Manual for CWDPRNP Package

William L. Miller
Pennsylvania Cooperative Fish and Wildlife Research Unit
Intercollege Graduate Degree Program in Ecology
The Pennsylvania State University
University Park, PA

W. David Walter
U. S. Geological Survey
Pennsylvania Cooperative Fish and Wildlife Research Unit
University Park, PA
Correspondence: wlm159psu@gmail.com

May 1, 2017

Contents

1 Summary of CWDPRNP Package

Chronic wasting disease (CWD) is a fatal, neurological disease caused by an infectious prion protein, which affects economically and ecologically important members of the family Cervidae. Single nucleotide polymorphisms within the prion protein gene (PRNP) have been linked to differential susceptibility to the disease in many species. Disease management efforts are seeking to create spatial epidemiological models that predict transmission across the landscape. These efforts would benefit from a knowledge of the relative susceptibility of each population to the establishment and propagation of CWD. Wildlife managers and researchers are seeking to identify the distribution and frequency of potentially susceptible PRNP genotypes in order to aid in these efforts. The CWDPRNP package, implemented in program R, provides a unified framework for analyzing PRNP variability and spatial genetic structure. Specifically, this package implements three modules that streamlines the analysis of PRNP sequences. These include: (1) the sequence alignment module, (2) the genetic variation module, and (3) the spatial analysis module.

2 Installation Instructions

2.1 Tutorial Instructions

Open the .Rnw file in the tutorial folder in RStudio (downloaded here: https://www.rstudio.com/). Make sure that all .ab1 files and spatial data are in the same working directory. Make sure to change the active working directory path in the **Ref** code chunk below (see Ref function below). From there, all code chunks (highlighted in grey) can be run using the example dataset. Code can either be run line-by-line or by using code chunk commands, which can be found in the dropdown menu labeled "Chunks" in the top right portion of the text editor window.

2.2 CWDPRNP Installation

The CWDPRNP package can be found at https://github.com/walterASEL/CWDPRNP. Once the package is downloaded, place the file containing the functions into the working directory folder. The following code will load the package into R:

```
source("CWDPRNP_ver0.R")
```

2.3 Installation of Dependencies

The CWDPRNP package has two functions to load the required dependencies from the CRAN and Bioconductor repositories. The following code can be used to: (1) verify that the necessary dependencies are installed, (2) install any missing dependencies from either CRAN or Bioconductor, and (3) load all required dependencies.

```
CheckCRANPackages(packages = c("seqinr", "ape", "openxlsx", "sp",
    "splancs", "rgdal", "maptools", "gstat", "raster", "plotrix"))
CheckBioconductorPackages(packages = c("DECIPHER", "sangerseqR",
    "Biostrings", "msa"))
```

CWDPRNP also requires Rtools in order to write .xlsx files if it is being run on a Windows device. The Rtools package and manual can be found here: (https://cran.r-project.org/bin/windows/Rtools/). Note that installation of Rtools is not necessary if working from a different operating system.

2.4 Version Notes

The current iteration of CWDPRNP was edited and tested using the following version of R:

- **R**, version 3.4.0 (04-21-2017)

The current iteration of CWDPRNP was edited and tested using the following versions for all dependencies:

- **seqinr**, version 3.3.6
- **ape**, version 4.1
- **openxlsx**, version 4.0.17
- **sp**, version 1.2.4
- **splancs**, version 2.1.40
- **rgdal**, version 1.2.6
- **maptools**, version 0.9.2
- **gstat**, version 1.1.5
- **raster**, version 2.5.8
- **plotrix**, version 3.6.4
- **DECIPHER**, version 2.3.1
- **sangerseqR**, version 1.11.0
- **Biostrings**, version 2.43.8
- **msa**, version 1.7.2

The current iteration of CWDPRNP was edited and tested using the following version of Rtools:

- **Rtools**, Rtools34.exe

2.5 References

1. Bivand, R.S. et al. (2008) Applied Spatial Data Analysis with R. Springer New York, New York, NY.
2. Bivand, R.S. et al. (2017) rgdal: bindings for the geospatial data abstraction library. R package version 1.2-6.
3. Bivand, R.S. and Lewin-Koh, N. (2017) maptools: tools for reading and handling spatial objects. R package version 0.9-2.
4. Bodenhofer, U. et al. (2015) msa: an R package for multiple sequence alignment. Bioinformatics, 31, 3997-3999.
5. Charif, D. and Lobry, J.R. (2007) SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis. In, Structural approaches to sequence evolution. Springer, pp. 207-232.

6. Hijmans, R.J. (2016) raster: geographic data analysis and modeling. R package version 2.5-8.

7. Hill, J.T. et al. (2014) Poly peak parser: Method and software for identification of unknown indels using sanger sequencing of polymerase chain reaction products. Dev. Dyn., 243, 1632-1636.

8. Lemon, J. (2006) Plotrix: a package in the red light district of R. R-News, 6, 8-12.

9. Pages, H. et al. (2017) Biostrings: string objects representing biological sequences, and matching algorithms. R package version 2.43.8.

10. Paradis, E. et al. (2004) APE: analyses of phylogenetics and evolution in R language. Bioinformatics, 20, 289-290.

11. Pebesma, E. and Bivand, R.S. (2005) Classes and methods for spatial data in R. R News, 5.

12. Pebesma, E.J. (2004) Multivariable geostatistics in S: the gstat package. Computers Geosciences, 30, 683-691.

13. Rowlingson, B. and Diggle, P. (2017) splancs: spatial and space-time point pattern analysis. R package version 2.01-40.

14. Walker, A. (2017) openxlsx: read, write and edit XLSX files. R package version 4.0.17.

15. Wright, E.S. (2016) Using DECIPHER v2. 0 to analyze big biological sequence data in R. The R Journal, 8, 352-359.

3 Module 1: Sequence Alignment Tools

3.1 Introduction

The purpose of this module is to provide a set of tools that reads sequence files, performs primary and secondary base calls, and performs sequence alignment. This part of the package relies on a reference sequence, which is called from GenBank, in order to perform alignment to a known sequence. This package was originally built to analyze PRNP sequences from cervids; however, it has the capability to analyze sequence variation from any gene or sequence that can be specified from GenBank. This is done by allowing the user to specify both the reference sequence and the variable sites of interest within the sequences being analyzed. The CWDPRNP package, in its current form, is formulated to work with one species at a time. This negates the influence of indels within the protein coding region. All that is necessary is for a user to specify the species of interest and the CWDPRNP package will download a reference sequence for that species and align the sense and antisense sequences to that reference. We will revisit interspecies variation in future releases if there is substantial user interest for such features.

Note: This module currently works with chromatograms in .ab1 format (Applied Biosystems). This package was built around this platform as it is the most commonly used data format in our lab group and many of our collaborators. We will be adding additional functionality for other data formats (e.g. .fasta, .scf, etc.) in future updates.

3.2 Core Functions

3.2.1 RefSeq: Call Reference Sequence

Description:
The RefSeq function calls a specified reference sequence from GenBank. The default commands utilize 771 bp fragments that span from start codon to stop codon for several cervid species (*Odocoileus virginianus*, *Odocoileus hemionus*, and*Cervus canadensis*). The user can also choose to specify the GenBank accession number of another reference sequence if desired.
Function:
RefSeq(x, accession)
Arguments:

x : A character string specifying species of interest or accession number(see details below)

accession : A character string specifying a GenBank accession number

Details:
Cervus canadensis (x==c("CECA"))

Odocoileus virginianus (x==c("ODVI"))
Odocoileus hemionus (x==c("ODHE"))
Other accession number (x==c("other"), accession==c("..."))
ex. (x==c("other"), accession=c("AY639093")) for Rangifer tarandus
Values:
Returns an R object of class DNAString.
Applications:
This is a required function for the rest of the workflow. The output of this function is needed for sequence alignment and trimming.
Interpretation:
The object returned is a previously archived sequence. This sequence is not interpretable on its own; rather, is necessary for other analyses as stated above.
Dependencies:
This function relies on the following inherited functions:
ape::read.GenBank
ape::write.dna
Biostrings::readDNAStringSet
Biostrings::DNAString
Example:

```
# Elk Reference Sequence
setwd("D:/CWDPRNP/manualprep/compilation")
ref <- RefSeq(x = ("CECA"))
ref

##   771-letter "DNAString" instance
## seq: ATGGTGAAAAGCCACATAGGCAGCTG...TCCTCATTTTTCTCATAGTAGGATAG

# Example of user-defined reference sequence.
## Sequence is from Neovison vison (American mink) prion gene
## sequence to highlight extension to taxa outside Cervidae.
ref2 <- RefSeq(x = c("other"), accession = c("EF508270"))
ref2

##   774-letter "DNAString" instance
## seq: ATGGTGAAAAGCCACATAGGCAGCTG...TGCTCATTCTCCTGATAGTGGGATGA
```

3.2.2 CallBases: Call Bases from DNA Chromatogram

Description: This function uploads all DNA chromatograms in .ab1 format from the working directory and performs primary and secondary base calls. This function is meant to work with two files from each sample, a sense (or forward) sequence and an antisense (or reverse complementary) sequence. It is suggested to label each file in the following format:
Sense orientation: "SampleID_F.ab1"
Antisense orientation: "SampleID_R.ab1"

Function:
CallBases(flab, rlab, label)
Arguments:

flab : a character string identifying the common label of all forward sequences

rlab : a character string identifying the common label of all reverse sequences

label : a character string identifying the point at which sample ID can be subset from rest of filename ("*text.ab1")

Values:
Returns a list of length 2. The first element is a list containing all forward reads as sangerseq objects. The second element is a list containing all reverse reads as sangerseq objects. Note that the user must assign each list to its own object in order to proceed.
Applications:
This function is used for producing calls for raw sequence files and is a required function for the rest of the workflow. The output of this function is needed for sequence alignment and trimming.
Interpretation:
The object returned contains primary and secondary base calls for sequence files in both orientations (sense and antisense). These are raw sequences, which include the PRNP ORF and priming regions. Sequences will be processed (trimmed to extent of ORF) in other functions. Antisense sequences are still complementary to sense sequences and will be converted in other functions.
Primary and secondary calls are used to identify potential polymorphisms. An individual that is heterozygous for a particular site will have differing primary and secondary calls. Primary and secondary sequences are coded using standard notation for nucleotides.
Dependencies
This function relies on the following inherited functions:
sangerseqR::readsangerseq
sangerseqR::makeBaseCalls
Example:

```
myfiles <- CallBases(flab = c("*F_223F.ab1"), rlab = c("*R_224R.ab1"),
    label = c("\\R_224R.*"))
forward <- myfiles[[1]]
reverse <- myfiles[[2]]
forward[[1]]@primarySeq

##   803-letter "DNAString" instance
## seq: CTGATGCCTGAGCACATAGGCAGCTG...CCTTCCTGATTTCACTCAGTCATACT

forward[[1]]@secondarySeq
```

```
##   803-letter "DNAString" instance
## seq: TTGCTGCCTGAGCACATAGGCAGCTG...CCTTCCTGATTTAAGTCAGTCGTACT
```

3.2.3 Chrom: Write Chromatograms

Description:
This function will produce a .pdf file for all elements of the lists created in the CallBases function. This will produce a chromatogram for both the sense and antisense sequences.
Function:
Chrom(forward, reverse)
Arguments:

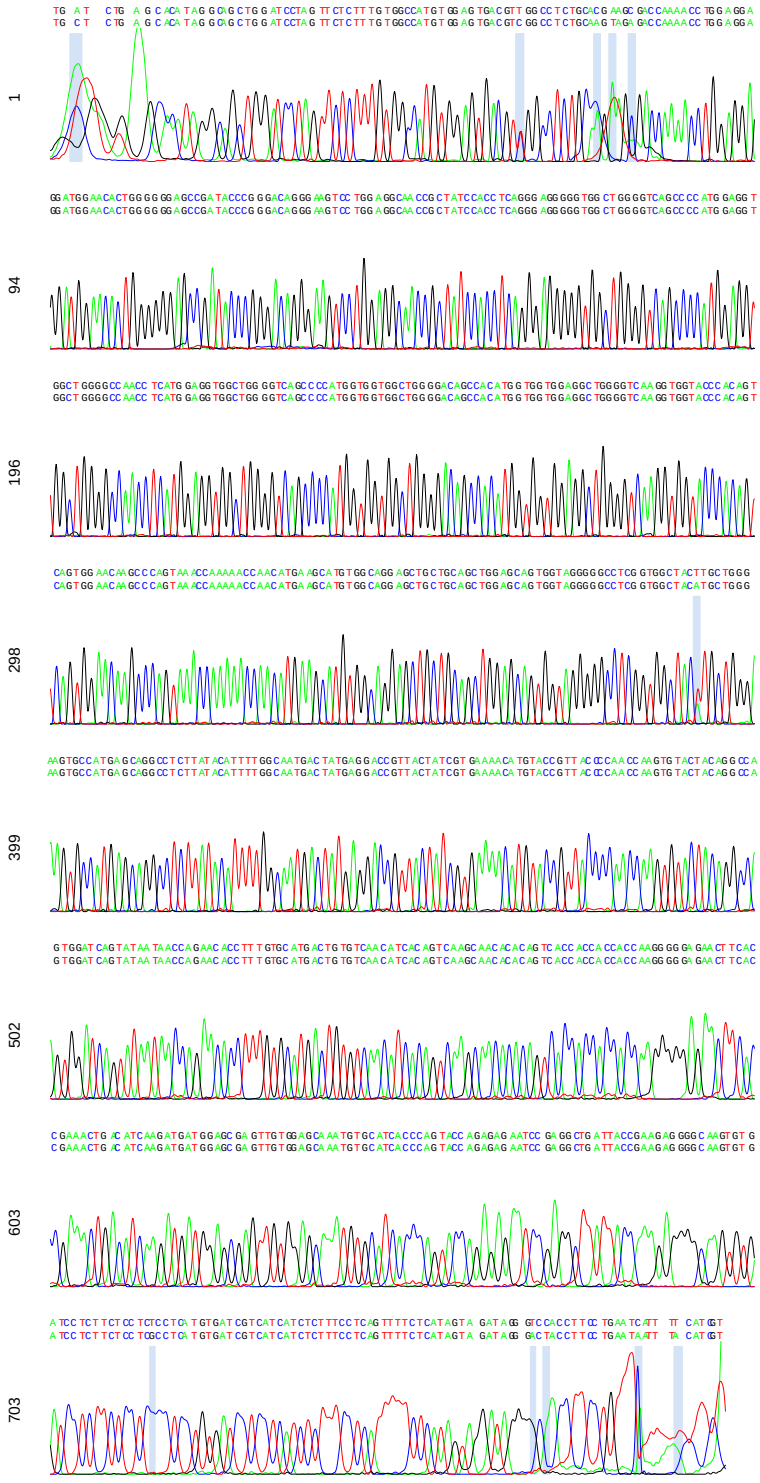
forward : A list of sangerseq objects containing forward reads

reverse : A list of sangerseq objects containing reverse reads

Values:
This function produces two .pdf files in the specified working directory for each sample. One file will contain the chromatogram for the sequence in the sense orientation and the other in the antisense orientation. The name of each file will correspond to the name of each individual sangerseq object in the specified list.
Applications:
The output of this function can be used to qualitatively assess confidence in base calls.
Interpretation:
The output of this function includes graphical representations of raw sequence files and includes the primary and secondary base calls above the chromatograms. Heterozygote calls are highlighted in light blue. The purpose of providing this output is to provide the user with a tool to visually assess the accuracy of base calls. This is especially useful for verifying the identity of potentially polymorphic sites identified in other functions.

These chromatograms can be read similarly to that of other programs (e.g. Sequencher). Note that chromatograms include the ORF and flanking regions. We hope to be able to trim these files to the extent of the ORF in future releases. Currently, the user must manually identify the protein coding region. We found the "search" tool of our PDF reader to be helpful in identifying the start and stop codons.
Dependencies
This function relies on the following inherited functions:
sangerseqR::chromatogram
Example:

```
Chrom(forward, reverse)
```


1
94
196
298
399
502
603
703

3.2.4 Align: Performs Sequence Alignment

Description:
This is one of the central functions of this package. The Align function aligns and clips all sequences to the extent of the reference sequence. The Align function will also convert antisense sequences into their sense-oriented complementary sequence.
Function:
Align(ref, forward, reverse, method)
Arguments:

ref : A DNAString object containing the reference sequence (see: RefSeq)

forward : A list of sangerseq objects containing forward reads

reverse : A list of sangerseq objects containing reverse reads

method : Indicates which alignment algorithm to use ("DECIPHER", "ClustalW", "ClustalOmega", "Muscle")

Details:
Alignments are performed for primary (allele 1) and secondary (allele 2) calls for sense and antisense sequences. Antisense sequences are converted to their sense-oriented complimentary sequence prior to alignment.
Values:
This function produces a list of length 4. Each element contains a matrix of aligned sequences. List[[1]] = forward primary calls, list[[2]] = forward secondary calls, list[[3]] = reverse primary calls, and list[[4]] = reverse secondary calls.

Note that there seems to be a trade-off between alignment accuracy and alignment speed between alignment algorithms. This seems to be particularly true in large datasets. Specifically, we have noticed alignment errors using the profile-to-profile method inherited from the DECIPHER package in large datasets. The other methods seem to improve accuracy; however, computing time did noticeably increase depending on the method chosen. It is up to the user to determine the trade-off between speed and accuracy. We have found ClustalW to work in the vast majority of cases. This method is the slowest, however.
Applications:
This is a required function for the rest of the workflow. This is the core function of the package and performs sequence alignment and processing. All other functions are then used to assess sequence variability for the prion gene sequences contained in the output.
Interpretation:
This function returns several matrices that include alignments for primary and secondary base calls for both orientations. Note that antisense sequences are converted into reverse complementary sequences. Aligned sequences are also

trimmed to the extent of the ORF. This means that all alignment matrices can be subset based on nucleotide position in the protein-coding region of the prion protein gene.
Dependencies:
This function relies on the following inherited functions:
Biostrings::DNAString
Biostrings::DNAStringSet
Biostrings::reverseComplement

This function also relies on the following housekeeping functions provided by the CWDPRNP package:
CWDPRNP::list.DNAStringSet
CWDPRNP::align.method
Example:

```
align <- Align(ref, forward, reverse, method = "ClustalW")
```

```
align[[1]][1:5, 200:219]

##        V200 V201 V202 V203 V204 V205 V206 V207 V208 V209
## ELK464    G    C    T    G    G    G    G    C    C    A
## ELK478    G    C    T    G    G    G    G    C    C    A
## ELK484    G    C    T    G    G    G    G    C    C    A
## ELK485    G    C    T    G    G    G    G    C    C    A
## ELK498    G    C    T    G    G    G    G    C    C    A
##        V210 V211 V212 V213 V214 V215 V216 V217 V218 V219
## ELK464    A    C    C    T    C    A    T    G    G    A
## ELK478    A    C    C    T    C    A    T    G    G    A
## ELK484    A    C    C    T    C    A    T    G    G    A
## ELK485    A    C    C    T    C    A    T    G    G    A
## ELK498    A    C    C    T    C    A    T    G    G    A
```

3.2.5 Poly: Identify Polymorphic Sites in Raw Alignments

Description:
One of the main purposes of this package is to identify variation at SNP loci already associated with CWD. We recognize that many researchers may also be interested in describing novel variation or variation at SNP loci not associated with CWD. This function is meant to identify potentially polymorphic sites within the sequences provided. This function is not necessary if the researcher is only interested in describing variation at CWD-associated loci. This function, along with the ExtractSNPs function, can be used to (1) identify novel polymorphisms and (2) identify genotyping errors that would otherwise give a

false positive for a monomorphic site when used in conjunction with the chromatograms produced earlier.
Function:
Poly(poly, align, ref)
Arguments:

poly : A logical argument that identifies whether a row should be added that indicates the level of polymorphism per site

align : A list object containing the sequence alignment matrices

ref : A DNAString object containing the reference sequence

Values:
This function produces a list of length 2. Each element contains a matrix of aligned sequences. List[[1]] = forward primary and secondary base calls, and list[[2]] = reverse primary and secondary base calls. If poly=T, then the last row will indicate the level of polymorphism (1,...,n; depending on the number of potential alleles).
Applications:
This function is used to identify potentially polymorphic sites in the aligned sequences. This function is not necessary for those solely interested in disease-associated SNPs. This function is helpful for users interested in identifying new SNPs or summarizing variation in other SNPs.
Interpretation:
This function will return objects containing full alignments and the level of polymorphism per site. Sites with polymorphism >1 indicate potentially-polymorphic SNPs. This function can also be used to assess potentially-spurious calls, since it is highly unlikely to observe more than 2 alleles/locus in a single species. Chromatograms can be used in conjunction with this output to rule out spurious calls.
Dependencies:
This function relies on the following inherited functions:
None
Example:

```
polysites <- Poly(poly = T, align = align, ref = ref)
polysites[[1]][72:82, 55:65]

##           55 56 57 58 59 60 61 62 63 64 65
## Elk771_a2  A  G  T  G  A  C  G  T  C  G  G
## Elk773_a1  A  G  T  G  A  C  G  T  C  G  G
## Elk773_a2  A  G  T  G  A  C  G  T  C  G  G
## Elk777_a1  A  G  T  G  A  C  G  T  C  G  G
## Elk777_a2  A  G  T  G  A  C  G  T  C  G  G
## Elk779_a1  A  G  T  G  A  C  G  T  C  G  G
```

```
## Elk779_a2  A  G  T  G  A  C  G  T  C  G  G
## Elk780_a1  A  G  T  G  A  C  G  T  T  G  G
## Elk780_a2  A  G  T  G  A  C  G  T  C  G  G
## Ref        A  G  T  G  A  C  G  T  T  G  G
## Poly       1  1  1  1  1  1  1  1  2  1  1
```

3.2.6 ExtractSNPs: Extract Polymorphic Sites

Description:
This function extracts all potentially polymorphic sites from the results of the full alignment.
Function:
ExtractSNPs(align)
Arguments:

align : A list object produced from the poly() function

Details:
This table is used to guide the researcher in choosing sites that are polymorphic. Note that sequencing errors, which are labeled using some non-nucleic acid indicator, will flag a site as polymorphic. It is up to the researcher to determine which sites should be used for subsequent functions. The row indicating the level of polymorphism in the Poly() function is also a helpful guide. It is highly unlikely any site with a level of polymorphism >2 is a true SNP;however, genotyping errors can cause polymorphic sites to flag with a level of polymorphism >2.
Values:
This function produces a list object containing three matrices: (1) Extracted sites with forward and reverse calls combined, (2) Extracted sites with forward calls only, and (3) Extracted sites for reverse calls only.
Applications:
This function is used to identify potentially-polymorphic sites in the aligned sequences and can be used to truncate the results from the Poly() function to those sites with >1 potential alleles per locus. This function is not necessary for those solely interested in disease-associated SNPs. This function is helpful for users interested in identifying new SNPs or summarizing variation in other SNPs.
Interpretation:
This function will return a matrix containing a subset of sites from the Poly() function that coincide with only polymorphic sites. This function can also be used to assess potentially-spurious calls, since it is highly unlikely to observe more than 2 alleles/locus in a single species. Chromatograms can be used in conjunction with this output to rule out spurious calls. This tool should be used to identify SNPs of interest that can be called in subsequent functions that summarize nucleotide/amino acid diversity.

Dependencies:
This function relies on the following inherited functions:
None
Example:

```
SNPs <- ExtractSNPs(polysites)
SNPs[[1]][151:161, 13:24]

##            394 694 697 699 720 745 752 753 755 756 766 771
## Elk777_a2F "A" "G" "G" "A" "C" "T" "T" "T" "T" "T" "T" "G"
## Elk777_a2R "A" "T" "G" "G" "C" "T" "T" "T" "-" "G" "G" "G"
## Elk779_a1F "A" "G" "G" "A" "C" "T" "T" "T" "T" "T" "C" "T"
## Elk779_a1R "A" "G" "A" "G" "C" "T" "T" "T" "C" "T" "A" "-"
## Elk779_a2F "A" "G" "G" "A" "C" "T" "T" "T" "T" "T" "T" "T"
## Elk779_a2R "A" "T" "G" "A" "C" "T" "T" "T" "-" "G" "G" "G"
## Elk780_a1F "T" "G" "G" "G" "C" "A" "T" "T" "T" "T" "C" "C"
## Elk780_a1R "A" "G" "A" "A" "C" "T" "T" "-" "C" "T" "T" "C"
## Elk780_a2F "A" "G" "G" "G" "C" "C" "T" "T" "T" "T" "C" "T"
## Elk780_a2R "T" "G" "G" "G" "C" "T" "A" "T" "-" "T" "-" "G"
## Ref        "T" "G" "G" "A" "C" "T" "T" "T" "T" "T" "G" "G"
```

3.3 Housekeeping Functions

3.3.1 list.DNAStringSet

Description:
This function converts a list of DNAString objects into a DNAStringSet. This function is used in the align() function and is not directly called by the user.
Dependencies:
This function relies on the following inherited functions:
Biostrings::DNAStringSet

3.3.2 align.method

Description:
This function calls the sequence alignment algorithm specified in the method argument of the align() function and is not directly called by the user.
Dependencies:
Profile-to-profile method: DECIPHER::AlignSeqs
Muscle algorithm: msa:msa
ClustalW algorithm: msa:msa
ClustalOmega algorithm: msa:msa

4 Module 2: Analysis of Genetic Variation

4.1 Introduction

The purpose of this module is to provide several functions that summarize genetic diversity within the sample of PRNP sequences. These functions can either summarize genetic diversity at loci already associated with CWD or for a set of user-defined sites. These analyses can be carried out for a single population, or, alternatively, for a set of populations. This module will provide several outputs: (1) a table of DNA variation among all individuals, (2) a table of amino acid variation among all individuals, and (3) multiple tables that summarize allele and genotype frequencies per population.

4.2 Core Functions

4.2.1 NucTab: Create Table of Nucleotides per Individual

Description:
This function extracts the genotypes for disease-associated SNP loci or those specified by the user.
Function:
NucTab(x, align, nuc)
Arguments:

x : A character string specifying the species of interest

align : A list object containing aligned sequences

nuc : A numeric vector containing the loci of interest (default=Null if using predefined loci)

Details:
x can be specified in several ways:
Cervus canadensis (x=c("CECA"))
Odocoileus virginianus (x=c("ODVI"))
Odocoileus hemionus (x=c("ODHE"))
User-defined SNPs (x=c("other")
If x="other", then the user must specify SNPs of interest. This is done through the nuc argument (ex. nuc=c("63")).

Note that multiple loci can be specified using the nuc argument. While this works for this function, there is a known bug that prevents the simultaneous summarization of genotype/allele frequencies in a downstream function. The current work-around is to specify each locus one at a time. We are currently working to fix this bug in future releases.
Values:
This function produces a character matrix that lists the genotype(s) for each

locus. This function will produce an "Error" for any genotypes where the reverse complement does not match the sense sequence.
Applications:
This function produces a table of genotypes indexed by individual and is used by subsequent functions to summarize amino acid variability.
Interpretation:
Each value of this table represents a genotype for a specified SNP locus within the PRNP gene. These are either disease-associated loci, if using the default settings for each species or are user-specified loci. The matrix will contain a column for each loci and each row is the individual sample.
Dependencies
This function relies on the following inherited functions:
None
Example:

```
## Option 1: Disease-associated loci
Nucleotides <- NucTab(x = c("CECA"), align = align)
head(Nucleotides)

##        nuc394
## ELK464 "A/A"
## ELK478 "A/A"
## ELK484 "A/T"
## ELK485 "A/A"
## ELK498 "A/A"
## ELK500 "A/T"

## Option 2: User-associated loci
Nucleotides2 <- NucTab(x = c("other"), align = align, nuc = c((63)))
head(Nucleotides2)

##        63
## ELK464 "C/C"
## ELK478 "C/C"
## ELK484 "Error"
## ELK485 "C/C"
## ELK498 "C/C"
## ELK500 "Error"
```

4.2.2 CodTab: Create Table of Amino Acids per Individual

Description:
This function extracts the amino acid type for disease-associated SNP loci or those specified by the user.
Function:
CodTab(x, Tab, align)

Arguments:

x : A character string specifying the species of interest

Tab : A matrix containing SNP genotypes per locus

align : A list object containing aligned sequences.

Details:
x can be specified in several ways:
Cervus canadensis (x=c("CECA"))
Odocoileus virginianus (x=c("ODVI"))
Odocoileus hemionus (x=c("ODHE"))
User-defined SNPs (x=c("other"))
Values:
This function produces a character matrix that lists the amino acid type for each locus.
Applications:
This function produces a table of amino acid indexed by individual and is used by subsequent functions to summarize amino acid variability.
Interpretation:
Each value of this table represents an amino acid type for a specified SNP locus within the PRNP gene. These are either disease-associated loci, if using the default settings for each species or are user-specified loci. The matrix will contain a column for each loci and each row is the individual sample.
Dependencies:
This function relies on the following inherited functions:
Biostrings::translate
Example:

```
## Option 1: Disease-associated loci
Codons <- CodTab(x = c("CECA"), Tab = Nucleotides)
head(Codons)

##        cod132
## ELK464 "M/M"
## ELK478 "M/M"
## ELK484 "M/L"
## ELK485 "M/M"
## ELK498 "M/M"
## ELK500 "M/L"

## Option 2: User-associated loci
Codons2 <- CodTab(x = c("other"), Tab = Nucleotides2, align = align)
head(Codons2)
```

```
##        21
## ELK464 "V/V"
## ELK478 "V/V"
## ELK484 "V/V"
## ELK485 "V/V"
## ELK498 "V/V"
## ELK500 "V/V"
```

4.2.3 write.Tabs: Write Individual Results

Description:
This function is used to write an Excel document (.xlsx format) containing a table of SNP genotypes and amino acid types, referenced by an individual identifier.
Function:
write.Tabs(x, NucTab, CodTab, filename)
Arguments:

x : A character string specifying the species of interest.

NucTab : A matrix of SNP genotypes for each sampled individual

CodTab : A matrix of amino acid types.

filename : A character string containing the desired excel file name, with extension.

Details:
x can be specified in several ways:
Cervus canadensis (x=c("CECA"))
Odocoileus virginianus (x=c("ODVI"))
Odocoileus hemionus (x=c("ODHE"))
User-defined SNPs (x=c("other"))
Values:
Produces a .xlsx file with two worksheets. Worksheet 1 (SNPs) contains the NucTab matrix. Worksheet 2 (AminoAcids) contains the CodTab matrix.
Applications:
This function saves the output of NucTab() and CodTab().
Interpretation:
Output should be interpreted in same way as NucTab() and CodTab().
Dependencies:
This function relies on the following inherited functions:
openxlsx::write.xlsx
Troubleshooting:
If R returns the following error message, "Error: zipping up workbook failed." follow these instructions:

1) Make sure that Rtools is installed.
2) Make sure the R_ZIPCMD is set to the file in Rtools with the zip file. This can be done using the Sys.setenv(R_ZIPCMD = path) command. For example: Sys.setenv(R_ZIPCMD = "C:/My Documents/Rtools/bin/zip")
Example:

```
# change path to directory containing Rtools
Sys.setenv(R_ZIPCMD = "C:/My Documents/Rtools/bin/zip")
write.Tabs(x = "CECA", Nucleotides, Codons, filename = "test.xlsx")
```

4.2.4 PropTab: Genotype/Allele Frequency Tables

Description:
This function is used to calculate genotype and allele frequencies for disease-associated or user-defined loci. This function can produce frequency tables for the global population or for a series of populations if a vector of population labels are provided.
Function:
PropTab(Pop, x, CodTab, PopLabels)
Arguments:

Pop : A logical vector indicating whether genotype/allele frequencies should be summarized by population

x : A character string specifying the species of interest

CodTab : A character matrix containing the amino acid types for each individual

PopLabels : A character vector containing the population labels (required if Pop=T)

Details:
x can be specified in several ways:
Cervus canadensis (x=c("CECA"))
Odocoileus virginianus (x=c("ODVI"))
Odocoileus hemionus (x=c("ODHE"))
User-defined SNPs (x=c("other"))

PopLabels must be specified if Pop=T. The user should make sure that the PopLabels and individual IDs in CodTab are in the same order.

IMPORTANT NOTE: This function has a known bug that only allows a user to pass one user-defined SNP at a time. For example, if x="other" you can only run SNP 63 or SNP 394, not both at the same time. Make sure to truncate

NucTab and CodTab functions to only pull one SNP until problem is resolved.
Values:
This function produces a list object containing several matrices (varies by species and/or loci specified). This list contains matrices that summarize genotype frequencies across all loci and populations, a matrix summarizing allele frequencies across all loci and populations, and matrices summarizing allele frequencies for each locus and population.
Applications:
This function is used to summarize allele/genotype frequencies for the entire population of interest and/or a list of specified populations. The tables produced by this function are needed for spatial analyses.
Interpretation:
The output of this function is a list that summarizes genotype/allele frequencies for the sample of interest and all specified populations.
Dependencies:
This function relies on the following inherited functions:
None
Example:

```
## Without Population Labels
Prop <- PropTab(Pop = F, x = c("CECA"), CodTab = Codons)
Prop[[1]]

##                   cod132
## Homozygote Major    0.75
## Heterozygote        0.25
## Homozygote Minor    0.00

## With Population Labels
Popxy <- read.csv("LocalityInfo.csv")
Popxy <- Popxy[order((Popxy[, 1])), ]
labels <- Popxy[, 1]
Popxy <- Popxy[, -1]
PopLabels <- as.character(Popxy[, 1])
Prop2 <- PropTab(Pop = T, x = c("CECA"), CodTab = Codons, PopLabels = PopLabels)
Prop2[["Genotype_All"]]

##                         Homozygote Major Heterozygote
## St Marys and Gray Hill              0.5          0.5
## Bear Hollow                         0.7          0.3
## Medix                               0.9          0.1
## Grant                               0.9          0.1
##                         Homozygote Minor
## St Marys and Gray Hill                0
## Bear Hollow                           0
```

```
## Medix                                    0
## Grant                                    0
```

4.2.5 write.PropTab: Write Genotype/Allele Frequency Tables

Description:
This function is used to write an Excel document (.xlsx format) containing allele and genotype frequency tables, referenced by population/loci.
Function:
write.PropTab(PropTab, filename)
Arguments:

PropTab : A list item containing allele and genotype frequency tables

filename : A character string containing the desired excel file name, with extension

Values:
Produces a .xlsx file with a worksheet for every element of the list produced by PropTab.
Applications:
This function is used to export the results of the PropTab() function.
Interpretation:
The output of this function is an excel file with worksheets for each element of the PropTab() function output.
Dependencies:
This function relies on the following inherited functions:
openxlsx::write.xlsx
Troubleshooting:
If R returns the following error message, "Error: zipping up workbook failed." follow these instructions:
1) Make sure that Rtools is installed.
2) Make sure the R_ZIPCMD is set to the file in Rtools with the zip file. This can be done using the Sys.setenv(R_ZIPCMD = path) command. For example: Sys.setenv(R_ZIPCMD = "C:/My Documents/Rtools/bin/zip")
Example:

```
Sys.setenv(R_ZIPCMD = "C:/My Documents/Rtools/bin/zip")
write.PropTab(Prop, filename = "ExampleProp.xlsx")
```

5 Module 3: Spatial Analyses for Prion Gene Variation

5.1 Introduction

This module currently provides three functions that can be used to perform spatial analyses of prion genotype and allele frequency data. This suite of functions was developed in order to allow the user to visualize trends to guide further analyses. We hope to provide additional functions in future updates that allow users to statistically test for genotype/allele frequency differences among sampling areas and populations. This module is flexible, and allows users to perform individual-based assessments of spatial patterns, through kernel-density estimation; or to perform population-based assessments, through the use of frequency plots and spatial interpolation.

Note: All following examples refer to analysis on default SNPs (disease-associated loci). It is possible to pass these functions over user-defined SNPs as well.

Important Note: Data Preparation
This module requires a matrix of geographic coordinates and a shapefile of the study area for data visualization. The geographic coordinates and shapefile MUST be in the same spatial projection, otherwise these functions will fail to produce usable results. The rgdal and sp packages can be used to assign geographic and projected coordinate systems in R. This is beyond the purpose of this package; however, there are a number of tutorials on how to assign coordinate systems to spatial objects. See the Manual of Applied Spatial Ecology, which can be found here: http://ecosystems.psu.edu/research/labs/walter-lab/manual/complete-manual-pdf, for a tutorial on how to manipulate the coordinate systems of spatial objects.

5.2 Core Functions

5.2.1 KDE: Individual-based Analysis Through Kernel Density Estimation

Description:
This function will produce a raster object containing a density map for specified prion alleles/genotypes.
Function:
KDE(method, Sus, locus, genotypes, shapefile, Codons, pixel, XYdat)
Arguments:

method : A character string that indicates whether to create a density map of genotypes or alleles

Sus : A character string indicating whether to map more susceptible alleles/genotypes or less susceptible alleles/genotypes

locus : A character string indicating codon identity

genotypes : A character vector containing possible genotypes per locus

shapefile : A character string indicating the name of a GIS shapefile in the working directory

Codons : A matrix of amino acid types for each sampled individual

pixel : A numeric value specifying the number of grid-lines used to create raster pixels

XYdat : A matrix of individual locations

Details:
Currently, the only available option for the Method argument is "Genotype". Future updates will provide additional functionality for allele-based assessments.

The Sus argument may be specified as:
Sus="More" for the most susceptible genotypes
Sus="Less" for the least susceptible genotypes

Elements of XYdat matrix must correspond to the individual identifiers in the Codons matrix.
Values:
This function produces a raster object containing a kernel density map for the defined allele/genotype and level of susceptibility. Note that the kernel density map is created using a rectangular bounding box around the shapefile, but the final extent is clipped to the actual shapefile in order to avoid interpolation to points outside of the study area.
Applications:
This function is used to produce a two-dimensional density plot of a specified genotype or allele. This function is best suited for a sample with explicit spatial coordinates for every individual. These data are best suited for assessing fine-scale heterogeneity in prion variability.

This function will not work if individuals are referenced by population.
Interpretation:
Kernel density surfaces represent the relative intensity of a spatial point process. This function produces a raster that represents the relative densities of individuals with the genotype/allele of interest. Intensity can be interpreted as the expected number of individuals with a given genotype/allele per unit area (pixel area). Optimal bandwidth is chosen using the mean square error minimization criterion outlined in Berman and Diggle (1989). The plot produced by the KDE() function highlights the bandwidth value that minimizes the mean

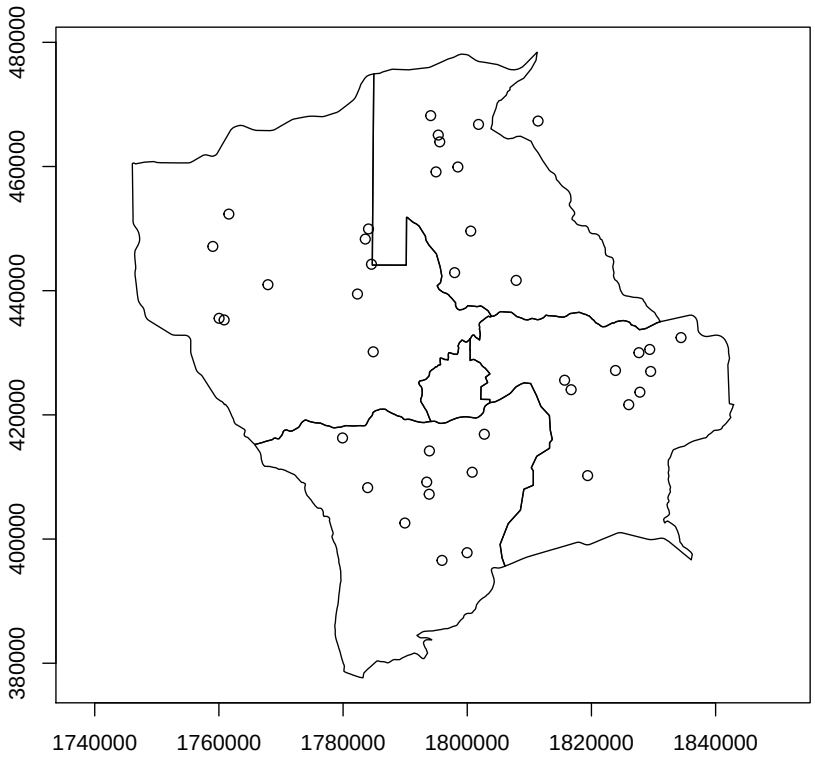
square error of the kernel smoothing estimator. See Diggle (2003) for details.
References:

1. Diggle, P.J. (2003) Statistical Analysis of Spatial Point Patterns. Oxford University Press Inc., New York, New York, NY.

Dependencies:
This function relies on the following inherited functions:
rgdal::readOGR
raster::extent
splancs::mse2d
splancs::kernel2d
raster::raster
raster::crop
raster::rasterize
raster::mask

Note that all plotting functions are inherited from the raster package.
Example:

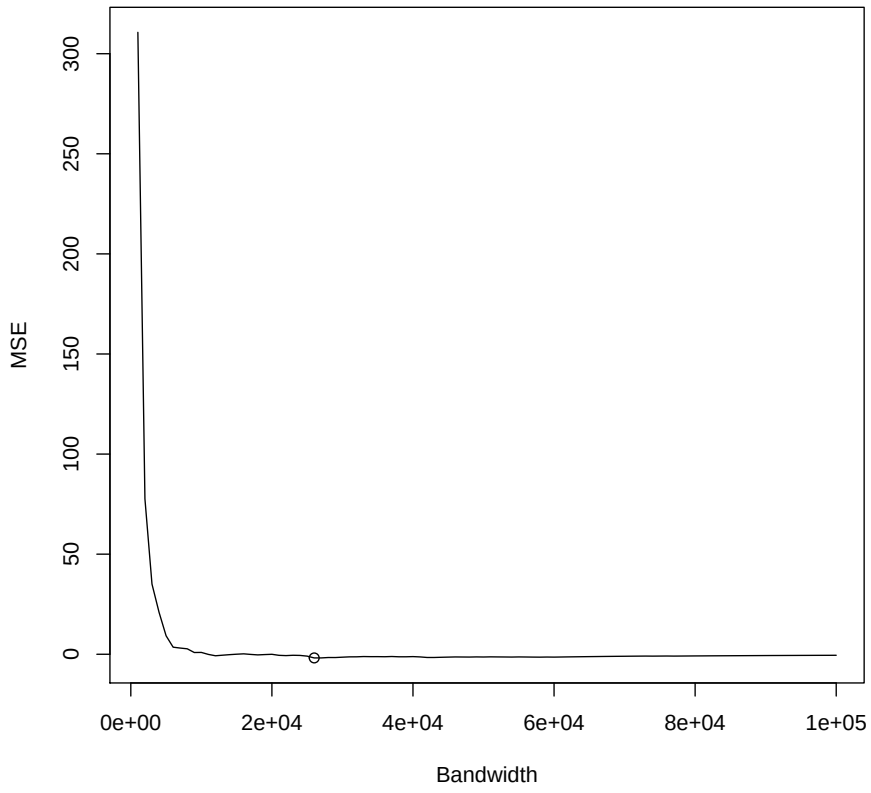
```
# Add spatial noise to XY data to convert population
# coordinates into individual coordinates for KDE example
Xj <- jitter(Popxy[, 2], factor = 6)
Yj <- jitter(Popxy[, 3], factor = 6)
XYj <- cbind(Xj, Yj)
Popxy <- cbind(as.numeric(XYj[, 1]), as.numeric(XYj[, 2]))
colnames(Popxy) <- c("x", "y")
rownames(Popxy) <- labels
# Load polygon of sampling area for mapping
poly <- readOGR(".", layer = "ElkSelection2")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "ElkSelection2"
## with 5 features
## It has 11 fields
## Integer64 fields read as strings:  Hunt_Zone Total

plot(poly, axes = T)
points(Popxy)
```




```
# Note: The jitter() function was used to convert
# population-based spatial data to something resembling
# individual-based genetic data for the purpose of this
# example.  Explicit XY coordinates were not collected for
# these samples so spatial noise was added in order to
# showcase the KDE() function.  The kernel density map
# produced by the following functions may not exactly match
# the example in the publication due to the fact that the
# jitter() function will not create the same spatial
# coordinates every time it is run.
```

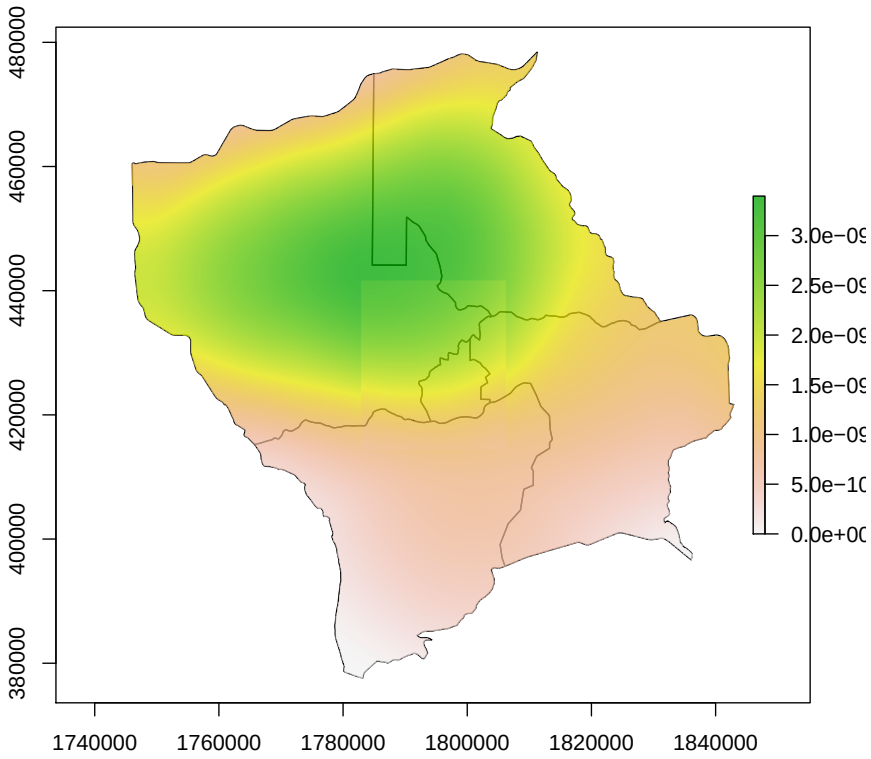
```
# Produce and map KDE raster
kdetest <- KDE(method = "Genotype", Sus = "Less", locus = "cod132",
    genotypes = c("M/M", "M/L", "L/L"), shapefile = "ElkSelection2",
    Codons = Codons, pixel = 1000, XYdat = Popxy[, 1:2])
```

```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "ElkSelection2"
## with 5 features
## It has 11 fields
## Integer64 fields read as strings:  Hunt_Zone Total
```




```
## Xrange is  1746055 1842924
## Yrange is  377654.2 478411.8
## Doing quartic kernel
```

```
plot(poly, axes = T)
plot(kdetest, add = T, alpha = 0.75)
```



5.2.2 IDW: Two-Dimensional Projection of Population Allele/Genotype Frequencies Through Spatial Interpolation

Description:
This function will produce a raster object containing an interpolated surface of allele/genotype frequencies.
Function:
IDW(method, Sus, PropTab,Centroid, X, Y, power, shapefile)
Arguments:

method : A character string that indicates whether to create a density map of genotypes or alleles

Sus : A character string indicating whether to map more susceptible alleles/genotypes or less susceptible alleles/genotypes

PropTab : A matrix containing the population-level genotype frequencies

Centroid : A matrix containing the geographic coordinates for population centroids

X : A numeric argument determining cell-size for spatial grid on x-axis

Y : A numeric argument determining cell-size for spatial grid on y-axis

power : A numeric argument determining the shape of the inverse distance weighting function for spatial interpolation

shapefile : A character string indicating the name of a GIS shapefile in the working directory

Details:
Currently, the only available option for the Method argument is "Genotype". Future updates will provide additional functionality for allele-based assessments.

The Sus argument may be specified as:
Sus="More" for the most susceptible genotypes
Sus="Less" for the least susceptible genotypes

The power parameter is decided by the user. Low values will produce a "bulls-eye" effect of concentric circles around centroids. Large values will return a surface resembling tiles of a Voronoi diagram. The "best" value is one that produces a smoothed surface without the substantial "blotting" or "tiling". We recommend beginning with power = 2 and varying this value until a smoothed surface is returned.

Note that there is a trade-off between computing power and "smoothness" related to the grid size parameters. Ideally, the user would want to specify the smallest grid size possible to avoid pixelation. This, however, can severely increase computing time.

Note that all plotting functions are inherited from the raster package.
Values:
This function produces a raster object containing an interpolated surface representing of allele/genotype frequencies.
Applications:
This function is used to produce a two-dimensional surface representing the expected proportion of a particular genotype/allele. This function is best suited for a sample where genotype/allele frequencies are summarized by subpopulation and mapped to common coordinates.

This function will not work for individually georeferenced data.
Interpretation:
The output of this function is a surface that represents the proportion of individuals within an area that are expected to have the specified genotype/allele. Another way to think of this is that a cell value for this raster represents the

probability that an individual sampled from that area will have the specified genotype.

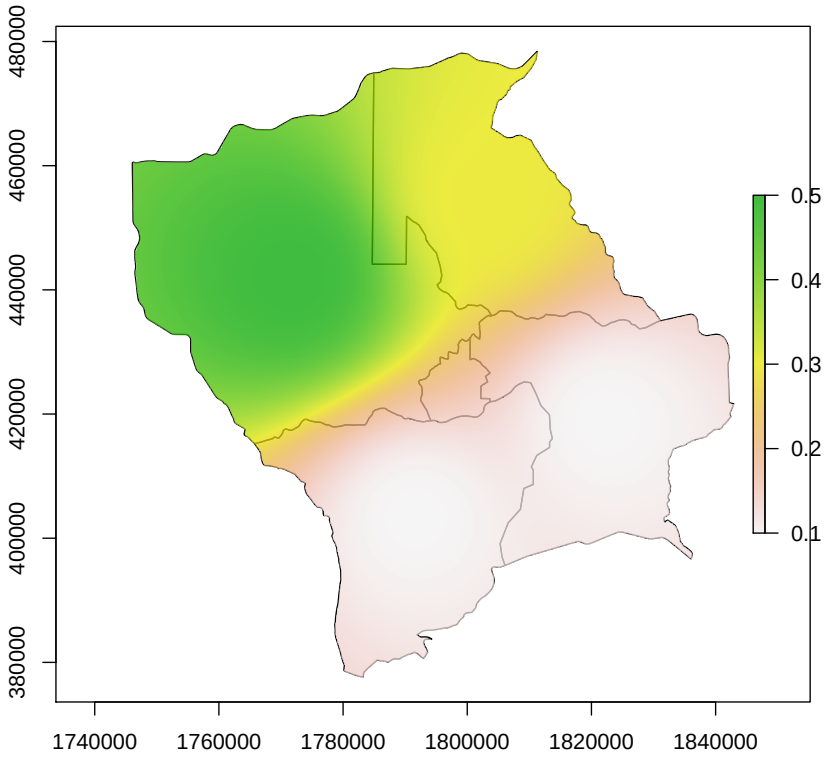
Dependencies:

This function relies on the following inherited functions:
rgdal::readOGR
raster::extent
sp::coordinates
sp::gridded
gstat::idw
raster::raster
raster::crop
raster::rasterize
raster::mask

Example:

```
# Extract table from PropTab list
GenProp <- Prop2[["Genotype_All"]]
# Format Centroid File
Popxy <- read.csv("LocalityInfo.csv")
Popxy <- Popxy[order((Popxy[, 1])), ]
labels <- Popxy[, 1]
Popxy <- Popxy[, -1]
PopLabels <- as.character(Popxy[, 1])
Popxy <- unique(Popxy)
Cent <- Popxy
Cent <- cbind(as.numeric(Cent[, 2]), as.numeric(Cent[, 3]))
colnames(Cent) <- c("x", "y")
# IDW
idwraster <- IDW(method = "Genotype", Sus = "Less", PropTab = GenProp,
    Centroid = Cent, X = 100, Y = 100, power = 3, shapefile = "ElkSelection2")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "ElkSelection2"
## with 5 features
## It has 11 fields
## Integer64 fields read as strings:  Hunt_Zone Total
## [inverse distance weighted interpolation]

# Map Results
poly <- rgdal::readOGR(".", layer = "ElkSelection2")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "ElkSelection2"
## with 5 features
## It has 11 fields
## Integer64 fields read as strings:  Hunt_Zone Total
```

```
plot(poly, axes = T)
plot(idwraster, add = T, alpha = 0.75)
```



5.2.3 Pie: Floating Pie Chart of Allele/Genotype Frequencies

Description:
This function will produce a plot of population allele/genotype frequencies. This will be in the form of a pie chart that overlays population centroids.
Function:
Pie(shapefile, PropTab, XYdat, radius)
Arguments:

shapefile : A character string indicating the name of a GIS shapefile in the working directory

PropTab : A matrix containing the population-level genotype frequencies

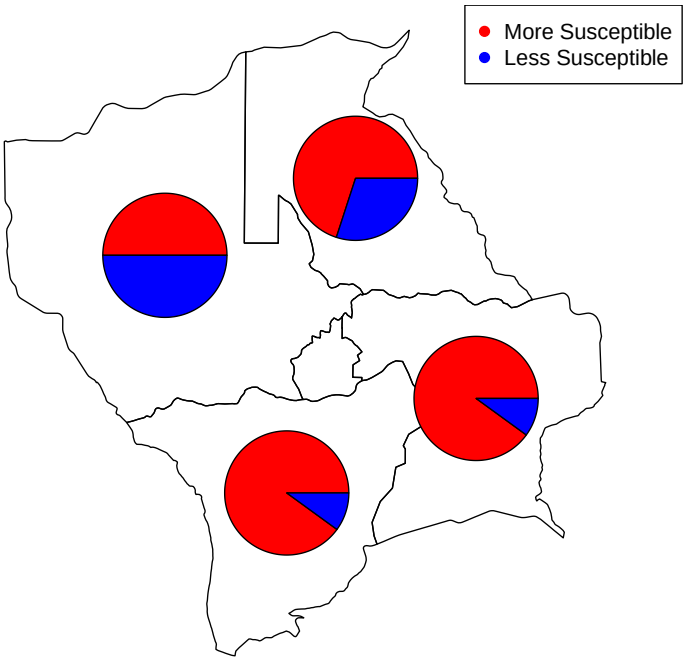
XYdat : A matrix containing population centroid coordinates

radius : A numeric value, which is a scaling constant to decide pie chart size

Details:
Note that the size of the pie chart will be proportional to the sample size for each population. The radius argument provides a scaling constant that will modulate the size of each pie chart, although the pie chart will always be proportional to sample size. Try different radius values until you find one that best fits the study area.
Values:
This function produces a map of population allele/genotype frequencies. Pie charts are mapped to the population centroid.
Application:
This function is used to map genotype/allele frequencies per subpopulation.

This function will not work for individually georeferenced data.
Interpretation:
This map is the easiest spatial figure to interpret. It simply represents the allele/genotype frequencies of a subpopulation, mapped to its centroid. Pie charts are scaled by sample size.
Dependencies:
rgdal::readOGR
plotrix::floating.pie
Example:

```
Pie(shapefile = "ElkSelection2", PropTab = GenProp, XYdat = Cent,
    radius = 10000)

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "ElkSelection2"
## with 5 features
## It has 11 fields
## Integer64 fields read as strings:  Hunt_Zone Total
```


More Susceptible
Less Susceptible

6 Additional Details

6.1 Package Limitations

The CWDPRNP package provides a set of tools for performing sequence alignment and analysis, for summarizing allele and genotype frequencies at disease-associated or user-specified loci, and for performing several basic spatial analyses. This package is also flexible enough to handle single-gene sequences that are not cervid prion sequences, so long as the user has a reference sequence to align to. This package does have several limitations that we hope to address in future releases. These are outlined here:

- The current version only has functionality for Applied Biosystems file formats.
- Current tools for identifying SNPs are qualitative and require considerable user input.
- Alignment protocols rely on a pre-defined reference sequence, which may not be available in all cases.
- Analyses (both genetic and spatial) are largely qualitative and descriptive in nature. We hope to future functions to include more rigorous statistical and population genetics approaches.
- Current analyses are for a single locus at a time. We hope to provide additional functionality for haplotype and multi-locus analyses in the future.

6.2 Future Directions

- Incorporate other sources of data (e.g. .fasta, .scf formats).
- Allow for the integration of phred scores, which will provide a quantitative method for assessing sequence accuracy and identifying additional polymorphic sites.
- Provide a function to produce consensus sequences that could be used for sequence alignment, analysis, and summarization.
- Add additional functions that can test for statistical differences between subpopulation genotype frequencies (e.g. pairwise Fisher tests, etc.).
- Provide tools to perform haplotype and multi-locus analyses.
- Provide tools and direction for performing multi-species sequence analysis.

7 Appendix 1: Description of Example Data

7.1 Elk Prion Gene Sequences

We have included PRNP sequences from 40 Rocky Mountain elk samples (*Cervus canadensis*), collected in northern Pennsylvania, USA. These samples were chosen to highlight the functionality of the spatial module of the CWDPRNP package and do not represent the underlying allele or genotype frequencies of the sample from which they were drawn.

7.2 Sequence Quality Control

Sequence quality was assessed by measuring the percentage of base pairs within a sequence with a Phred score ≥ 20. All sequences have $> 90\%$ of bases with a Phred score ≥ 20. Visual inspection of chromotagrams also suggested that sequence quality was adequate for use in this tutorial. We also reamplified and resequenced 20% of the total sample and ensured that results could be replicated between the original and replicate dataset. We found that results were able to be replicated for the loci used in this example (nucleotides 63 and 394). It is important to note, however, that three samples failed to give callable results for nucleotide 63 (two in original and one in replicate sample) and these results were discarded from the QAQC analysis. We feel that this is most likely a consequence of nucleotide 63 being located within the terminal end of the antisense sequence, which has a greater number of base pairs with lower quality scores and is more likely to produce an error related to the sense and antisense not being complimentary (listed as "Error" in NucTab and CodTab output). We decided to include nucleotide 63 in the example because it was one of the only variable sites that was not associated with disease, and it was only used to highlight the ability of the NucTab and CodTab functions to extract user-defined SNPs. This also highlights the present internal controls for ensuring accurate results. We are confident in the results from the locus used in the spatial analysis (nucleotide 394, codon 132) as the base calls matched 100% between the original and replicate dataset.

7.3 Spatial Data

We provide geographic coordinates for the 40 elk samples and a shapefile representing the management area where they were sampled. The shapefile is made up of four management sub-units and geographic coordinates correspond to the centroid of the sub-unit in which each individual was sampled. All spatial data is in the "Lambert Conformal Conic" projection.