QFASA R Package

by

Justin Kamerman

MSc Computer Science, UNISA, 2004
BSc Mechanical Engineering, Univ. of Witwatersrand, 1993

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MSc Statistics

In the Graduate Academic Unit of Mathematics and Statistics

THE UNIVERSITY OF NEW BRUNSWICK

May, 2019

Abstract

The primary contribution of this project is to package R source code created to support the fitting and evaluation of Quantitative Fatty Acid Signature Analysis (QFASA) models into a FOSS module available on CRAN. The existing code is widely used by the QFASA community but is inconsistently documented and maintained. This makes it difficult for new users to get up to speed with new and current QFASA methodologies, and to distribute code fixes and improvements.

Creating an R package is a well-defined process and encourages the use of software engineering best practices and the production of well-documented modules that are easy to install and maintain.

This report describes the process of diet estimation via the QFASA methodology and reviews some of the underlying statistical methodologies. We detail the R packaging process and our interaction with CRAN to publish the package, and our implementation of parallel computing methods to improve the speed and efficiency of model inference by making use of multi-core processors. Finally, for comparison, we review a similar QFASA module, *qfasar*, which was released subsequently.

Table of Contents

Abstract **ii**

Table of Contents **vi**

List of Tables **vii**

List of Figures **viii**

Abbreviations **ix**

1 Overview **1**

2 Quantitative Fatty Acid Signature Analysis **3**

2.1 Introduction 3

2.2 Diet Estimation 3

2.2.1 Fatty Acids 5

2.2.2 Model 5

2.2.3 Distance Measures 7

2.2.4 Standard Error of Estimates 9

2.2.5 Simulation 10

2.3 Compositional Data 13

2.3.1 Aitchison Simplex 13

2.3.2 Issues with Compositional Data 14

2.3.3 Parametric Models for Compositional Data 16

2.4 Modelling Compositional Data with Zeros 18

3 Inference from QFASA **24**

3.1 Introduction 24

3.2 Confidence Intervals 24

3.2.1 Non-parametric Bootstrap 25

3.2.2 Parametric Bootstrap 26

3.2.3 Bias 27

3.3 Testing for Differences in Compositional Data 28

4 QFASA R Package **29**

4.1 Overview 29

4.2 The R Language 29

4.3 R Packages 31

4.3.1 CRAN 32

4.3.2 Package Structure 32

4.3.3 R Code 34

4.3.4 Metadata 35

4.3.5 Documentation 38

4.3.6 Vignettes 40

4.3.7 Testing 40

4.3.8 Namespaces 43

4.3.9 Other Components . . . 43
4.4 QFASA . . . 44
4.4.1 Finding Roots . . . 46
4.4.2 Parallelization . . . 48
4.5 CRAN Submission . . . 48
4.6 *qfasar* R package . . . 50

5 Conclusion 53

References 57

A QFASA R Package version 1.0.3 58

QFASA . . . 58
AIT.dist . . . 59
AIT.more . . . 60
AIT.obj . . . 60
beta.meths.CI . . . 61
bias.all . . . 64
CC . . . 66
chisq.CA . . . 66
chisq.dist . . . 67
CS.more . . . 68
CS.obj . . . 68
create.d.mat . . . 69
FAset . . . 70
KL.dist . . . 71

KL.more 71

KL.obj 72

MEANmeth 73

p.QFASA 74

prey.cluster 76

pseudo.pred 77

QFASA.const.eqn 78

split.prey 79

testfordiff.ind.boot 80

testfordiff.ind.boot.fun 81

testfordiff.ind.pval 81

Vita

List of Tables

4.1 Simulation 95% CIs average widths by prey species 47

4.2 Simulation 95% CIs coverage proportions by prey species 47

List of Figures

2.1 Hierarchical cluster analysis on mean FA signature using Aitchison distance 11

2.2 Hierarchical cluster analysis on mean FA signature using Kullback-Liebler distance 12

2.3 Hierarchical cluster analysis on mean FA signature using Chi-Square distance 12

4.1 KDNuggets Analytics, Data Science, Machine Learning Software Poll 2016-2018 30

4.2 Creating a package template with *Devtools* 34

4.3 DESCRIPTION file for *QFASA* package 37

4.4 CRAN web page generated from package metadata 37

4.5 *Roxygen* markup tags on package function 39

4.6 *Roxygen* function comments available via help() function 39

4.7 Portion of vignette markup for the *QFASA* package 41

4.8 Part of the *QFASA* vignette on CRAN 41

4.9 Definition of *testthat* test function 42

4.10 Execution of *testthat* test function 43

4.11 *QFASA* confidence interval estimation execution time as a fraction of the time taken to execute on a singel core 49

4.12 CRAN downloads of QFASA versus qfasar packages 52

Abbreviations and Notation

AIT	Aitchison statistical distance measure used to fit the QFASA model
API	Application Programming Interface
C_j	calibration coefficent for fatty acid type j
CRAN	The Comprehensive R Archive Network
CS	Chi-square statistical distance measure used to fit the QFASA model
FA	Fatty acid
FOSS	free open-source software
I	number of prey species represented in a diet composition
j	fatty acid index $1..n_{fa}$
k	prey species index $1..I$
l	predator index $1..n_s$
KL	Kulback-Liebler statistical distance measure used to fit the QFASA model
MLE	Maximum Likelihood Estimator
n_{fa}	number of fatty acid types in the QFASA model
n_k	number of prey of species k in the prey database
n_s	number of individual predators for which we require diet estimates
p_k	estimated diet proportion of prey species k
POSIX	Portable Operating System Interface
QFASA	Quantitative Fatty Acid Signature Analysis
R	R statistical computing language
RMSE	Relative Mean Squared Error
ZIB	Zero-inflated Beta distribution

Chapter 1

Overview

Quantitative Fatty Acid Signature Analysis (QFASA) is a statistical model that aims to provide quantitative estimates of the proportion of prey species in the diets of individual predators based on their fatty acid signatures. The primary contribution of this project is to package R source code created in support of research into and application of QFASA diet estimation models [14, 22–25] into a FOSS module available on CRAN. Code developed in support of these studies is widely used by the QFASA community but is inconsistently documented and maintained. This makes it difficult for new users to get up to speed with new and current QFASA methodologies, and to distribute code fixes and improvements.

Creating an R package is a well-defined process and encourages the use of software engineering best practices and the production of well-documented modules that are easy to install and maintain.

This report describes the process of diet estimation via the QFASA methodology and reviews some of the underlying statistical methodologies. We detail the R packaging process and our interaction with CRAN to publish the package, and our implementation of parallel computing methods to improve the speed and efficiency of model inference by making use of multi-core processors. Finally, we review, for comparison, a similar QFASA module [5].

Chapter 2 is an overview of the motivation and theories behind the use of fatty acid signatures to estimate a predator's diet. We review relevant literature supporting the development of the original QFASA R code and recent extensions. This includes different methods employed to handle essential zeros in the QFASA diet estimates.

In Chapter 3 we describe QFASA mixture models that incorporate essential zeros and detail specifically an improved method of handling them based on the *zero-inflated beta* distribution. This method is the basis of the bootstrap simulations used in the QFASA R package to generate confidence intervals for the true diet.

Chapter 4 details the implementation of our R package encapsulating and extending QFASA functions from [22–24]. We describe additions to the original source code required to create the package as well as the CRAN submission process. We detail the optimizations made to leverage third-party root-finding tools and multicore processors. We also review a complementary R package which leverages the same underlying theories but with a more specific focus on conducting predator simulations.

Finally, in Chapter 5, the contributions of the QFASA R package towards making the QFASA diet estimation methods more accessible and scalable to a wider group of practitioners are summarized.

Appendix A contains detailed application programming interface (API) documentation on the functions implemented in our R QFASA package.

Chapter 2

Quantitative Fatty Acid Signature Analysis

2.1 Introduction

This chapter is an overview of the motivation and methods underlying Quantitative Fatty Acid Signature Analysis (QFASA), the use of fatty acid signatures (proportions of individual fatty acids that sum to one) to estimate a predator's diet. We review literature supporting the development of the original QFASA implementation and subsequent extensions, including methods to handle essential zeros and to perform inference on predator diet estimates. These methods form the basis of our QFASA R package.

2.2 Diet Estimation

Direct observation of predators feeding over long periods of time is difficult, particularly for marine mammals. For this reason, indirect methods may be necessary to reconstruct diet. Indirect methods mostly rely on the recovery of digestion-resistant prey structures from feces, stomach contents, or spewings. However, differential rates of digestion among prey species can significantly bias estimates. Also, such methods provide only a snapshot of the most recent meal, may not be representative of long term diet, and are lethal [14]. Other techniques involve analysis of the stable isotope ratios of carbon and nitrogen and do not depend on the recovery of digestant-resistant hard parts but have demonstrated limited success and usually cannot determine the species composition of the diet.

QFASA [14] is a statistical model that aims to provide quantitative estimates of the proportion of prey species in the diets of individual predators based on their fatty acid (FA) signatures, obtained through a non-lethal biopsy, and is intended to address shortcomings of the techniques described above. QFASA relies on a distance metric rather than a model-based formulation to estimate the most likely diet proportions. It was the first quantitative approach to estimating diet proportions using FAs and it has already seen widespread use, particularly in studies of marine mammals.

Some recent examples of the use of QFASA include a study on long term diets of New Zealand sea lions that found QFASA estimates to be sensitive to calibration coefficients (discussed in Subsection 2.2.2) but in strong agreement with commercial catches over the same period [16]. In [7], QFASA was used to compare diet estimates for Pacific North West harbour seals with previous estimates obtained via scat analysis. In [9], diets of Atlantic salmon were estimated using QFASA, a first application of the method in fish, and established a set of FAs and calibration coefficients, which reflect the extent to which specific FAs undergo deposition or metabolism, to begin subsequent studies. In [12], an evaluation of QFASA was conducted using controlled feeding experiments of lake trout and found diet estimates to be highly accurate in cases of limited prey diets. In [17], a Bayesian mixture model for diet estimation was proposed which combines fatty acid signatures and stable isotope measurements. The method provides point estimates and probability distributions for individual and population level diet proportions. The R package `fastinR` implements this model as well as provides simulated examples and the analysis of experimental data. This code is separate from that involved in our QFASA package.

2.2.1 Fatty Acids

Fatty acids are the main constituents of lipids. Unlike other nutrients, such as proteins, that are broken down during digestions, some fatty acids are released from ingested lipid molecules (prey fat) but not degraded. Since a relatively limited number of fatty acids can be bio-synthesized by animals [10], it is possible to distinguish between dietary versus non-dietary components [14].

Although some metabolism of fatty acids occurs within the predator, such that the composition of predator tissue will not exactly match that of their prey, fatty acids can be deposited in adipose tissue with little modification and in a predictable way [14]. In practice, calibration coefficients (discussed in Subsection 2.2.2), constants usually derived from feeding trials, are used to account for differential metabolism of individual fatty acids.

2.2.2 Model

The original QFASA model was proposed by [14] and forms the basis of derivative works [22–24], and our QFASA R package. To obtain QFASA diet estimates we require a set of predator FA signatures as well as a database of FA signatures from individual prey of I different species. Each row of the prey database represents a single prey FA signature, which sums to one, corresponding to FAs in the predator signatures. The model attempts to find the proportions of prey species which best explain a predator's FA signature. More formally, we find the diet proportions p_k for each prey species $k = 1 \dots I$ that minimize the distance between a predator's observed FA signature, $\mathbf{Y}$, and $\hat{\mathbf{Y}}$, the result of combining prey FA signatures in said proportions. These estimated proportions are referred to as the QFASA diet estimates. Equation 2.1 shows how the predator FA signature is estimated from diet proportions and prey database averaged by prey species, $\bar{\mathbf{X}}_k$ of dimension n_{fa} by 1 where n_{fa} is the number of FAs being considered in the model.

$$\hat{\mathbf{Y}} = \sum_{k=1}^{I} p_k \bar{\mathbf{X}}_k \tag{2.1}$$

A number of different distance measures have been used in various QFASA studies. No single measure seems to be optimal for all applications. We discuss the main contenders in Section 2.2.3.

Fitting of the QFASA model is performed in the `p.QFASA()` function in the QFASA `R` package. It uses a non-linear optimizer [33] to minimize a statistical distance between actual and estimated predator fatty acid signature subject to the constraints that $\sum p_k = 1$ and $0 \leqslant p_k \leqslant 1$.

Iverson et al. [14] found that more accurate diet estimates could be obtained by introducing factors to account for different levels of fat content in prey as well as different rates of deposition and bio-synthesis of fatty acids in predators.

Prey species with higher fat content will contribute proportionately more per unit intake to a predator's signature than species with a lower fat content. To take this between-species variation into account, [14] makes an adjustment on dietary estimates. The actual estimate of diet proportion derived from prey type k is calculated as:

$$p_k = \frac{p_k^*/f_k}{\sum_k p_k^*/f_k}$$

where f_k is the average fat content and p_k^* the diet proportion estimated by the QFASA model of prey type k.

To account for different rates of deposition and biosynthesis of different fatty acids across predator species, the QFASA model makes use of calibration coefficients, a vector of fatty acid weights. This vector is aligned with the fatty acids included in the model and encapsulates how a particular composition of ingested fatty acids would manifest in the predator's tissues. In the original QFASA study [14], calibration coefficients are derived empirically from feeding experiments on grey and harp seals where the seals are fed a single species for an extended period of time. Simulation studies conducted in [6, 8] concluded that the choice of distance measure affects the robustness of QFASA to errors in calibration coefficients.

Calibration coefficients c_j are derived for each FA type j by comparing the predator FA signature in the experimental studies to their prey signatures. One way to include these calibration coefficients in the compositional distance measures is by replacing the predator's observed proportion of FA j, y_j^* by

$$y_j = \frac{\frac{y_j^*}{c_j}}{\sum_{j=1}^{n_{fa}} \frac{y_j^*}{c_j}}$$

where n_{fa} is the number of FA types being considered in the model. In practice, after incorporating calibration coefficients, a subset of prey FAs (usually comprised of dietary FAs) are used in predator diet modeling. The process of extracting relevant FAs is discussed in [14] and is supported by functions in the QFASA `R` package.

2.2.3 Distance Measures

The `R` function `p.QFASA()` can be directed to optimize based on any one of the following distance measures between actual and estimated predator FA signatures when fitting the QFASA model:

- **Kullback–Liebler (KL) Distance** is the sum of the forward and backward Kullback-Liebler Divergence. The **KL** Divergence is a general measure of how similar two distributions are to one another. It can be interpreted as the relative entropy of Y given $\hat{Y}$, or how well $\hat{Y}$ approximates Y. This is the measure used in the original QFASA model [14] and is defined as

$$KL(\mathbf{Y},\hat{\mathbf{Y}}) = \sum_{j=1}^{n_{fa}} (y_j - \hat{y}_j) \log\left(\frac{y_j}{\hat{y}_j}\right).$$

- **Aitchison (AIT) Distance** is a measure developed specifically for working with compositional data and is defined within the context of Aitchison's geometry of the compositional simplex [1]. Aitchison [1] argues that this is the preferred measure for working with compositional data without zero-valued components, and it defined as:

  $$AIT(\mathbf{Y},\hat{\mathbf{Y}}) = \sum_{j=1}^{n_{fa}} \left(\log\left(\frac{y_j}{g(\mathbf{Y})}\right) - \log\left(\frac{\hat{y}_j}{g(\hat{\mathbf{Y}})}\right)\right)^2,$$

  where $g(.)$ is the geometric mean of the FA signature in this case

  $$g(\mathbf{Y}) = \left(\prod_{i=1}^{n_s} \mathbf{Y}_i\right)^{1/n_s}.$$

  where n_s is number of individual predators for which we require diet estimates. This distance measure is based on Aitchison's *centered-logratio* transformation and satisfies the principles of *scale invariance* and *subcompositional coherence* which are detailed in Section 2.3.

- **Chi-Square (CS) Distance** is a test statistic proposed by [25], and examined further in [23], specifically to handle essential zeros in compositional QFASA data. The intent was to formulate a distance measure that did not require essential zeros to be altered and to satisfy the principles of scale invariance and subcompositional coherence at least approximately. Section 2.4 describes the issue of essential zeros in more detail.

  The distance is based on [11] and was shown empirically to converge to the **AIT** distance when the data is power transformed (i.e $\mathbf{Y}^\gamma$) and re-closed, as γ tends to zero.

  The power of the transformation γ needs to be chosen before calculating the distance measure. Stewart [23] recommended using $\gamma = 1$.

$$CS(\mathbf{Y}, \hat{\mathbf{Y}}, \gamma) = \frac{1}{\gamma}\sqrt{2n_{fa}}\left(\sum_{j=1}^{n_{fa}} \frac{(Z_j - \hat{Z}_j)^2}{c_j}\right)^{\frac{1}{2}}$$

where Z and $\hat{Z}$ are the re-closed power transforms by γ of Y and $\hat{Y}$ respectively, and $c_j = Z_j + \hat{Z}_j$.

As mentioned previously, the QFASA diet estimates are affected by the distance measure used and the most appropriate measure seems to depend on the application specifics. In [6] it was found that the Aitchison distance measure was most robust to errors in the calibration coefficients and that the **KL** distance was most robust to the consumption of prey not represented in the prey database. Of the three distance measures described above, only **CS** distance can be used with zero components since **AIT** and **KL** both involve logarithms. If zero components are considered to be *rounded zeros*, such as those occurring in FAs, adjustments can be made to accommodate **AIT** and **KL** distance measure. However, if the zero components are considered to be *essential zeros*, as is the case for predator diets, these adjustments are not appropriate and only **CS** distance can be used for applications involving measuring distance between diet estimates. The distinction between rounded and essential zeros is described in more detail in Section 2.4.

2.2.4 Standard Error of Estimates

The modeling procedure used in [14] does not take into account within-prey-species variation in FA composition as we use the species means in the QFASA model (Equation 2.1). Also, it does not consider other sources of variability, such as variability in calibration coefficients or prey fat content. To estimate the *standard error* in the QFASA diet estimates, [14] uses a procedure in which the prey species FA signatures are bootstrapped as follows:

1. Generate $\bar{\mathbf{X}}_k^{*b}$ by randomly sampling each prey type with replacement, for $k = 1 \ldots I$.

2. Estimate $\mathbf{p}^{*b}$ using $\bar{\mathbf{X}}_k^{*b}$, in the QFASA model for each $k = 1 \ldots I$.

The standard error for each prey type k is estimated as:

$$SE(p_k) = \sqrt{\frac{\sum_{b=1}^{B} [p_k^{*b} - \bar{p}_k^{*b}]^2}{B-1}}$$

where B is the number of bootstrap estimates.

A similar bootstrap approach was used in [24] to estimate variance arising from within-species FA signatures but incorporates variability due to unknown calibration coefficients and fat content. These more sophisticated methods of error estimation allow the construction of valid confidence intervals and are discussed in [22,24]. The methods devised in [22] are implemented in the QFASA `R` package function `beta.meths.CI()` and are discussed in Section 2.4.

2.2.5 Simulation

Simulation has been used extensively to assess the fit of the QFASA model as well as estimate variance introduced by factors not included in the model. Generally this involves generating *pseudo-predators* by resampling the prey database proportionately according to a selected *true diet*. The QFASA model is subsequently used to estimate the diets of these *pseudo-predators*. The distribution of the estimates can be used to validate new methodologies for QFASA.

Simulations were conducted in [14] to evaluate the sensitivity of QFASA to choice of distance measure and the amount of random noise added to the simulated diet. This noise is meant to represent the proportion of the diet made up of incidental consumption of prey not represented by a prey species in the model. Simulations were used in [23] to compare QFASA distance measures. Pseudo-predator simulations were used in [22, 24] to assess coverage probability of estimated confidence intervals. Simulations were also used in [6,8]

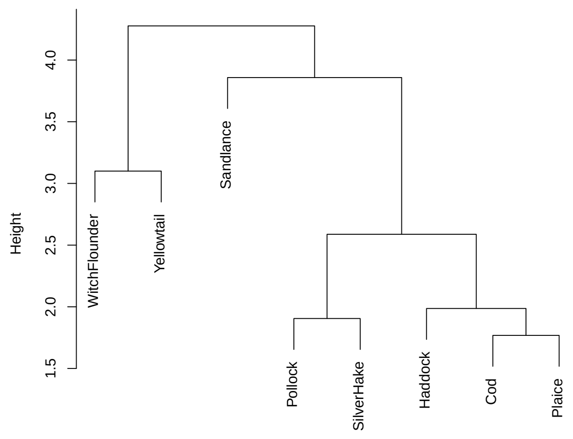


Figure 2.1: Hierarchical cluster analysis on mean FA signature using Aitchison distance

to assess the impact of distance measure selection and the robustness of QFASA to errors in calibration coefficients. The *qfasar* R package [5] encapsulates methods and functions used in this research.

To simulate predators for the assessment of their model, [24] used one of the diets developed by [14] considered to be a difficult modeling case because of the similarity of the FA signatures of the prey species involved. The diet uses 8 species out of 28 in the original prey database. This same diet is used in a simulation study described in Section 4.4.1.

Figures 2.1, 2.2, and 2.3 show these 8 species clustered hierarchically using the Aitchison, Kullback-Liebler, and Chi-Square distance measures respectively. The y axis shows the respective distance measure at which adjacent clusters are agglomerated. The dendrograms were generated by plotting the output of QFASA R package's `prey.cluster()` function. It is evident that resulting cluster structure is affected by choice of distance measure, although the same cluster hierarchy arises for both KL and CS distance measures.

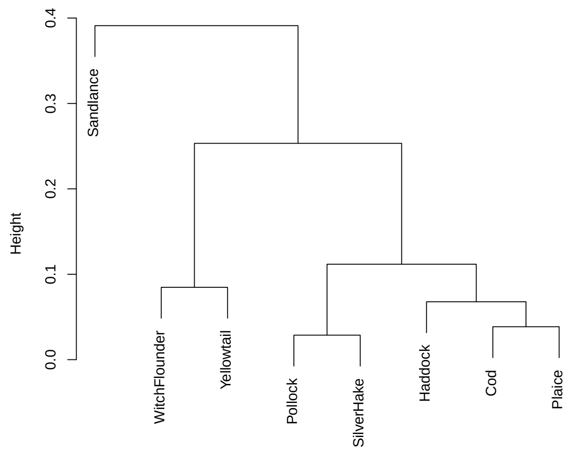


Figure 2.2: Hierarchical cluster analysis on mean FA signature using Kullback-Liebler distance

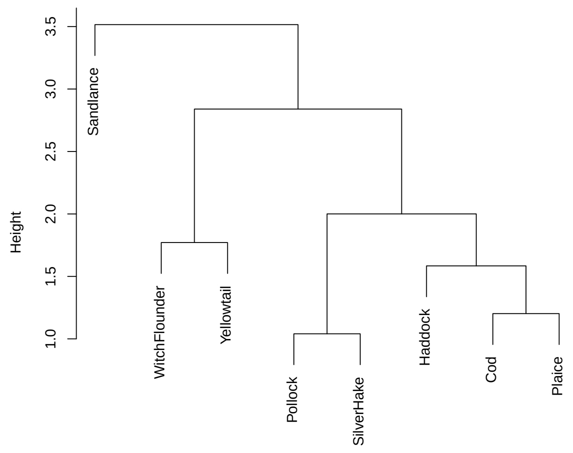


Figure 2.3: Hierarchical cluster analysis on mean FA signature using Chi-Square distance

2.3 Compositional Data

A predator diet represents the relative amount of each prey species consumed by the predator. This type of data is known as *compositional*, where each component represents a proportion of the whole. Compositional data presents issues when subject to standard statistical analysis methods, motivating Aitchison [1] to create a framework, within which most modern methods for compositional analysis are defined. We outline these issues and Aitchison's methods in the following section.

2.3.1 Aitchison Simplex

A compositional observation can be represented as a constant-sum real vector with positive components. This vector's span is a *simplex* on which an alternative sample space $\mathcal{S}^D$ can be defined:

$$\mathcal{S}^D = \left\{ (x_1, x_2, ..., x_D) : \; x_i > 0 \text{ where } i = 1, 2, ..., D \text{ and } \sum_{i=1}^{D} x_i = \kappa \right\} \tag{2.2}$$

It is within this space that [1] defines the *Aitchison Distance* (Section 2.2.3). Herein each composition is considered an equivalency class and any two compositions $\mathbf{Y}$ and $\mathbf{Z}$ are considered equivalent if $\mathbf{Y} = \lambda \mathbf{Z}$ for any $\lambda > 0$.

The vector operation assigning the constant sum representation of compositional observations is called *closure* and is denoted by $\mathcal{C}[.]$:

$$\mathcal{C}[x_1, x_2, ..., x_D] = \left[\frac{x_1}{\sum_{i=1}^{D} x_i}, \frac{x_2}{\sum_{i=1}^{D} x_i}, ..., \frac{x_D}{\sum_{i=1}^{D} x_i} \right] \tag{2.3}$$

A *subcomposition* is defined as any re-closed subset of the components of a composition.

Within this compositional sample space [1] stipulated that valid operators preserve the following properties:

- **Scale Invariance:** A composition carries only relative information, that being given by the ratios between sub-components. Consequently a composition scaled by any positive constant is considered equivalent to the original.
- **Permutation Invariance:** A function is *permutation-invariant* if it yields the same result when the ordering of the components is changed. This is generally the case for unordered compositions.
- **Subcompositional Coherence:** Subcompositional coherence demands that inferences regarding a composition will be consistent with inferences regarding a subcomposition thereof with respect to the ratios of corresponding components. A subcompositionally coherent method would derive the same conclusions on a composition as it would for a sub-composition thereof.

2.3.2 Issues with Compositional Data

As early as 1897, Pearson [19] identified an issue with what he called *spurious correlations*, when indices of the form w_1/w_3 and w_2/w_3 are compared. Although w_1, w_2, and w_3 may be independent, the common denominator correlates the indices. In the context of compositional data, one would expect this issue to be even more prevalent since the closure operator (Equation 2.3) introduces ratios with common elements in the denominator as well as a constant sum constraint on the composition components.

Aitchison [1] enumerated the following issues which invalidate the application of standard multivariate statistical methods to compositional data:

- **Negative Bias:** In a D-part composition subject to the unit sum constraint, since the covariance of any component with the sum of all components is zero and the variance of the chosen component is positive, at least one of the pairwise covariance terms on the left-hand-side must be negative. This means that in each row of the standard covariance matrix, at least one entry must be negative and hence correlations are not free to range over the usual $[-1, 1]$ interval. That is,

$$\begin{aligned} x_1 + \ldots + x_D &= 1 \\ \Rightarrow Cov(x_1, x_1 + \ldots + x_D) &= 0 \\ \Rightarrow Cov(x_1, x_2) + Cov(x_1, x_3) + \ldots + Cov(x_1, x_D) &= -Var(x_1). \end{aligned}$$

- **Subcompositional Issues:** Since knowledge of a composition allows us to create any sub-composition thereof, one might expect to find some relationship between the standard covariance matrix of a sub-composition and that of the full composition. It is easy to confirm empirically that no such relationship exists: correlations may vary substantially and there may be no apparent pattern.
- **Basis Issues:** Since the construction of a composition from a basis is a constraining operation similar to the construction of a subcomposition, we might also expect to see some relationship between covariance matrices of bases and compositions. Just as the case for subcompositions, no patterns are evident.
- **Singular Covariance Structures:** If the covariance matrix $Cov(\mathbf{X})$ is not *positive-definite* then there exists some $\mathbf{a} \in \mathbb{R}^D$ such that $Cov(\mathbf{X})\mathbf{a} = \mathbf{0}$. Hence

$$\begin{aligned} 0 &= \mathbf{a}' Cov(\mathbf{X})\mathbf{a} \\ &= \sum_{ij} a_j Cov(X_i, X_j) a_i \\ &= \sum_i Var(a_i X_i) + 2 \sum_{i<j} Cov(a_i X_i, a_j X_j) \\ &= Var(\sum_i a_i X_i). \end{aligned}$$

So, in general, there is some linear combination of the components of $\mathbf{X}$ which has zero variance and hence is constant. By the constant-sum constraint on the *Aitchison*

Simplex, this condition always holds for $\mathbf{a} = [1, 1, ..., 1]$ and so the covariance matrix is not invertible.

2.3.3 Parametric Models for Compositional Data

Aitchison [1] defined several parametric models for the analysis of compositional data with strictly positive components, including the *Dirichlet* distribution, the *additive logistic normal* distribution, and the *multiplicative logistic normal* distribution.

Every *Dirichlet* composition can be visualized as the composition formed from a basis of independent, equally scaled gamma-distributed components. This implies a very strong independence structure, unlikely to be of use in describing compositions whose components have even weak forms of dependence [1].

The *additive logistic normal distribution* is based on Aitchison's *additive logistic transformation* and its inverse, the *additive log-ratio transformation.* A composition $\mathbf{p}$ is said to have an *additive logistic normal* distribution $\mathscr{L}^{D-1}(\mu, \Sigma)$ when the additive log-ratio transformed data $\mathbf{w}$, is multivariate normal:

$$\mathbf{w} = \left[\log\left(\frac{p_k}{p_D}\right), \ldots, \log\left(\frac{p_{D-1}}{p_D}\right)\right] \sim \mathcal{N}_{D-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

The inverse logistic transformation is defined as:

$$p_k = \frac{e^{w_k}}{(e^{w_1} + e^{w_2} + \ldots + 1)}, \qquad k = 1, \ldots, D-1$$
$$p_D = 1 - w_1 - ... - w_d = \frac{1}{(e^{w_1} + e^{w_2} + \ldots + 1)}.$$

This pair of transformations allows one to move back and forth between the compositional simplex and unrestricted real space. This opens the path to transforming compositional data into a space where standard parametric distributions and multivariate analysis methods can be used and the outcomes converted back to the original sample space.

In predator diet estimation where the number of diet components is typically large relative to sample sizes, multivariate inference procedures for a vector of parameters are not practical and univariate methods based on marginal distributions may be more desirable. Because of an amalgamation property that allows marginal distributions to be easily derived, [24] uses Aitchison's *multiplicative logistic normal distribution* as the basis for building mixture models for compositional data with essential zeros instead of the *additive logistic normal distribution.*

Similarly to the *additive logistic normal* distribution, the *multiplicative logistic normal* distribution is based on the *multiplicative logistic transformation* and its inverse, the *multiplicative log-ratio transformation.* A composition $\mathbf{p}$ is said to have an *multiplicative logistic normal* distribution $\mathscr{M}^{D-1}(\mu, \Sigma)$ when the multiplicative log-ratio transformed data $\mathbf{w}$, is multivariate normal:

$$\mathbf{w} = \left[\log\left(\frac{p_1}{1-p_1}\right), \ldots, \log\left(\frac{p_{D-1}}{1-p_1-\ldots-p_{D-1}}\right)\right] \sim \mathcal{N}_{D-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

The inverse logistic transformation is defined as:

$$\begin{aligned} p_k &= \frac{e^{w_k}}{[(1+e^{w_1})\ldots(1+e^{w_k})]}, \qquad k = 1, \ldots, D-1 \\ p_D &= \frac{1}{[(1+e^{w_1})\ldots(1+e^{w_k})]} \end{aligned}$$

The $\mathscr{M}^{D-1}$ distribution can be extended to include a shape parameter that allows for skewness in the transformed data. A composition $\mathbf{p}$ is said to have *multiplicative logistic skew-normal* distribution $S_k\mathscr{M}^{D-1}(\mu, \Sigma, \alpha)$ when:

$$w_k = \log\left(\frac{p_k}{1 - p_1 - \ldots - p_k}\right) \sim S\mathcal{N}(\mu,\, \sigma,\, \alpha), \qquad k = 1, \ldots, D-1$$

where α is an $(D-1)$-dimension shape parameter [3].

2.4 Modelling Compositional Data with Zeros

An *essential zero* in compositional data is a zero component which is not caused by rounding or some other difficulty in measurement. Essential zeros are defined in [15] as the absolute absence of the part in the observation. Dealing with essential zeros is a fundamentally different problem than handling *rounded zeros*, or *below-detection level zeros* which [15] define as *present but below detection levels* and for which multiple substitution and imputation strategies exist. In QFASA, zeros in FA signatures are generally treated as rounded zeros while zeros in the diet estimates are considered essential zeros as they represent the absence of a species in the diet.

A key feature of compositional data is that the ratios of components contain all pertinent information about the composition. Essential zeros complicate this feature in that they contain no information about the other components of the composition, an observation containing an essential zero being at the boundary of the simplex and technically a composition of smaller dimensions [4].

The initial QFASA model [14] yields point estimates of diet and these usually contain essential zeros. Inference requires modeling these diet estimates but the models outlined in Section 2.3 cannot be used due to essential zeros and the log transformations involved in them. A method is used in [24] for handling essential zeros using a parametric mixture model. The strategy divides the compositions into sub-populations according to where zeros occur in the components and specifies the distribution of the non-zero components in each population conditional on the zero elements.

More specifically, consider the general case of $\mathbf{p}$ being a composition with possible zero components. Let $\mathbf{V}$ denote the vector of indices indexing the non-zero components of $\mathbf{p}$ and let $\mathbf{p}_V$ be the vector of these non-zero components. Supposing that $f_V(\mathbf{p}_V)$ is the density of $\mathbf{p}_V$, then $f_V(\mathbf{p}_V)$ may be any density suitable for compositional data. In [24] the *multiplicative logistic normal* distribution was used.

To model $\mathbf{p}$ it is assumed that there are separate populations for every possible value of $\mathbf{V}$. Let $\theta_{\mathbf{v}} = P[\mathbf{V} = \mathbf{v}]$ be the marginal probability that an observation comes from population with non-zero components indexed by $\mathbf{V}$, where $\Sigma_{b=1}^{B}\theta_{\mathbf{v}_b} = 1$ where B denotes the number of populations.

The joint density of $\mathbf{p}$ and $\mathbf{V}_b$ can then be written:

$$f_{\mathbf{p},\mathbf{v}_b}(\mathbf{p},\mathbf{v}_b) = f_{\mathbf{p}|\mathbf{v}_b}(\mathbf{p}|\mathbf{v}_b)\,\theta_{\mathbf{v}_b} = f_{\mathbf{v}_b}(\mathbf{p}_{\mathbf{v}_b})\,\theta_{\mathbf{v}_b}.$$

Then

$$f_{\mathbf{p}}(\mathbf{p}) = \Sigma_{b=1}^{B} f_{\mathbf{p},\mathbf{v}_b}(\mathbf{p},\mathbf{v}_b) = f_{\mathbf{v}_p}(\mathbf{p}_{\mathbf{v}_p})\,\theta_{\mathbf{v}_p}, \tag{2.4}$$

where $\mathbf{v}_p$ is the only $\mathbf{v}_b$ that corresponds to the non-zero components of $\mathbf{p}$ and such that $f_{\mathbf{p},\mathbf{v}_b}(\mathbf{p},\mathbf{v}_b) \neq 0$. Multivariate inference procedures such as confidence regions are usually not practical or interpretable in QFASA, where dimensions are large and sample sizes relatively small, so univariate procedures based on the marginal distributions are used instead. The marginal distribution of component k of $\mathbf{p}$ can be derived by integrating $f_{\mathbf{p},\mathbf{v}_b}(\mathbf{p},\mathbf{v}_b)$ over p_j, $j \neq k$:

$$f_k(p_k,\mathbf{v}_b) = \begin{cases} \theta_{\mathbf{v}_b} & \text{if } p_k = 0,\ k \notin \mathbf{v}_b \\ \theta_{\mathbf{v}_b} \int \ldots \int f_{\mathbf{v}_b}(\mathbf{p}_{\mathbf{v}_b}) d\mathbf{p}_{-k} & \text{if } 0 < p_k < 1,\ k \in \mathbf{v}_b \\ 0, & \text{otherwise.} \end{cases}$$

By reordering the composition $\mathbf{p}$ so that component k is in position 1 and using the amalgamation property in [1], if $k \in \mathbf{v}_b$ then:

$$(p_k, 1 - p_k) \sim \mathscr{M}(\mu_k^{\mathbf{v}_b},\, \sigma_k^{2\mathbf{v}_b})$$

and

$$\begin{aligned} f_k(p_k) &= \Sigma_{b=1}^{B} f_k(p_k, \mathbf{v}_b) \\ &= \Sigma_{b:k\notin\mathbf{v}_b}^{B} f_k(p_k, \mathbf{v}_b) + \Sigma_{b:k\in\mathbf{v}_b}^{B} f_k(p_k, \mathbf{v}_b) \\ &= \begin{cases} \Sigma_{b:k\notin\mathbf{v}_b}\theta_{\mathbf{v}_b} & \text{if } p_k = 0 \\ \Sigma_{b:k\notin\mathbf{v}_b}\theta_{\mathbf{v}_b}\,\mathscr{M}(\mu_k^{\mathbf{v}_b},\, \sigma_k^{2\mathbf{v}_b}) & \text{if } 0 < p_k < 1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Making the simplifying assumption $\mu_k^{\mathbf{v}_b} = \mu_k$ and $\sigma_k^{2\mathbf{v}_b} = \sigma_k^2\, \forall k$ such that $k \in \mathbf{v}_b$, gives a marginal mixture distribution:

$$f_k(p_k) = \begin{cases} \Sigma_{b:k\notin\mathbf{v}_b}\theta_{\mathbf{v}_b} & \text{if } p_k = 0 \\ \Sigma_{b:k\in\mathbf{v}_b}\theta_{\mathbf{v}_b}\,\mathscr{M}(\mu_k,\, \sigma_k^2) & \text{if } 0 < p_k < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Further, if we let $\theta_k = \Sigma_{b:k\notin\mathbf{v}_b}\theta_{\mathbf{v}_b}$ then since $\Sigma_{b:k\notin\mathbf{v}_b}\theta_{\mathbf{v}_b} + \Sigma_{b:k\in\mathbf{v}_b}\theta_{\mathbf{v}_b} = 1$, we can write the above distribution as:

$$f_k(p_k) = \begin{cases} \theta_k & \text{if } p_k = 0 \\ (1 - \theta_k)\mathscr{M}(\mu_k,\, \sigma_k^2) & \text{if } 0 < p_k < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

If p_k has the simplified density in Equation 2.5, [24] uses the notation below:

$$p_k \sim \mathrm{SMixM}(\theta_k,\, \mu_k,\, \sigma_k^2).$$

In this model, we are in effect assuming that $p_k = 0$ with probability θ_k and that $\log[p_k/(1-p_k)] \sim \mathcal{N}(\mu_k,\, \sigma_k^2)$ for $p_k > 0$. This is called the *zero-inflated logistic normal* (ZIL) distribution.

The parameters of the ZIL distribution are estimated via maximum likelihood (MLE). Given a random sample $p_{1k}, \ldots, p_{n'k}$ from the distribution, where n' denotes the number of non-zero proportions in the sample, the MLEs are

$$\begin{aligned}
\hat{\theta_k} &= (n-n')/n \\
\hat{\mu_k} &= 1/n'\, \Sigma_{i=1}^{n'} \log[p'_{ik}/(1-p'_{ik})] \\
\hat{\sigma_k^2} &= 1/n'\, \Sigma_{i=1}^{n'} [\log[p'_{ik}/(1-p'_{ik})] - \hat{\mu_k}]^2.
\end{aligned}$$

The skew extension, to the above, the *zero-inflated logistic skew normal* (ZILS) distribution, is as follows and requires numerical methods for parameter estimation:

$$f_k(p_k) = \begin{cases} \theta_k & \text{if } p_k = 0 \\ (1-\theta_k) S_k \mathscr{M}(\mu_k,\, \sigma_k^2,\, \alpha_k) & \text{if } 0 < p_k < 1 \\ 0 & \text{otherwise.} \end{cases}$$

In [22] the use of the *zero-inflated beta* (ZIB) distribution [18] is examined as an alternative to the zero-inflated distributions used in [24]. The advantage of the ZIB distribution method is that it does not require a transformation of the original data.

The ZIB distribution defined in [18] is:

$$f_B(p) = \begin{cases} \theta & \text{if } p = 0 \\ (1-\theta)\mathcal{B}(p;\mu,\,\phi) & \text{if } 0 < p_k < 1, \end{cases} \tag{2.6}$$

where $\mathcal{B}$ denotes the beta distribution with density function:

$$\mathcal{B}(p;\mu,\,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} p^{\mu\phi-1}(1-p)^{(1-\mu)\phi-1}$$

and $0 < \mu < 1$ and $\phi > 0$. Also $E[p \| p \in (0,1)] = \mu$ and $Var[p \| p \in (0,1)] = \mu(1-\mu)/(\phi + 1)$.

As is the case for the other zero-inflated distributions, it can be shown that the MLE of θ is the proportion of zeros in the sample, i.e. $\hat{\theta} = (n - n')/n$. The estimation of the parameters μ and ϕ is discussed in detail in [18]. The MLEs can be obtained using the *gamlss* package in *R*.

Subsequent recent work in [4] proposes a mixture model for handling essential zeros based on the additive logistic transform that offers an alternative framework for multivariate inference, albeit with certain restrictions. This work has not been extended to QFASA.

In the context of QFASA, the mean of these zero-inflated distributions is of particular interest since it is typically close to that of the true diet [24]. In the case of the ZIB distribution, the mean is given by $\beta = (1-\theta)\mu$ and is in the same space as the sample data. In contrast, the zero-inflated logistic distributions involve a transformation of the original data, so the resulting means must be reverted to the original sample space for inference.

A simulation study was conducted in [22] to compare estimator performance for the ZIL, ZILS, and ZIB distributions. The ZIL, ZILS, and ZIB distributions were fit to diet estimates of simulated pseudo predators. Samples of various sizes were generated from each distribution using distribution parameters yielding approximately equal mean and variance for each distribution. Comparing results indicated a similar bias and variance from each of the distributions. Since all the distributions treat the proportion of zeros in the same manner, only the fit of the non-zero proportions were compared. Based on a comparison of deviance, the ZIB and ZILS models outperformed the ZIL model.

The bootstrap methods used in [24] to derive confidence intervals for true diets based on ZIL and ZILS were extended in [22] to the ZIB distribution. A *confidence interval* (CI) is an interval in parameter space with an associated *confidence level* that quantifies the confidence that the true parameter value (in our case, the true diet) lies within the interval. The `beta.meths.CI()` function in the QFASA `R` package implements this estimation process. An outline of the methods is covered in Chapter 3.

Chapter 3

Inference from QFASA

3.1 Introduction

The QFASA R package contains code to compute the confidence intervals for the true diet, as well as testing for changes in diets. These aspects are discussed in detail in the following sections.

3.2 Confidence Intervals

The confidence interval methods developed in [24], and subsequently in [22], are intended for the case where the sample size of predators is small relative to the dimension of QFASA diet estimates. The variability due to unknown biological factors such as calibration coefficients and fat content is handled by using bootstrapping techniques.

In [22, 24] confidence intervals were constructed by bootstrapping and inverting a hypothesis test. In a hypothesis test we postulate that a population parameter is a certain value. In the case of QFASA, this parameter is a prey diet proportion. This postulated value, the *null hypothesis*, defines a probability distribution, the *null distribution*, which is the distribution we expect for sample statistics from the population if the null is true. The p-value of a sample statistic is the probability, under the null distribution, of encountering a value as or more extreme than the observed sample statistic. If this probability is lower than a predetermined *significance level*, the null hypothesis is rejected by the test. The non-rejection region of a two-sided hypothesis test with significance level α can be interpreted as a confidence interval with confidence level $(1 - \alpha)$.

For the zero-inflated distributions used in the QFASA model to handle essential zeros, it is difficult to derive the non-rejection region analytically so bootstrap estimation was used in [22, 24]. Bootstrap samples $1 \ldots B$ are generated from a specific null distribution via parametric bootstrap and the corresponding p-value calculated as the proportion of bootstrap samples which exceed the observed test statistic. This process is repeated iteratively to find the parameter values which bound the non-rejection region.

In the case of the QFASA R package, we first use a non-parametric bootstrap (section 3.2.1) to estimate nuisance parameters in the ZIB distribution, followed by a parametric bootstrap in which estimates are generated from the ZIB null distribution (section 3.2.2).

3.2.1 Non-parametric Bootstrap

The bootstrapping p-value algorithm consists of an initial non-parametric bootstrap in which the nuisance parameters ϕ_k and θ_k in Equation 2.6 are estimated. More specifically, these are estimated by generating many pseudo-predators, estimating their diet, and estimating these parameters for the bootstrap sample. While the bootstrapping could be repeated a number of times to account for variability in the nuisance parameters due to variance in predators, the process is computationally intensive as is and [24] anticipates that the results would not differ significantly.

3.2.2 Parametric Bootstrap

Ignoring the variability due to the prey, we assume that $\mathbf{p}(\mathbf{Y},\, \boldsymbol{\mu_X}) \approx \mathbf{p}(\mathbf{Y},\, \bar{\mathbf{X}})$ where $\boldsymbol{\mu_X} = E[\bar{\mathbf{X}}]$ so our diet estimates may be considered independent. Based on the assumption that our estimates $p_{k,1}(\mathbf{Y}_1, \boldsymbol{\mu_X}), \ldots, p_{k,n_s}(\mathbf{Y}_{n_s}, \boldsymbol{\mu_X})$ have a ZIB distribution (equation 2.6), we derive confidence intervals for estimates of each species proportion k by inverting the hypothesis test

$$H_0 : \beta_k = \beta_{k0}$$
$$H_1 : \beta_k \neq \beta_{k0}$$

where $\beta_k = (1-\theta_k)\mu_k$ is the mean of the ZIB distribution, which we assume to be close to the true diet proportion. A subsequent adjustment is made to bootstrap CIs to compensate for the difference between β_k and the true diet proportion, a process described in section 3.2.3. The bootstrap procedure is as follows:

1. Given estimates of θ and ϕ from the non-parametric bootstrap, θ_{boot} and ϕ_{boot} respectively, pick values of β_{k0} over the interval [0, 1] and for each value of β_{k0}, calculate

$$\mu_{k0} = \frac{\beta_{k0}}{(1-\theta_{boot})}.$$

   (a) Generate B bootstrap samples of the test statistic and for each bootstrap sample $b = 1 \ldots B$ calculate T_b^* as follows:

      i. Generate $p^*_{k,1}, \ldots, p^*_{k,n^*}$ from distribution $\mathcal{B}(p; \mu_{k0},\, \phi_{boot})$ and calculate MLE $\hat{\mu}^*_{k0}$.

      ii. Calculate $\hat{\beta}^*_{k0} = (1-\theta_{boot})\hat{\mu}^*_{k0}$

      iii. Calculate $T_b^* = ||\hat{\beta}^*_{k0} - \beta_{k0}||$

   (b) Calculate the bootstrap *p-value* corresponding to β_{k0} as the proportion of bootstrap samples of T_b^* which are greater than $T = \|\hat{\beta}_k - \beta_{k0}\|$, where $\hat{\beta}_k$ is calculated from the sample proportions $p_{k,1}, \ldots, p_{k,n}$:

$$p^*(\beta_{k0}) = \frac{1}{B} \sum_{b=1}^{B} I_{\{T_b^* \geqslant T\}}.$$

2. Numerically solve for the roots of $p^*(\beta_{k0}) - \alpha = 0$, the upper and lower limits of $(1-\alpha)\,100\%$ confidence interval, say β_{k0}^{*L} and β_{k0}^{*U}.

In [22, 24], the roots in Step 2 are solved by a custom bisection process. We have subsequently replaced this by a built-in R bisection method as part of the improvements made in the QFASA R package implementation. A detailed description of the change and simulation results comparing the two methods are given in Section 4.4.1.

3.2.3 Bias

If we generate many pseudo-predators with true diet $\boldsymbol{\pi}$ and estimate each of their diets using QFASA, the average of these diet estimates, say $\boldsymbol{\beta}$, is not $\boldsymbol{\pi}$. We call the difference between $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ the bias of our estimator. Stewart et al. [24] cites possible sources of this bias to include the similarity between the FA signatures of certain species in the diet, the choice of FA subsets in the QFASA model, and the estimated FA calibration coefficients and prey fat content.

The confidence intervals derived in [24] are shifted by an estimate of the bias. The algorithm involves generating pseudo-predators from each estimate of diet for the n_s predators in our sample. From each of the n_s sets of pseudo-predators, we calculate the value of $\hat{\beta}_k$ for $k = 1, \ldots, I$. The difference between the true diet and estimated parameter values is computed for each predator and the n_s estimates of bias are averaged, resulting in an estimate of bias for each prey species.

3.3 Testing for Differences in Compositional Data

Detecting changes in predator diets is an important ecological problem. QFASA establishes a link between FA signatures of predators and their diets. Consequently, changes in these FA signatures can be used to infer changes in diet. [23, 25] devised a method to test for a difference between compositional data with zeros which can be applied directly to predator FA signatures or diet estimates. In this case, p-values are obtained by multivariate permutation tests with a test statistic based on CS distance. These methods are implemented in the following functions in our QFASA R package:

- `testfordiff()`
- `testfordiff.ind.boot.fun()`
- `create.d.mat()`
- `chisq.CA()`
- `testfordiff.ind.pval`

Chapter 4

QFASA R Package

4.1 Overview

In this chapter we describe the implementation of our R package encapsulating and extending QFASA functions from [22–24]. We detail additions to the original source code required to create the package and satisfy the Comprehensive R Archive Network (CRAN) submission process. We go into detail regarding optimizations added to leverage third-party root finding tools and multicore processors. We also review a complementary R package, *qfasar*, which leverages the same underlying theories but with a more specific focus on conducting predator simulations and diagnosing goodness-of-fit of diet estimates.

4.2 The R Language

R [21] is a software environment for statistical computing and graphics. It first appeared in the early 1990s as a free open source implementation of the S language which was developed in the 1970's at Bell Laboratories by John Chambers and colleagues. Today it boasts a diverse and highly active community of core language maintainers and package

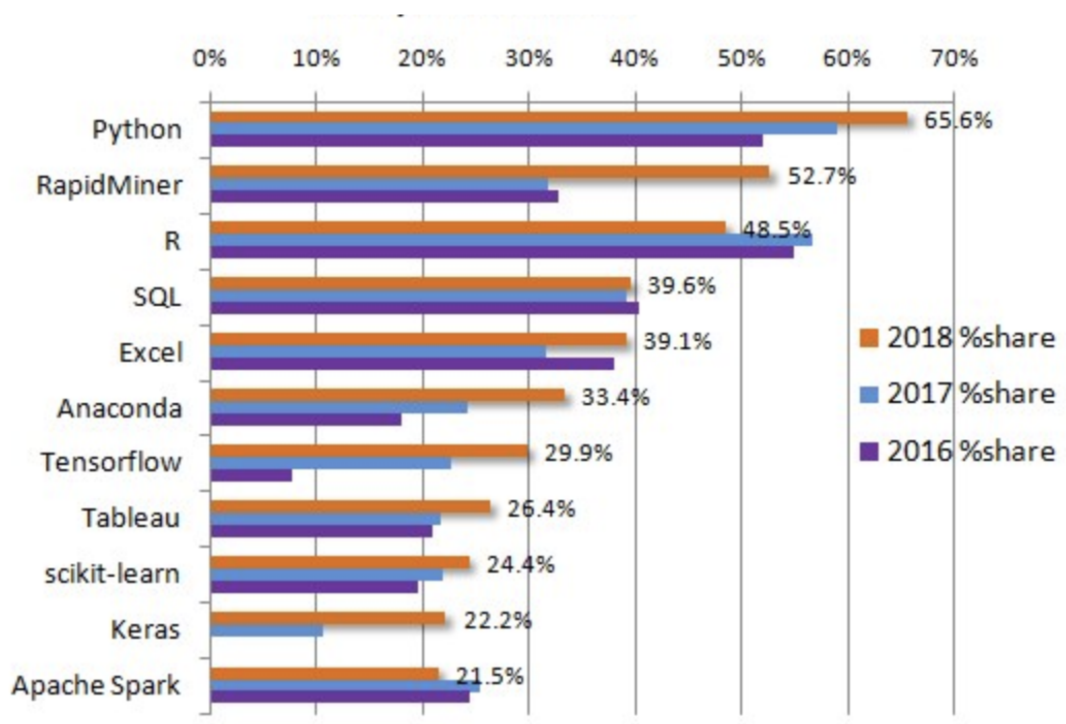


Figure 4.1: KDNuggets Analytics, Data Science, Machine Learning Software Poll 2016-2018

developers, and is one of the primary tools used today in the fields of data science and statistical analysis. The R user and development community is highly active, with new language features, improvements, and extensions being continually added.

Figure 4.1 shows results of a 2016 survey [20] by KDNuggets, a datascience community portal, in which R is a top language tool favoured by self proclaimed data scientists. However, it is interesting to note that adoption of the Python language in this community is growing faster and considered at this point to have surpassed R. It is unclear whether this is due to users switching tools or more people from software engineering, where Python is prevalent, calling themselves Data Scientists and Analysts [20].

The capabilities of R are extended through user-created packages. These packages are developed primarily in R, and sometimes in Java, C, C++, and Fortran. The R packaging system is also used to organise and distribute research data in a systematic way for sharing and archiving.

A core set of packages is included with the base installation of R and more than 11,000 additional packages are available from CRAN, a network of file and web servers around the world that store identical, up-to-date, versions of base R and packages. CRAN follows the CTAN (Comprehensive TeX Archive Network) distibution model. CTAN is the authoritative place where TEX related material and software can be found for download.

The R manual [27] for creating a new package is a 180+ page document. It is highly prescriptive with respect to how code is to be structured and arranged into directories as well as minimal standards for documenting functions and data included in the new package. The goal of the specification is to create high quality additions that work as intended in new environments and uphold the quality of the platform.

4.3 R Packages

An R package is not required to distribute R code. R code is most commonly shared as individual scripts. When code or data are to be shared to a wide audience which will typically use the software unsupported, it becomes worthwhile to spend the effort to create a package which encapsulates all code, documentation, examples, and data sets required to use the new functions or methods. A package can also include information regarding what other R packages it uses and these will be automatically installed by the package manager when the package is installed.

Distributing R code as a package is particularly useful where the target audience is non-technical and will easily abandon the process of trying to use the new code if there are any errors which occur. This was a key motivating factor for the creation of the *QFASA* package. The *QFASA* package allows supporting software to be distributed and upgraded in a modular and repeatable way with a low technical burden on the user.

4.3.1 CRAN

Although R packages can be distributed independent of CRAN, CRAN provides a consistent and easy to use interface for distribution and installation. The submission process is stringent but the requirements generally promote good practice in documentation, testing, and software engineering. We describe the submission process in more detail in Section 4.5.

When packages are accepted to CRAN they are automatically propagated to all mirror sites. How quickly this happens depends on the refresh schedules of the individual mirrors. There is a well documented [26] and easy to follow process for setting up a CRAN mirror site. Dalhousie University in Halifax is currently a CRAN mirror.

Also, packages are versioned and separate versions maintained on CRAN. Users can select specific versions of the package from the archive or seamlessly upgrade to the latest version available.

4.3.2 Package Structure

Devtools [29] is an R package created to abstract many of the lower level details of creating an R package and to encapsulate many best practices. *Devtools* can be used directly via the R shell but integrates well with *RStudio*, a graphical user interface to the R environment. That said, regardless of whether *Devtools* or *RStudio* is used, the underlying process of creating a package is the same [28]. The *Devtools* package was used extensively in the creation of the *QFASA* package.

Package source code can be zipped up and distributed as a *bundled package*. However, if you want to distribute your package to an R user who does not have the development tools installed required to compile the package, you need to compile the package source code into a *binary package*. A binary package is required for distribution via CRAN. Like a bundled package, a binary package is a single compressed file, However the contents are different from a source package:

- R source files are compiled into three files that store the parsed functions in an efficient format. This has the result as loading the source code into an R environment and then saving the functions with `save()`.
- A `meta/` directory contains a number of *Rds* files which contain cached metadata about the package. These files help to make package loading faster by storing the results of costly computations.
- An `html/` directory contains the files needed for HTML help.
- Native source code in `src/`, if any, is compiled into 32-bit and 64-bit libraries and stored in `i386/` and `x64/` respectively.

The contents of the `meta/`, `html/`, `i386/`, and `x64/` directories are generated by the package development toolchain. Binary packages are platform specific and must generally be created separately for each platform on which they are to be used.

Figure 4.2 provides code and output showing how a template package directory structure can be created using `devtools::create()`. This creates template files and directories:

- **DESCRIPTION:** package metadata, including authors, licensing, and dependencies.
- **NAMESPACE:** defines the R namespace within which the package functions will be defined in order to avoid conflicts with other package and the R base functions.
- **R/:** R source files.
- **devtools.Rproj:** a text file that integrates the created structure with *RStudio*, making it aware of the project structure and the artifacts therein.

```
devtools::create("~/tmp/QFASA")

## Package: QFASA
## Title: What the Package Does (one line, title case)
## Version: 0.0.0.9000
## Authors@R: person("First", "Last", email = "first.last@example.com", role = c("aut", "cre"))
## Description: What the package does (one paragraph).
## Depends: R (>= 3.5.1)
## License: What license is it under?
## Encoding: UTF-8
## LazyData: true
##  Writing 'QFASA.Rproj'
##  Adding '.Rproj.user' to '.gitignore'
##  Adding '^QFASA\\.Rproj$', '^\\.Rproj\\.user$' to '.Rbuildignore'

dir("~/tmp/QFASA")

## [1] "DESCRIPTION" "NAMESPACE"   "QFASA.Rproj" "R"
```

Figure 4.2: Creating a package template with *Devtools*

4.3.3 R Code

There is no shortage of recommendations on conventions for R coding style, variable naming, function naming etc. The R package manual is non-prescriptive in this sense and any working code can be packaged. However it is probably a good idea to adopt at least some convention as these usually help in the long run and are usually the result of past mistakes made by other developers that you can hopefully avoid.

Also there are tools available to make writing well formatted and structured code easier. The following packages have functions which can be used for this purpose:

- **Formatr:** will clean up poorly formatted code itself [32].
- **Lintr:** will produce warnings and recommendations where code does not fit a defined style but leaves actual fixing of issues to the programmer [13].

Style components include variable naming, indentation, spacing, line length, code comment, and positioning of braces to demarcate code blocks.

In general, packages should avoid adjusting global options and parameters that affect the environment as a whole. This includes things like `options(digits=n)` to set the number of significant digits to display when printing numeric values, and `par(mfrow = c(2,2))` to set the layout for plots. If set, these options persist for the session. If necessary, packages should always restore options and parameters to their original values when done.

If packages introduce new environment options, they should be prefixed with the package name to prevent conflict with existing options.

Changing the working directory is considered poor form and should be avoided or changed back as soon as possible. Changing the working directory using `setwd()` within a package function can cause unexpected behaviour for the user.

Where it is necessary to initialize a package before use, such as setting up a connection to an external system, packages can use the `onLoad()` and `onAttach()` function hooks. Also, packages can use `onUnLoad()` to clean up and release any resources that are no longer needed.

4.3.4 Metadata

The DESCRIPTION file stores important metadata about your package. This includes:

- **Title and Description:** the title is a one-line description of the package and the Description field a more detailed overview.
- **Author:** identifies contributors to the package, their email addresses and their different roles. The main roles are creator or maintainer, author, contributors, and copyright holders. The email address of the creator role will be used by CRAN to contact you about your package.

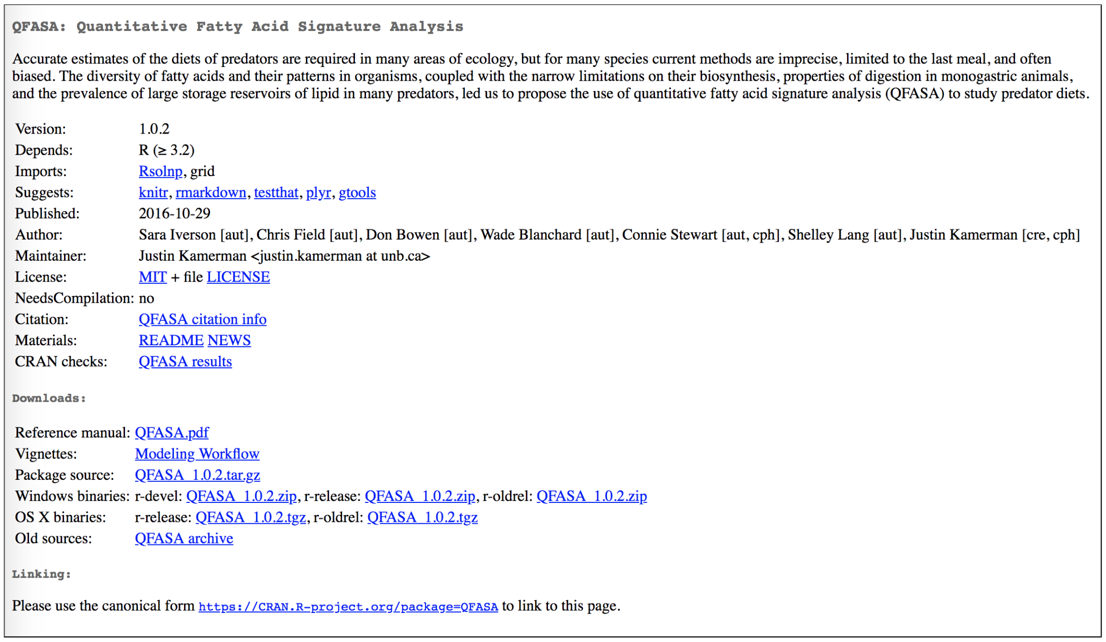
- **License:** If you want to release your package on CRAN, you must choose a standard software license. Otherwise it is difficult for CRAN to determine whether or not it is legal to distribute your package. The most common licenses are MIT, GPL-2, and Creative Commons:
  - **MIT** is a simple and permissive template license, stipulating only that redistribution of the software in any form also include the given license. This is the license under which the *QFASA* package has been released.
  - **GPL-2** is a copy-left license in that it requires that any derived work also be distributed under the GPL-2 license.
  - **Creative Commons** relinquishes any rights you have to the package and effectively puts it in the public domain.
- **Dependencies:** the packages necessary for your package to work: Packages listed in *Imports* field must be present for your package to work. Any time your package is installed those packages, if not already present, will be installed on your computer. The *QFASA* package requires the `Rsolnp`, `boot`, and `grid` packages. Packages listed in the *Suggests* field are used by your package but it does not require them to work. These are not automatically installed with your package.
- **Version:** required so that users can identify which release of your package they are using and whether they require an upgrade to have access to the latest features and/or fixes. Also, within the R package management system, packages depend on other packages. Being able to specify exactly which version of a dependency a package requires is essential to maintaining a stable analysis environment.

The information contained in the DESCRIPTION file is used by CRAN to publish the package on their Web archive. A DESCRIPTION file for the *QFASA* R package is shown in Figure 4.3 and the corresponding CRAN page in Figure 4.4.

```
Package: MyPackage
Title: My R Package
Version: 1.0.2
Authors@R: c(person("Connie", "Stewart", email = ...)

Description: Accurate estimates of the diets of predators are required
in many areas of ecology, ...
Depends:
R (>= 3.2)
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Imports: Rsolnp, grid, boot
RoxygenNote: 5.0.1
VignetteBuilder: rmarkdown, knitr
```

Figure 4.3: DESCRIPTION file for *QFASA* package



QFASA: Quantitative Fatty Acid Signature Analysis

Accurate estimates of the diets of predators are required in many areas of ecology, but for many species current methods are imprecise, limited to the last meal, and often biased. The diversity of fatty acids and their patterns in organisms, coupled with the narrow limitations on their biosynthesis, properties of digestion in monogastric animals, and the prevalence of large storage reservoirs of lipid in many predators, led us to propose the use of quantitative fatty acid signature analysis (QFASA) to study predator diets.

Version:	1.0.2
Depends:	R (≥ 3.2)
Imports:	Rsolnp, grid
Suggests:	knitr, rmarkdown, testthat, plyr, gtools
Published:	2016-10-29
Author:	Sara Iverson [aut], Chris Field [aut], Don Bowen [aut], Wade Blanchard [aut], Connie Stewart [aut, cph], Shelley Lang [aut], Justin Kamerman [cre, cph]
Maintainer:	Justin Kamerman <justin.kamerman at unb.ca>
License:	MIT + file LICENSE
NeedsCompilation:	no
Citation:	QFASA citation info
Materials:	README NEWS
CRAN checks:	QFASA results

Downloads:

Reference manual:	QFASA.pdf
Vignettes:	Modeling Workflow
Package source:	QFASA_1.0.2.tar.gz
Windows binaries:	r-devel: QFASA_1.0.2.zip, r-release: QFASA_1.0.2.zip, r-oldrel: QFASA_1.0.2.zip
OS X binaries:	r-release: QFASA_1.0.2.tgz, r-oldrel: QFASA_1.0.2.tgz
Old sources:	QFASA archive

Linking:

Please use the canonical form `https://CRAN.R-project.org/package=QFASA` to link to this page.

Figure 4.4: CRAN web page generated from package metadata

4.3.5 Documentation

Object documentation is the information accessed by the `help()` function in R. This type of documentation is typically used as a reference to determine the arguments taken by different functions in your package and how the arguments affect behaviour and outputs.

R provides a standard way of documenting the objects and functions in a package: you write `.Rd` files in the `man/` directory. These files use a custom syntax, loosely based on LaTeX and rendered to HTML, plain text, and PDF for viewing.

Instead of writing documentation explicitly, the *Devtools* package can be used to streamline the process. *Devtools* uses a markup system called *Roxygen2* [30] which dynamically inspects objects and functions in your package and generates documentation augmented by markup comments. It is based on *Doxygen*, a tool for generating documentation from annotated C++ sources.

Roxygen2 comments start with `#` and come before a function. All *Roxygen2* lines before a function are called a block. See Figure 4.5 which shows the block of comments before the function `AIT.dist` in the *QFASA* R package. Blocks are broken into tags which are are defined for a wide variety of information types:

- `@param` defines a function parameter.
- `@return` defines the type of value returned by the function.
- `@example` sample code demonstrating the use of the function.
- `@seealso` reference to other functions similar to this function.
- `@references` references to other documentation.

The code and markup in Figure 4.5 becomes part of the *QFASA* package documentation accessible in R via the `help()` function as per Figure 4.6.

```r
#' Returns the distance between two compositional vectors using Aitchison's
#'     distance measure.
#'
#' @export
#' @param x.1 compositional vector
#' @param x.2 compositional vector
#'
#' @references Aitchison, J., (1992) On criteria for measures of
#'     compositional difference. Mathematical Geology, 24(4), pp.365-379.
#' @references Stewart, C. (2016) An approach to measure distance between compositional
#'     diet estimates containing essential zeros. Journal of Applied Statistics,
#'     10.1080/02664763.2016.119384.
#'
AIT.dist <- function(x.1, x.2) {
    return(sqrt(sum((log(x.1/mean.geometric(x.1)) - log(x.2/mean.geometric(x.2)))^2)))
}
```

Figure 4.5: *Roxygen* markup tags on package function

```
> library(QFASA)
> help(AIT.dist)

AIT.dist                 package:QFASA                 R Documentation

Description:
 Returns the distance between two compositional vectors using Aitchison's
 distance measure.
Usage:
 AIT.dist(x.1, x.2)
Arguments:
 x.1: compositional vector
 x.2: compositional vector
References:
 Aitchison, J., (1992) On criteria for measures of compositional
 difference. Mathematical Geology, 24(4), pp.365-379.

 Stewart, C. (2016) An approach to measure distance between
 compositional diet estimates containing essential zeros. Journal
 of Applied Statistics, 10.1080/02664763.2016.119384.
```

Figure 4.6: *Roxygen* function comments available via help() function

4.3.6 Vignettes

A vignette is a long-form guide to using your package. It is like a chapter in a book or an academic paper. A vignette should divide functions into useful categories and how to co-ordinate multiple functions to solve problems. Vignettes are also useful if you want to explain the implementation details of your package.

Before R 3.0.0 the only way to create a vignette was with a tool included with the R base package called *Sweave* which works with LaTeX. Now a package can provide a reference to a vignette engine for turning files into HTML or PDF format. A commonly used vignette engine is *RMarkdown* [2]. This is a very lightweight markup language, not as powerful as LaTeX or DocBook but very easy to write and to read even in raw form.

Another tool, *Knitr* [31], allows vignette documentation to intermingle text and actual R code results. *Knitr* takes R code, executes it, captures the output, and translates it into formatted Markdown. This also allows the inclusion of plots and figures directly into vignettes.

Figure 4.7 shows a portion of a vignette source in RMarkdown from the *QFASA* R package. The header section contains meta-data regarding how the document is to be formatted and the body uses *RMarkdown* to define headings and other style elements. The rendered vignette from CRAN is shown in Figure 4.8.

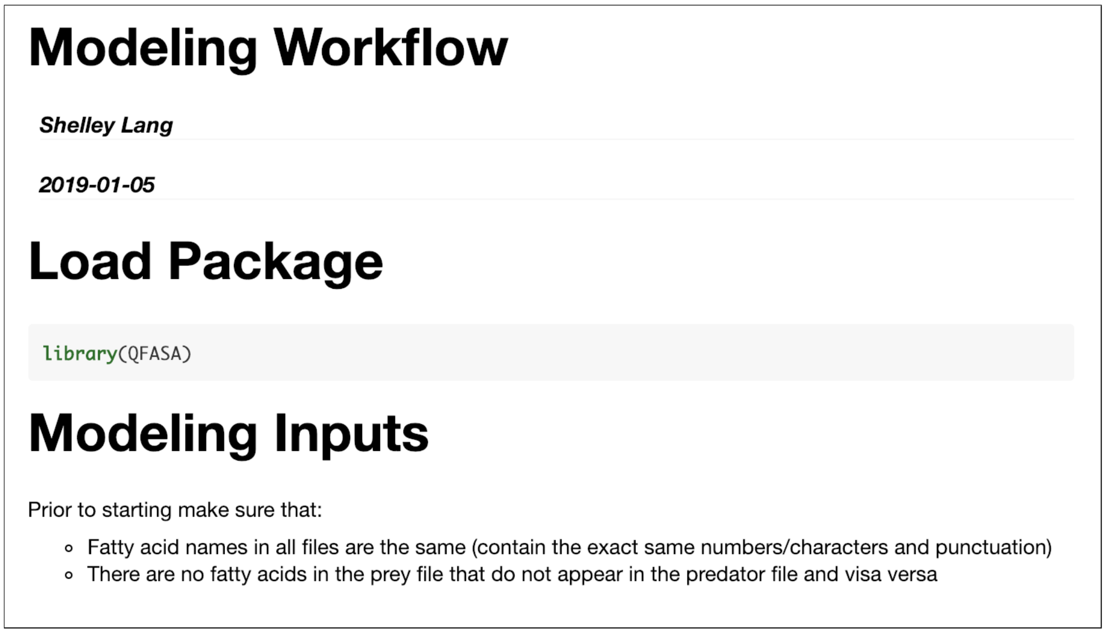
4.3.7 Testing

Testing is a software engineering best practice for repeatable delivery of reliable software. To automate software testing, code is written to test functionality and performance of a software unit, in our case an R package.

Automated testing explicitly defines a process which a programmer usually performs informally anyway. The effort expended in writing the test code is amortized over the lifetime of the project and usually pays back quickly. The tests serve as a check that any changes

~~~
---
title: "Modeling Workflow"
author: "Shelley Lang"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette:
  %\VignetteIndexEntry{Modeling Workflow}
  %\VignetteEngine{knitr::rmarkdown}
  %\VignetteEncoding{UTF-8}
---
```{r, eval=TRUE}
library(QFASA)
```
# Modeling Inputs
Prior to starting make sure that:
* Fatty acid names in all files are the same (contain the exact same
  numbers/characters and punctuation).
* There are no fatty acids in the prey file that do not appear in
  the predator file and visa versa.
~~~

Figure 4.7: Portion of vignette markup for the *QFASA* package



Modeling Workflow

Shelley Lang

2019-01-05

Load Package

```
library(QFASA)
```

Modeling Inputs

Prior to starting make sure that:

- Fatty acid names in all files are the same (contain the exact same numbers/characters and punctuation)
- There are no fatty acids in the prey file that do not appear in the predator file and visa versa

Figure 4.8: Part of the *QFASA* vignette on CRAN

```
test_that("Aitchison Distance Measure", {

    x.1 <- c(1, 1, 1, 1, 1)
    x.2 <- c(2, 99, 99, 99, 99)
    expect_equal(QFASA:::AIT.dist(x.1, x.2), 3.49003, tolerance = 1e-06)

    x.3 <- c(1, 1, 1, 1, 1)
    x.4 <- c(0, 0, 0, 0, 0)
    expect_equal(is.nan(QFASA:::AIT.dist(x.3, x.4)), TRUE)
})
```

Figure 4.9: Definition of *testthat* test function

made to a package implementation have not broken any existing functionality and that new functionality is implemented correctly. Since the tests are automated in this way they can be run often with almost no extra effort. Many tools exist to support automated testing in a variety of programming languages viz: *JUnit* for Java and *pytest* for Python.

The *Devtools* framework provides a wrapper around the *testthat* package for easy inclusion of test cases into a package. The workflow is generally to create a test file in the `tests/testthat/` directory. Its name must begin with *test* and it contains *test_that* declarations which set up a test context and then apply assertions against the result to pass or fail the test. All test cases defined in the package will be executed by calling function `devtools::test()`.

Figure 4.9 shows sample test cases for the Aitchison distance measure function, `QFASA::AIT.dist()` (see Figure 4.5). In the first case we check that the distance between vectors `x.1` and `x.2` is equal to the expected result, 3.49003, within a tolerance of 1e-6. The second test case checks that the distance function applied to `x.3` and `x.4`, a zero vector, is undefined (`is.nan()`).

Figure 4.10 shows the results of executing the test cases defined in Figure 4.9 using the `devtools::test()` function. The first test case fails since the result is not within the tolerance of the expected result.

```
> devtools::test()

Loading QFASA
Testing QFASA
1.........
Failed -------------------------------------------------------------------------
1. Failure: Aitchison Distance Measure (@testAIT.R#5) ------------------------
QFASA:::AIT.dist(x.1, x.2) not equal to 3.49003.
1/1 mismatches
[1] 7.45e-16 - 3.49 == -3.49

DONE ============================================================================
```

Figure 4.10: Execution of *testthat* test function

4.3.8 Namespaces

Namespacing helps you to avoid conflicting variables and functions in your package with those defined in other packages. It involves specifying explicitly what functions and variables are available in your package. An example of conflicting packages is the `dplyr` and `stats` packages which both implement a `filter()` function. Using these packages together requires that the conflicting function be prefixed with its package name (`dplyr::filter()` and `stats::filter()`).

The NAMESPACE file is used in a package definition to define the namespace thereof. It can be created by hand but using *Roxygen2* markup simplifies the process. Inline with object documentation, functions can be tagged in their comments as being exported by a package using the `@export` tag (see Figure 4.5). This makes code more readable in that it is clear from the code itself what functions are available to users of the package.

4.3.9 Other Components

Other components that can be included in a package structure are:

- `data/` external data you want to distribute with the package. This may include sample datasets, results of experiments, or possibly data used by automated tests. The following data sets are included the in the *QFASA* R package:

  - **CC:** sample calibration coefficients.
  - **FAset:** sample list of fatty acids.
  - **predatorFAs:** sample predator fatty acid signatures.
  - **preyFAs:** sample prey database.

- `src/` source code for package components implemented in compiled languages such as C/C++ or Fortran. This is usually done for performance reasons or to leverage a functionality in a native library such as GPU accelerated linear algebra methods.
- `inst/` any other files you want to have installed along with your package. This might include documentation, graphics files, or Java code.
- `exec/` executable scripts.
- `tools/` miscellaneous tools.

4.4 QFASA

To create the *QFASA* R package, code created in support of [14,22–25], and related papers, was re-organized and annotated to conform with the expected structure of an R package detailed in the previous sections of this chapter. The application programming interface (API) implemented by the package is detailed in Appendix A. This function-level documentation was generated automatically as part of the package from *Roxygen2* annotations in the source code itself. This is the same *QFASA* package documentation accessible in R via the `help()` function as per Figure 4.6.

Version 1.0.0 of the *QFASA* R package that we released to CRAN contained functions to support the original implementation [14]. The main function, `p.QFASA()`, calculated point estimates of predator diets from a prey FA database and predator FA samples. The compatible distance functions, AIT, KL, and CS, were also implemented as part of this release.

Versions 1.0.1 and 1.0.2 included some minor bug fixes to the original package which came to light after the package was distributed to biologists and researchers performing *QFASA* diet estimations.

Version 1.0.3 (unreleased) added three groups of functions:

- Estimating confidence intervals using ZIB distribution (see Section 3.2):
  - `beta.meths.CI()`
  - `bias.all()`
  - `prey.cluster()`
- Simulations (see Section 2.2.5):
  - `gen.pseudo.seals()`
- Comparison of diet estimates and tests for detecting dietary (see Section 3.3):
  - `testfordiff()`
  - `testfordiff.ind.boot.fun()`
  - `create.d.mat()`
  - `chisq.CA()`
  - `testfordiff.ind.pval`

In addition to restructuring the QFASA code, a significant effort was spent improving the implementation of the original confidence interval code. These improvements mostly dealt with optimizing the runtime of inference functions. These optimizations fall into one of two main categories, *root-finding* and *parallelization* which we detail in Sections 4.4.1 and 4.4.2.

4.4.1 Finding Roots

Because the mixture model used in QFASA to handle essential zeros does not yield a closed-form solution for deriving confidence intervals, bootstrap methods are used instead. As described in Section 3.2.2, this involves finding roots of $p^*(\beta_{k0}) - \alpha = 0$. In [22], a custom bisection method was used to find these roots. This custom method was replaced in the R package by the built-in `uniroot` function. In our application, root solving is computationally expensive and reducing the number of iterations required has a considerable impact on the time taken to estimate confidence intervals. Using `uniroot` reduced the number of iterations required to converge. Also, leveraging third-party packages for common algorithms allows us to abstract their implementation and focus on the particulars of our specific domain problem. Any improvements or bug fixes in these packages are easily absorbed into our work by upgrading the package in question. In both the original root finding implementation and `uniroot`, the interval to be searched for a single root is passed as an argument.

To compare confidence intervals obtained with the original bisection method compared to those obtained using `uniroot`, we conducted a simulation study. We generated 10 pseudo-predators using prey sampled from the prey database used in [22] and diet 1 defined in the same study (see Table 4.1), estimated the diet proportions using the QFASA model and obtained individual 95% confidence intervals for each species using ZIB. In the non-parametric bootstrap, 100 pseudo-predators were generated to estimate ZIB nuisance parameters. This process was repeated 1000 times. The results of this study (see Table 4.1 and 4.2) indicate that `uniroot` intervals are generally narrower and have better coverage than the original bisection method. Note that in this simulation calibration coefficients and fat content were not used so we would expect coverage probabilities to be lower in practice.

Table 4.1: Simulation 95% CIs average widths by prey species

species	uniroot	bisect	true diet (diet 1)
Cod	0.242	0.339	0.300
Haddock	0.281	0.413	0.300
Plaice	0.356	0.348	0.025
Pollock	0.128	0.193	0.150
Sandlance	0.106	0.117	0.025
SilverHake	0.122	0.153	0.150
WinterFlounder	0.158	0.165	0.025
Yellowtail	0.188	0.195	0.025

Table 4.2: Simulation 95% CIs coverage proportions by prey species

species	uniroot	bisect	true diet (diet 1)
Cod	1.00	1.00	0.300
Haddock	1.00	1.00	0.300
Plaice	1.00	0.99	0.025
Pollock	1.00	0.43	0.150
Sandlance	1.00	0.82	0.025
SilverHake	1.00	0.80	0.150
WinterFlounder	1.00	0.76	0.025
Yellowtail	0.99	0.81	0.025

4.4.2 Parallelization

The bootstrap process described in Section 4.4.1 is *embarrassingly parallel*, meaning little to no effort is required to separate the task into a number of independent sub-tasks that can execute concurrently. From a computing perspective, such tasks can easily make use of multiple CPU cores, achieving close to linear speedup in ideal cases.

Conveniently, the R *bootstrap* package is able to exploit a multi-core processor via a parallel cluster of processes. These processes are managed via the `parallel` package. The `parallel::makeCluster()` function starts a cluster with the specified number of child processes which is then used to perform bootstrap tasks in parallel.

A test was conducted in which we repeatedly performed the CI estimation described in Section 4.4.1 on different numbers of processor cores. For this purpose an AWS EC2 m4.4xlarge instance was used with 16 cores virtualized on a 2.4 GHz Intel Xeon E5-2676 v3 processor 64 GB RAM.

Figure 4.11 shows execution time as a fraction of the time taken to execute on a single processor core, plotted against the number of cores and bootstrap replicants. As is evident from the figure, execution time is reduced as the number cores increases. For 1000 parametric bootstrap replicants the speedup is significant, with execution time halving as the number of processors is doubled, and diminishing returns as we scale beyond eight processor cores. The parallelization effect is much less marked for 100 replicants. In general, saturation will occur as the overall task becomes less computationally bound.

4.5 CRAN Submission

If your intent is to release your package to CRAN, R provides tools for automatic checking of all the components covered so far. When your package is submitted to CRAN, their servers will execute the same set of checks as a first step before moving the release candidate along to the next phase. *Devtools* provides a helpful wrapper for the R checks as well as

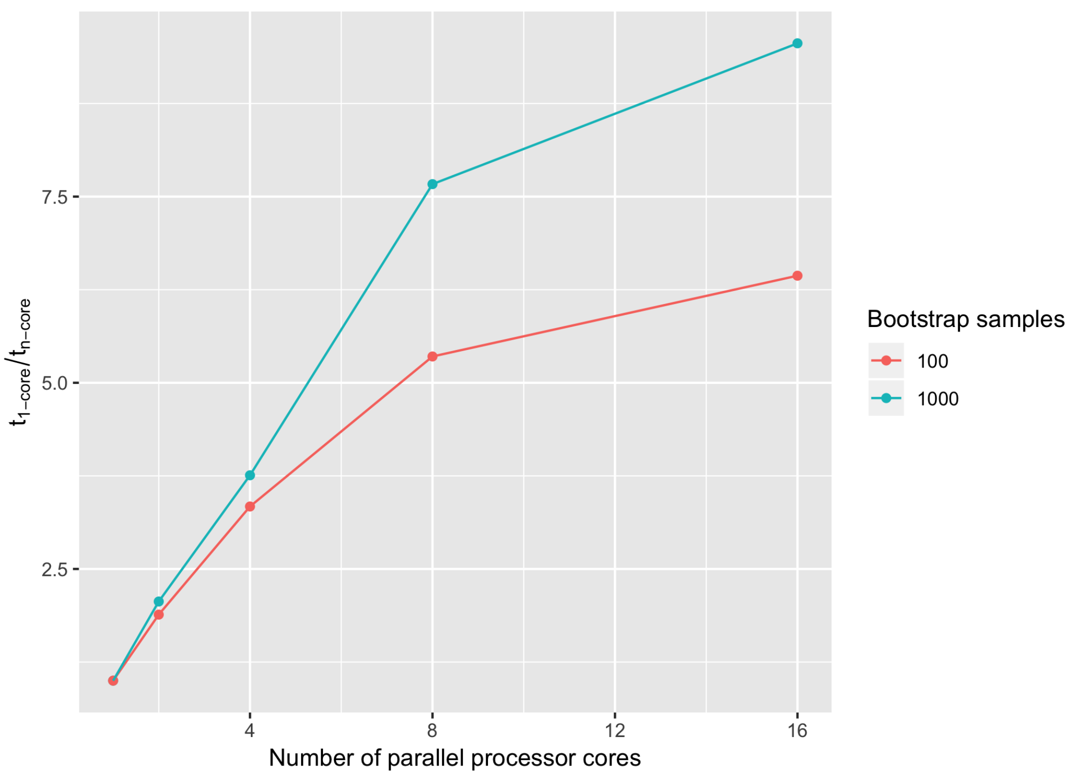


Figure 4.11: *QFASA* confidence interval estimation execution time as a fraction of the time taken to execute on a singel core

implementing additional verification that may cause issues with CRAN.

The checks may result in ERRORS and WARNINGS, which must be fixed before submission to CRAN, and NOTES, mild issues which may or may not be accepted by CRAN.

Finally, the package is built without errors or warnings, typically using `devtools:build()` and submitted to CRAN using `devtools::release()`.

You will be notified within a few minutes that the package has been received and usually within an hour whether it has passed the CRAN automated checks. If not, the package is rejected and you must address the issues listed and retry. Minor issues will be passed on to a CRAN staff member who will email you directly regarding next steps.

CRAN is staffed by volunteers who all have other full-time jobs. In a typical week there will be hundreds of submissions and only a handful of volunteers to process them all. The less work you make for them, the more likely you are to have a pleasant submission experience [28].

Upon acceptance, your package will be available on the CRAN website and propagated to mirrors with a day or two.

4.6 *qfasar* R package

The *qfasar* R package [5] is a collection of 19 functions which support published QFASA research as well as unpublished procedures intended for research support. The package was published after our *QFASA* R package and encompasses:

- Data preparation functions to validate and clean model inputs.
- Diet estimation functions using the Aitchison [25], Kullback–Liebler [14], or Chi-squared distances [25]. This functionality mirrors that provided by `p.QFASA()` in the *QFASA* R package.

- Diagnostic functions to assess the performance of a prey library using prey-on-prey simulations without calibration coefficients. This involves splitting prey-type data into two subsets and using cross validation to estimate the attribution of the leave-out group to its own type. The idea is that the degree to which estimates for a prey group are attributed to that type measures its distinctiveness within the library. Misallocation between types is indicative of prey the model may have difficulty distinguishing. In principle this is a similar process to the clustering of prey libraries in [24] demonstrated in Section 2.2.5 but can be used to assess the impact on QFASA.

- Diagnostic functions to assess goodness-of-fit of the predator diet estimate model. This diagnostic is based on the notion that the minimized distance of each predator, a byproduct of diet estimation, will be small when the fitted model explains the observed data well, and larger otherwise. The package implements a normalized goodness-of-fit metric (unpublished) based on comparing these distances to a simulated maximum-variance null distribution for each predator's minimized distance measure.

- Functions to support predator simulation.

The package does not perform inference on diet estimates and does not handle essential zeros. Download numbers from CRAN for both packages are compared in Figure 4.12 and, apart from a large spike for *qfasar* in early 2017, were found to be very similar though this data does indicate which of the packages were actually used.

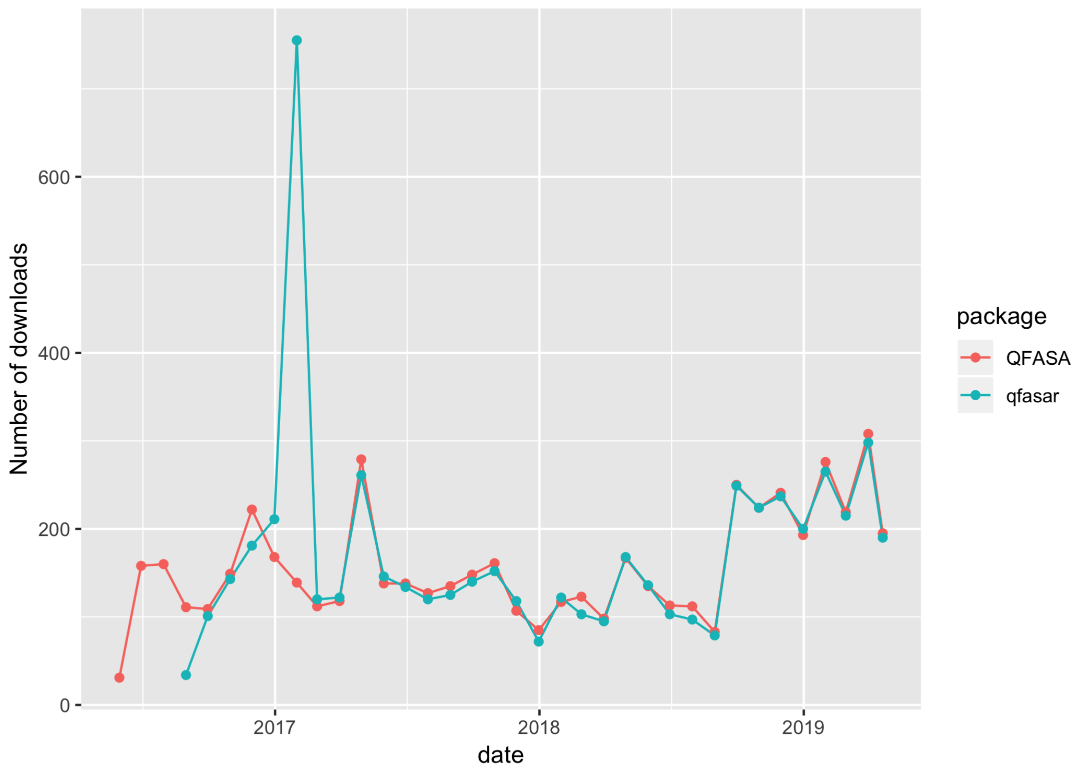


Figure 4.12: CRAN downloads of QFASA versus qfasar packages

Chapter 5

Conclusion

QFASA is an established and widely used technique used by biologists and ecologists to estimate predator diets. A body of source code has accummulated since the original study but has mostly been distributed piecemeal and updates and fixes not consistently distributed to the user community.

The primary contribution of this project was to package R source code created in support of [14, 22–25] and related papers into a FOSS module available on CRAN and to make improvements both in terms of execution speed and accuracy.

We have described the process of diet estimation via the QFASA methodology and reviewed some of the underlying statistical methodologies. We have detailed the R packaging process and our interaction with CRAN to publish the package. We described our implementation of parallel computing methods to improve the speed and efficiency of model inference by making use of multi-core processors. Finally, we also reviewed, for comparison, a similar QFASA module [5].

References

[1] John Aitchison, *The statistical analysis of compositional data*, Chapman and Hall London, 1986.

[2] JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, and Winston Chang, *rmarkdown: Dynamic documents for r*, 2018, R package version 1.10.

[3] A. Azzalini and A. Capitanio, *Statistical applications of the multivariate skew normal distribution*, Journal of the Royal Statistical Society: Series B (Statistical Methodology) **61** (1999), no. 3, 579–602.

[4] John Bear and Dean Billheimer, *A logistic normal mixture model for compositional data allowing essential zeros*, Austrian Journal of Statistics **45** (2016), no. 4, 3–23.

[5] Jeffery F. Bromaghin, *qfasar: quantitative fatty acid signature analysis with R*, Methods in Ecology and Evolution **8** (2017), no. 9, 1158–1162.

[6] Jeffrey F. Bromaghin, Suzanne M. Budge, Gregory W. Thiemann, and Karyn D. Rode, *Assessing the robustness of quantitative fatty acid signature analysis to assumption violations*, Methods in Ecology and Evolution **7** (2016), no. 1, 51–59.

[7] Jeffrey F. Bromaghin, Monique M. Lance, Elizabeth W. Elliott, Steven J. Jeffries, Alejandro Acevedo-Gutirrez, and John M. Kennish, *New insights into the diets of harbor seals (phoca vitulina) in the Salish sea revealed by analysis of fatty acid signatures.*, Fishery Bulletin **111** (2013), no. 1, 13 – 26.

[8] Jeffrey F. Bromaghin, Karyn D. Rode, Suzanne M. Budge, and Gregory W. Thiemann, *Distance measures and optimization spaces in quantitative fatty acid signature analysis*, Ecology and Evolution **5** (2015), no. 6, 1249–1262.

[9] Suzanne M. Budge, Sarah N. Penney, and Santosh P. Lall, *Estimating diets of Atlantic salmon (salmo salar) using fatty acid signature analyses; validation with controlled feeding studies*, Canadian Journal of Fisheries and Aquatic Sciences **69** (2012), no. 6, 1033–1046.

[10] Harold W. Cook and Christopher R. McMaster, *Chapter 7: Fatty acid desaturation and chain elongation in eukaryotes*, Biochemistry of Lipids, Lipoproteins and Membranes, 4th edition, New Comprehensive Biochemistry, vol. 36, Elsevier, 2002, pp. 181 – 204.

[11] Michael Greenacre, *Measuring subcompositional incoherence*, Mathematical Geosciences **43** (2011), no. 6, 681–693.

[12] Austin Happel, Logan Stratton, Colleen Kolb, Chris Hays, Jacques Rinchard, and Sergiusz Czesny, *Evaluating quantitative fatty acid signature analysis (QFASA) in fish using controlled feeding experiments*, Canadian Journal of Fisheries and Aquatic Sciences **73** (2016), no. 8, 1222–1229.

[13] Jim Hester, *lintr: A 'linter' for R code*, 2018, R package version 1.0.3.

[14] Sara J. Iverson, Christopher Field, William. Don Bowen, and Wade Blanchard, *Quantitative fatty acid signature analysis: a new method of estimating predator diets*, Ecological Monographs **74** (2004), no. 2, 211–235.

[15] J. A. Martín-Fernández, C. Barceló-Vidal, and V. Pawlowsky-Glahn, *Dealing with zeros and missing values in compositional data sets using nonparametric imputation*, Mathematical Geology **35** (2003), no. 3, 253–278.

[16] Laureline Meynier, Patrick C. H. Morel, B. Louise Chilvers, Duncan D. S. Mackenzie, and Pádraig J. Duignan, *Quantitative fatty acid signature analysis on New Zealand sea lions: model sensitivity and diet estimates*, Journal of Mammalogy **91** (2010), no. 6, 1484–1495.

[17] Philipp Neubauer and Olaf P. Jensen, *Bayesian estimation of predator diet composition from fatty acids and stable isotopes*, PeerJ **3** (2015), e920.

[18] Raydonal Ospina and Silvia LP Ferrari, *Inflated beta distributions*, Statistical Papers **51** (2010), no. 1, 111–126.

[19] K. Pearson, *On a form of spurious correlation which may arise when indices are used in the measurement of organs*, Royal Soc., London, Proc. **60** (1897), 489–502.

[20] G. Piatetsky, *Python eats away at R: Top software for analytics, data science, machine learning in 2018: Trends and analysis*, `https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html`, 2018, [Online; accessed 20-April-2019].

[21] R Core Team, *R: A language and environment for statistical computing*, `https://www.R-project.org/`, 2017, [Online; accessed 20-April-2019].

[22] Connie Stewart, *Zero-inflated beta distribution for modeling the proportions in quantitative fatty acid signature analysis*, Journal of Applied Statistics **40** (2013), no. 5, 985 (eng).

[23] Connie Stewart, *An approach to measure distance between compositional diet estimates containing essential zeros*, Journal of Applied Statistics **44** (2017), no. 7, 1137–1152.

[24] Connie Stewart and Christopher Field, *Managing the essential zeros in quantitative fatty acid signature analysis*, Journal of Agricultural, Biological, and Environmental Statistics **16** (2011), no. 1, 45–69.

[25] Connie Stewart, Sara Iverson, and Christopher Field, *Testing for a change in diet using fatty acid signatures*, Environmental and Ecological Statistics **21** (2014), no. 4, 775–792.

[26] R Core Team, *CRAN Mirror HOWTO/FAQ*, `https://cran.r-project.org/`, 2019, [Online; accessed 20-April-2019].

[27] ______, *Writing R extensions*, `https://cran.r-project.org/doc/manuals/r-release/R-exts.html`, 2019, [Online; accessed 20-April-2019].

[28] Hadley Wickham, *R packages*, 1st ed., O'Reilly Media, Inc., 2015.

[29] Hadley Wickham and Winston Chang, *devtools: Tools to make developing R packages easier*, 2017, R package version 1.13.3.

[30] Hadley Wickham, Peter Danenberg, and Manuel Eugster, *roxygen2: In-line documentation for r*, 2018, R package version 6.1.0.

[31] Yihui Xie, *knitr: A general-purpose package for dynamic report generation in r*, 2018, R package version 1.20.

[32] ______, *formatr: Format R code automatically*, 2019, R package version 1.6.

[33] Yinyu Ye, *Interior algorithms for linear, quadratic, and linearly constrained non-linear programming*, Ph.D. thesis, Department of ESS, Stanford University, 1987.

Appendix A

QFASA R Package version 1.0.3

`QFASA`	*QFASA: A package for Quantitative Fatty Acid Signature Analysis*

Description

Accurate estimates of the diets of predators are required in many areas of ecology, but for many species current methods are imprecise, limited to the last meal, and often biased. The diversity of fatty acids and their patterns in organisms, coupled with the narrow limitations on their biosynthesis, properties of digestion in monogastric animals, and the prevalence of large storage reservoirs of lipid in many predators, led us to propose the use of quantitative fatty acid signature analysis (QFASA) to study predator diets.

`AIT.dist`	*Returns the distance between two compositional vectors using Aitchison's distance measure.*

Description

Returns the distance between two compositional vectors using Aitchison's distance measure.

Usage

```
AIT.dist(x.1, x.2)
```

Arguments

`x.1`	compositional vector
`x.2`	compositional vector

References

Aitchison, J., (1992) On criteria for measures of compositional difference. Mathematical Geology, 24(4), pp.365-379.

Connie Stewart (2017) An approach to measure distance between compositional diet estimates containing essential zeros, Journal of Applied Statistics, 44:7, 1137-1152, DOI: 10.1080/02664763.2016.1193846

`AIT.more`	*Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Aitchison distance measure.*

Description

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Aitchison distance measure.

Usage

```
AIT.more(alpha, predator, prey.quantiles)
```

Arguments

`alpha`	compositional QFASA diet estimate.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

`AIT.obj`	*Used in* `solnp()` *as the objective function to be minimized when Aitchison distance measure is chosen.*

Description

Used in `solnp()` as the objective function to be minimized when Aitchison distance measure is chosen.

Usage

```
AIT.obj(alpha, predator, prey.quantiles)
```

Arguments

`alpha`	vector over which minimization takes place.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

`beta.meths.CI` *Returns individual confidence intervals and simultaneous confidence intervals based on the zero-inflated beta distribution (not bias corrected - see note below).*

Description

For details see: Stewart, C. (2013) Zero-Inflated Beta Distribution for Modeling the Proportions in Quantitative Fatty Acid Signature Analysis. Journal of Applied Statistics, 40(5), 985-992.

Usage

```
beta.meths.CI(predator.mat, prey.mat, cal.mat, dist.meas, noise, nprey,
  R.p, R.ps, R, p.mat, alpha, FC, ext.fa)
```

Arguments

`predator.mat`	matrix containing the fatty acid signatures of the predators.

`prey.mat`	prey database. A dataframe with first column a Species label and other columns fatty acid proportions. Fatty acid proportions are compositional.
`cal.mat`	matrix of calibration coefficients of predators. Each column corresponds to a different predator. At least one calibration coefficient vector must be supplied.
`dist.meas`	distance measure to use for estimation: 1=KL, 2=AIT or 3=CS
`noise`	proportion of noise to include in the simulation.
`nprey`	number of prey to sample from the the prey database when generating pseudo-predators for the nuisance parameter estimation.
`R.p`	number of beta diet distributions to generate for the nuisance parameters.
`R.ps`	number of pseudo predators to generate when estimating nuisance parameters.
`R`	number of bootstrap replicates to use when generating p-values for confidence interval estimation.
`p.mat`	matrix of predator diet estimates for which we are trying to find confidence interavls.
`alpha`	confidence interval confidence level.
`FC`	vector of prey fat content. Note that this vector is passed to the `gen.pseudo.seals()` which expects fat content values for individual prey samples while `pseaudo.seal()` and `p.QFASA()` expect a species average.
`ext.fa`	subset of fatty acids to be used to obtain QFASA diet estimates.

Details

Note:

- These intervals are biased and should be corrected using the output from `bias.all()`.
- `CI.L.1` and `CI.U.1` contain the simultaneous confidence intervals.
- Slow because of bisection and lots of repetition.
- Need to replace `p.prey` with `p.QFASA()` eventually but just use p.prey() for now. Use example where we estimate a single predator diet to compare the estimates from each method.

Value

Individual confidence intervals and simultaneous confidence intervals based on the zero-inflated beta distribution. These intervals are biased and should be corrected using the output from `bias.all()`. `ci.l.1` and `ci.u.1` contain the simultaneous confidence intervals.

References

Stewart, C. (2013) Zero-inflated beta distribution for modeling the proportions in quantitative fatty acid signature analysis. Journal of Applied Statistics, 40(5), 985-992.

Examples

```
 ## Fatty Acids
 data(FAset)
 fa.set = as.vector(unlist(FAset))

 ## Predators
 data(predatorFAs)
 tombstone.info = predatorFAs[,1:4]
 predator.matrix = predatorFAs[,5:(ncol(predatorFAs))]
 npredators = nrow(predator.matrix)

 ## Prey
 data(preyFAs)
 prey.sub=(preyFAs[,4:(ncol(preyFAs))])[fa.set]
 prey.sub=prey.sub/apply(prey.sub,1,sum)
 group=as.vector(preyFAs$Species)
 prey.matrix=cbind(group,prey.sub)
 prey.matrix=MEANmeth(prey.matrix)


# Diet estimate
diet.est <- p.QFASA(predator.mat = predator.matrix,
                    prey.mat = prey.db.summarized,.matrix,
                    cal.mat = rep(1, nrow(FAset)),
                    dist.meas = 2,
                    ext.fa = colnames(prey.db.summarized))[['Diet Estimates']]

# Confidence intervals
ci = beta.meths.CI(predator.mat = predator.matrix,
                   prey.mat = prey.matrix,
                   cal.mat = rep(1, nrow(FAset)),
                   dist.meas = 2,
                   noise = 0,
                   nprey = 50,
                   R.p = 1,
                   R.ps = 10,
                   R = 10,
                   p.mat = diet.est,
                   alpha = 0.05,
                   FC = rep(1, nrow(prey.db)),
                   ext.fa = FAset$FA)

# Bias correction
bias <- bias.all(p.mat = diet.est,
                 prey.mat = prey.matrix,
                 cal.mat = as.matrix(rep(1, nrow(FAset))),
                 fat.cont = rep(1, nrow(prey.db)),
                 R.bias = 10,
                 noise = 0,
                 nprey = 50,
                 specify.noise = rep(FALSE, nspecies),
                 dist.meas = 2,
                 ext.fa = FAset$FA)

# SIMULTANEOUS CONFIDENCE INTERVALS:
# LOWER LIMIT
ci[[1]] - bias[3,]

# UPPER LIMIT
ci[[2]] - bias[3,]
```

`bias.all` *Calculate bias correction for confidence intervals from* `beta.meths.CI()`.

Description

Calculate bias correction for confidence intervals from `beta.meths.CI()`.

Usage

```
bias.all(p.mat, prey.mat, cal.mat, fat.cont, R.bias, noise, nprey,
  specify.noise, dist.meas, ext.fa = ext.common.fa.list)
```

Arguments

`p.mat`	matrix containing the fatty acid signatures of the predators.
`prey.mat`	matrix containing a representative fatty acid signature
`cal.mat`	matrix of calibration factors where the i th column is to be used with the i th predator. If modelling is to be done without calibration coefficients, simply pass a vector or matrix of ones.
`fat.cont`	prey fat content
`R.bias`	botstrap replicates
`noise`	noise
`nprey`	number of prey
`specify.noise`	noise
`dist.meas`	distance measure
`ext.fa`	subset of FA's to use.

Value

Row 1 is Lambda CI, row 2 is Lambda skew, and row 3 is Beta CI

`CC` *Fatty acid calibration coefficients.*

Description

Fatty acid calibration coefficients.

Usage

```
CC
```

Format

A data frame with 66 observations and 2 variables:

FA fatty acid names

CC calibration coefficient for corresponding fatty acid

`chisq.CA` *Called by* `create.d.mat()` *to compute the chi-square distance.*

Description

Called by `create.d.mat()` to compute the chi-square distance.

Usage

```
chisq.CA(x1, x2)
```

Arguments

`x1`	vector
`x2`	vector

`chisq.dist` *Returns the distance between two compositional vectors using the chi-square distance.*

Description

Returns the distance between two compositional vectors using the chi-square distance.

Usage

```
chisq.dist(x.1, x.2, gamma)
```

Arguments

`x.1`	compositional vector
`x.2`	compositional vector
`gamma`	power transform taken to be 1.

References

Stewart, C., Iverson, S. and Field, C. (2014) Testing for a change in diet using fatty acid signatures. Environmental and Ecological Statistics 21, pp. 775-792.

Connie Stewart (2017) An approach to measure distance between compositional diet estimates containing essential zeros, Journal of Applied Statistics, 44:7, 1137-1152, DOI: 10.1080/02664763.2016.1193846

`CS.more`	*Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and chi-square distance measure.*

Description

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and chi-square distance measure.

Usage

```
CS.more(alpha, predator, prey.quantiles, gamma)
```

Arguments

`alpha`	compositional QFASA diet estimate.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
`gamma`	power transform exponent (see `chisq.dist()`).

`CS.obj`	*Used in* `solnp()` *as the objective function to be minimized when chi-square distance measure is chosen. Unlike* `AIT.obj()` *and* `KL.obj()`*, does not require modifying zeros.*

Description

Used in `solnp()` as the objective function to be minimized when chi-square distance measure is chosen. Unlike `AIT.obj()` and `KL.obj()`, does not require modifying zeros.

Usage

```
CS.obj(alpha, predator, prey.quantiles, gamma)
```

Arguments

`alpha`	vector over which minimization takes place.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
`gamma`	power transform exponent (see `chisq.dist()`).

`create.d.mat` *Called by* `testfordiff.ind.boot.fun()` *to create a matrix of distances.*

Description

Called by `testfordiff.ind.boot.fun()` to create a matrix of distances.

Usage

```
create.d.mat(Y.1, Y.2)
```

Arguments

`Y.1`	vector
`Y.2`	vector

`FAset` *List of fatty acids used in sample prey and predator data sets,* `preyFAs` *and* `predatorFAs` *respectively.*

Description

List of fatty acids used in sample prey and predator data sets, `preyFAs` and `predatorFAs` respectively.

Usage

```
FAset
```

Format

A data frame with 39 observations and 1 variable:

FA Fatty acid name

`KL.dist`	*Returns the distance between two compositional vectors using Kullback–Leibler distance measure.*

Description

Returns the distance between two compositional vectors using Kullback–Leibler distance measure.

Usage

```
KL.dist(x.1, x.2)
```

Arguments

`x.1`	compositional vector
`x.2`	compositional vector

References

S.J. Iverson, C. Field, W.D. Bowen, and W. Blanchard (2004) Quantitative fatty acid signature analysis: A new method of estimating predator diets, Ecological Monographs 72, pp. 211-235.

`KL.more`	*Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Kullback-Leibler distance measure.*

Description

Used to provide additional information on various model components evaluated at the optimal solution, i.e., using the QFASA diet estimates and Kullback-Leibler distance measure.

Usage

```
KL.more(alpha, predator, prey.quantiles)
```

Arguments

`alpha`	compositional QFASA diet estimate.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

`KL.obj` *Used in* `solnp()` *as the objective function to be minimized when Kullback–Leibler distance measure is chosen.*

Description

Used in `solnp()` as the objective function to be minimized when Kullback–Leibler distance measure is chosen.

Usage

```
KL.obj(alpha, predator, prey.quantiles)
```

Arguments

`alpha`	vector over which minimization takes place.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.

`MEANmeth` *Returns the multivariate mean FA signature of each prey group entered into the QFASA model. Result can be passed to prey.mat in* `p.QFASA()`*.*

Description

Returns the multivariate mean FA signature of each prey group entered into the QFASA model. Result can be passed to prey.mat in `p.QFASA()`.

Usage

```
MEANmeth(prey.mat)
```

Arguments

`prey.mat`	matrix containing the FA signatures of the prey. The first column indexes the prey group.

`p.QFASA`	*Computes the diet estimate for each predator in seal.mat using either the Kullback-Leibler Distance (KL), the Aitchison Distance (AIT) or the Chi-Square Distance (CS).*

Description

Computes the diet estimate for each predator in seal.mat using either the Kullback-Leibler Distance (KL), the Aitchison Distance (AIT) or the Chi-Square Distance (CS).

Usage

```
p.QFASA(predator.mat, prey.mat, cal.mat, dist.meas, gamma = 1,
  FC = rep(1, nrow(prey.mat)), start.val = rep(0.99999,
  nrow(prey.mat)), ext.fa)
```

Arguments

`predator.mat`	matrix containing the FA signatures of the predators.
`prey.mat`	matrix containing a representative FA signature from each prey group (usually the mean). The first column must index the prey group.
`cal.mat`	matrix of calibration factors where the i th column is to be used with the i th predator. If modelling is to be done without calibration coefficients, simply pass a vector or matrix of ones.
`dist.meas`	distance measure to use for estimation: 1=KL, 2=AIT or 3=CS
`gamma`	parameter required for calculations using CS distance (passed to CS.obj). Currently being set to 1.
`FC`	vector of fat content
`start.val`	initial vector of parameters to be optimized
`ext.fa`	subset of fatty acids to be used to obtain QFASA diet estimates.

Value

a list with components:

Diet Estimates	A matrix of the diet estimates for each predator where each row corresponds to a predator and the columns to prey species. The estimates are expressed as proportions summing to one.
Additional Measures	For each predator for which a diet estimate was obtained:
`ModFAS`	the value of the modelled fatty acid (i.e., after CCs have been applied and the fatty acids subsetted and renormalised over the designated fatty acid set). These are expressed as proportions summing to one.
`DistCont`	The contribution of each fatty acid to the final minimized distance.
`PropDistCont`	The contribution of each fatty acid to the final minimized distance as a proportion of the total.
`MinDist`	The final minimized distance.

Examples

```
## Fatty Acids
data(FAset)
fa.set = as.vector(unlist(FAset))

## Predators
data(predatorFAs)
tombstone.info = predatorFAs[,1:4]
predator.matrix = predatorFAs[,5:(ncol(predatorFAs))]
npredators = nrow(predator.matrix)

## Prey
data(preyFAs)
prey.sub=(preyFAs[,4:(ncol(preyFAs))])[fa.set]
prey.sub=prey.sub/apply(prey.sub,1,sum)
group=as.vector(preyFAs$Species)
prey.matrix=cbind(group,prey.sub)
prey.matrix=MEANmeth(prey.matrix)
```

```
FC = preyFAs[,c(2,3)]
FC = as.vector(tapply(FC$lipid,FC$Species,mean,na.rm=TRUE))

## Calibration Coefficients
data(CC)
cal.vec = CC[,2]
cal.mat = replicate(npredators, cal.vec)

# Run QFASA
Q = p.QFASA(predator.matrix,
            prey.matrix,
            cal.mat,
            dist.meas=1,
            gamma=1,
            FC,
            start.val = rep(1,nrow(prey.matrix)),
            fa.set)
```

`prey.cluster`	*This function performs a hierarchical cluster analysis of prey fatty acid signatures using a matrix of dissimilarities for the n objects being clustered. Initially, each object is assigned as its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, until there is just a single cluster.*

Description

This function performs a hierarchical cluster analysis of prey fatty acid signatures using a matrix of dissimilarities for the n objects being clustered. Initially, each object is assigned as its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, until there is just a single cluster.

Usage

```
prey.cluster(prey.fa, method, FUN)
```

Arguments

`prey.fa`	data frame of prey fatty acid signature samples. Species column is used to group samples. Other columns are assumed to be fatty acid proportions.
`method`	the agglomeration method to be used. This should be one of `'single'`, `'complete'`, `'average'`, `'median'`, `'centroid'`.
`FUN`	distance function

Value

an object of class `hclust` which describes the tree produced by the clustering process.

`pseudo.pred` *Generate a pseudo predator by sampling with replacement from prey database.*

Description

Note: if preysize=1, then one prey is selecting from each species. otherwise, a sample of size n_k (number of species k) is sampled with replacement.

Usage

```
pseudo.pred(diet, preybase, cal.vec, fat.vec, preysize = 2)
```

Arguments

`diet`	compositional vector of proportions that sums to one. Length is equal to the number of prey species.
`preybase`	prey database with first column providing the species name.
`cal.vec`	vector of calibration coefficients.
`fat.vec`	vector of fat content whose length is the same as the number of species.

Value

a simulated predator FA signature

Examples

```
data(preyFAs)
p.mat <- matrix(rep(NA,100*11),nrow=100)
for (i in 1: 100) {
    my.seal <- pseudo.pred(rep(1/11,11),
                           preyFAs[,-c(1,3)],
                           rep(1,ncol(preyFAs[,-c(1,3)])-1),
                           rep(1,11))
    p.mat[i,] <- p.QFASA(my.seal,
                         MEANmeth(preyFAs[,-c(1,3)]),
                         rep(1,length(my.seal)),
                         2,
                         ext.fa=colnames(preyFAs[,-c(1:3)]))$`Diet Estimates`
}

# Average diet estimate
round(apply(p.mat,2,mean),3)
```

`QFASA.const.eqn` *Returns* `sum(alpha)` *and used in* `solnp()`*.*

Description

Returns `sum(alpha)` and used in `solnp()`.

Usage

```
QFASA.const.eqn(alpha, predator, prey.quantiles, gamma)
```

Arguments

`alpha`	vector over which minimization takes place.
`predator`	fatty acid signature of predator.
`prey.quantiles`	matrix of fatty acid signatures of prey. Each row contains an individual prey signature from a different species.
`gamma`	power transform exponent (see chisq.dist).

`split.prey` *Splits prey database into a simulation set (1/3) and a modelling set (2/3). If number of samples of a prey type is less than or equal to 5, then* `prey.mod` *and* `prey.sim` *are duplicated instead of split.*

Description

Splits prey database into a simulation set (1/3) and a modelling set (2/3). If number of samples of a prey type is less than or equal to 5, then `prey.mod` and `prey.sim` are duplicated instead of split.

Usage

```
## S3 method for class 'prey'
split(prey.mat)
```

Arguments

`prey.mat`	matrix of individual prey fatty acid signatures where the first column denotes the prey type

Value

list of simulation prey database and modelling prey database.

`testfordiff.ind.boot`

Called by `testfordiff.ind.pval()`*.*

Description

Called by `testfordiff.ind.pval()`.

Usage

```
testfordiff.ind.boot(data, ns1, R)
```

Arguments

`data`	sample of compositional data
`ns1`	sample size of compdata.1
`R`	number of bootstrap samples. default is 500.

`testfordiff.ind.boot.fun` *Called by* `testfordiff.ind.boot()`*.*

Description

Called by `testfordiff.ind.boot()`.

Usage

```
testfordiff.ind.boot.fun(data, i, ns1, change.zero = 1e-05)
```

Arguments

`data`	sample of compositional data
`i`	
`ns1`	sample size of compdata.1
`change.zero`	

`testfordiff.ind.pval` *Test for a difference between two independent samples of compositional data. Zeros of any type are allowed.*

Description

Test for a difference between two independent samples of compositional data. Zeros of any type are allowed.

Usage

```
testfordiff.ind.pval(compdata.1, compdata.2, ns1, R = 500)
```

Arguments

`compdata.1`	sample of compositional data.
`compdata.2`	sample of compositional data.
`ns1`	sample size of compdata.1.
`R`	number of bootstrap samples, default is 500.

Value

p-value obtained through a multivariate permutation test with test statistic based on chi-square distances.

References

Stewart, C., Iverson, S. and Field, C. (2014) Testing for a change in diet using fatty acid signatures. Environmental and Ecological Statistics 21, pp. 775-792.

Examples

```
## Prey
data(preyFAs)

## Capelin FA sig
capelin.sig=preyFAs[preyFAs$Species=="capelin",4:(ncol(preyFAs))]
capelin.sig=capelin.sig/apply(capelin.sig,1,sum)

## Sandlance FA sig
sandlance.sig=preyFAs[preyFAs$Species=="sandlance",4:(ncol(preyFAs))]
sandlance.sig=sandlance.sig/apply(sandlance.sig,1,sum)
```

```
## Run testfordiff.ind.pval.1
testfordiff.ind.pval(as.matrix(capelin.sig),
                     as.matrix(sandlance.sig),
                     nrow(capelin.sig))
```

Vita

Candidate's full name: Justin Kamerman
University attended:
MSc Computer Science, Univ. of South Africa, 2004
BSc Mechanical Engineering, Univ. of Witwatersrand, 1993
Publications: None
Conference Presentations: None