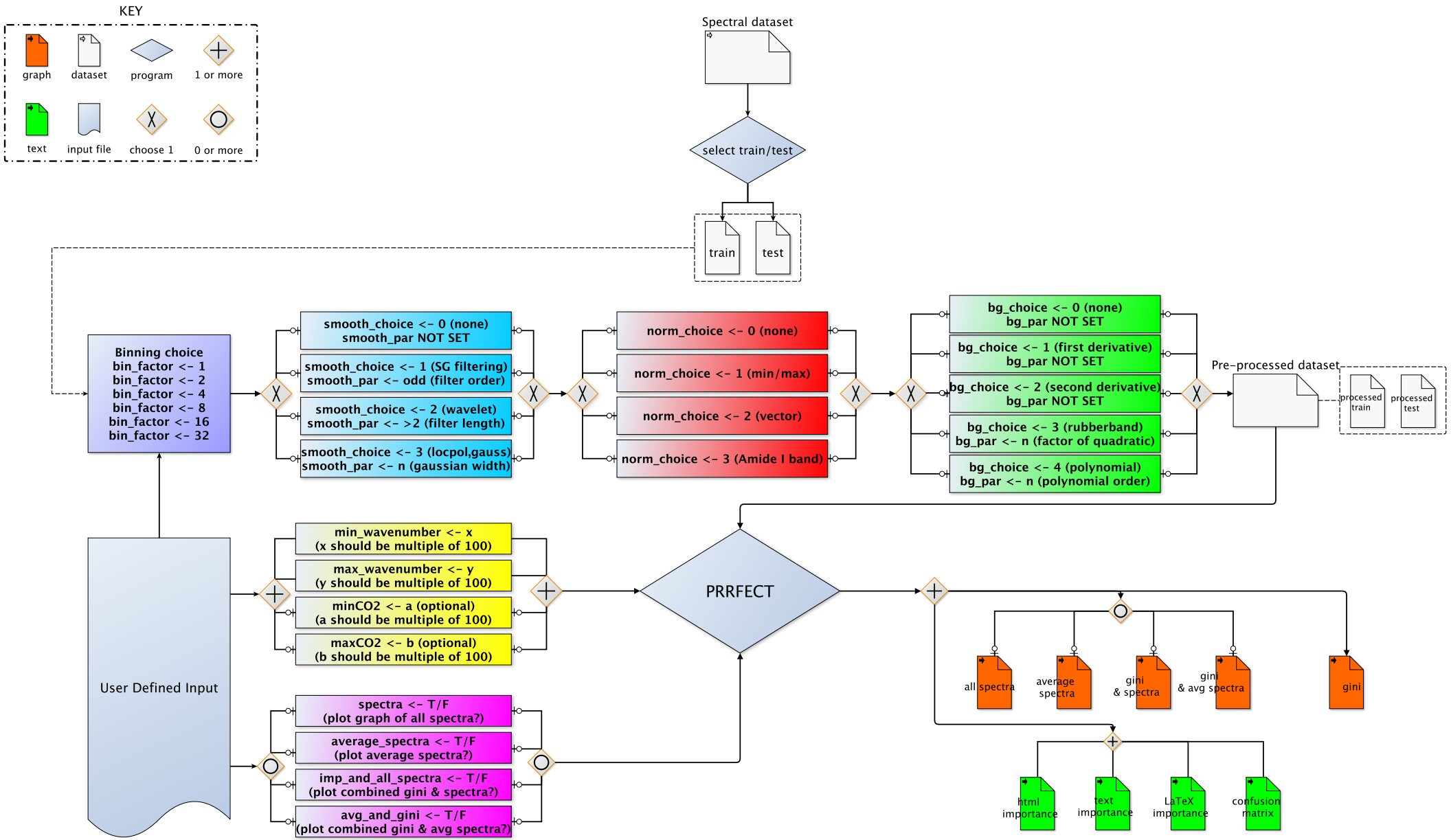
Figure 6.4: Example Workflow - general : html table

Class	pacCV	mccCV	sensCV	specCV	pprCV	nprCV	pacTS	mccTS	sensTS	specTS	pprTS	nprTS
Greece	1	1	1	1	1	1	0.964	0.876	0.8	1	1	0.958
Italy	0.978	0.949	0.929	1	1	0.97	0.893	0.786	0.727	1	1	0.85
Portugal	0.967	0.862	0.846	0.987	0.917	0.975	0.929	0.679	1	0.923	0.5	1
Spain	0.946	0.889	0.971	0.93	0.895	0.981	0.929	0.861	1	0.889	0.833	1

Figure 6.5: Example Workflow - general : Average spectra and Gini



The next page features a full map of the output and input to PRRFECT.


KEY
graph
dataset
program
1 or more
text
input file
choose 1
0 or more
Spectral dataset
select train/test
train
test
Binning choice
bin_factor <- 1
bin_factor <- 2
bin_factor <- 4
bin_factor <- 8
bin_factor <- 16
bin_factor <- 32
smooth_choice <- 0 (none)
smooth_par NOT SET
smooth_choice <- 1 (SG filtering)
smooth_par <- odd (filter order)
smooth_choice <- 2 (wavelet)
smooth_par <- >2 (filter length)
smooth_choice <- 3 (locpol,gauss)
smooth_par <- n (gaussian width)
norm_choice <- 0 (none)
norm_choice <- 1 (min/max)
norm_choice <- 2 (vector)
norm_choice <- 3 (Amide I band)
bg_choice <- 0 (none)
bg_par NOT SET
bg_choice <- 1 (first derivative)
bg_par NOT SET
bg_choice <- 2 (second derivative)
bg_par NOT SET
bg_choice <- 3 (rubberband)
bg_par <- n (factor of quadratic)
bg_choice <- 4 (polynomial)
bg_par <- n (polynomial order)
Pre-processed dataset
processed train
processed test
User Defined Input
min_wavenumber <- x
(x should be multiple of 100)
max_wavenumber <- y
(y should be multiple of 100)
minCO2 <- a (optional)
(a should be multiple of 100)
maxCO2 <- b (optional)
(b should be multiple of 100)
spectra <- T/F
(plot graph of all spectra?)
average_spectra <- T/F
(plot average spectra?)
imp_and_all_spectra <- T/F
(plot combined gini & spectra?)
avg_and_gini <- T/F
(plot combined gini & avg spectra?)
PRRFECT
all spectra
average spectra
gini & spectra
gini & avg spectra
gini
html importance
text importance
LaTeX importance
confusion matrix

Chapter 7

Workflow - Advanced

7.1 Grid Search

Some advanced users may wish to automatically run many iterations of different pre-processing methods to find the best classification for their dataset. A common way to accomplish this is to use a grid search. PRRFECT does not have built-in grid search functionality, as the R language is not known for its ability to complete complex parallel operations. What is needed is a simple external script to iterate the parameters in the user defined input file. This input file has the advantage of being written in plain text, so it is easy to change the parameters via an external script. An example is provided in the repository, written in bash.

The basic structure of the code is nested for loops to iterate the parameters, and then commands to run the script in R from a bash environment. The call R –slave is recommended, as this produces minimal terminal output.

7.2 Multiple Training and Test Set Iterations

This is another aspect of running a very thorough grid search. Training and test sets can be selected multiple times, in order to mix the samples across both sets. This involves running the same user defined input parameters on several different train and test set selections. The overall size of the training and test sets is the same, but each time they will contain different spectra in each. This process is regarded as optional, and is useful if there are slight outliers in the dataset.

Script "select_96_sets.sh" provides an example of running the same parameters with 96 iterations of training and test selection.

Script "gridsearch1.sh" provides an example of running 96 iterations of training

and test selection **for each parameter iteration.** It is highly recommended to run such a grid search on a cluster.
Here, a generic version of the user_defined_input.r file would be used. Instead of entering numbers manually, the script has to be able to find the point in the file where the numbers for selecting pre-processing are. To allow for this, "markers" such as BRSbin and BRSschoice are put into the input file, where numbers selecting different processing and parameters would go. These are then replaced by numbers by the gridsearch1.sh script.
The file "grid_list1.txt" is used as input for the gridsearch1.sh script. It is a list of every parameter increment. Should one wish to explore further outside this grid size, a new list must be generated. This is best done by using nested for loops to generate the file.

7.3 Administration

If a parameter grid search and/or multiple set selection is being run, one needs a way to collect the statistical output of this. We recommend that "results_ALL.txt" file is used for this. It provides an easy way to "grab" data using simple tools such as grep, awk, or sed. Examples of such scripts are provided in the repository.

A basic admin script is provided (basicadmin.sh). It will collect the results of 96 iterations of training and test selection. It will condense 96 separate results from 96 folders, and produce a single file documenting them all. It will require some changes to match a particular dataset and filenames etc.

Note about these scripts - They will not work immediately in the form they are in for your dataset. They require changing of filenames etc. They are provided as a very basic example of the kind of scripts required to administer a grid search and collect the results. This is mainly why they are in the "Advanced" section, as this requires scripting/programming knowledge.

Chapter 8

Troubleshooting and Known Bugs

Incorrect wavenumber scale on graphs - this can sometimes happen if the **min_wavenumber & max_wavenumber** are not set to a multiple of 100. It is purely a graphical issue, the underlying classification will still work. Occasionally, graphs will be produced with a "-" symbol in front of the wavenumbers on the x-axis. This is again just a graphical issue. It can sometimes appear when the input file has the wavenumber columns in reverse numerical order.

Some pre-processing combinations don't work . This issue can sometimes be found with certain combinations. It is due to mathematics and the incompatibility of certain combinations with some dataset sizes. It is very difficult to predict which methods will produce an error with a particular dataset.