```
gap> g:= SymmetricGroup( 4 );
Sym( [ 1 .. 4 ] )
gap> tbl:= CharacterTable( g );;  HasIrr( tbl );
false
gap> tblmod2:= CharacterTable( tbl, 2 );
BrauerTable( Sym( [ 1 .. 4 ] ), 2 )
gap> tblmod2 = CharacterTable( tbl, 2 );
true
gap> tblmod2 = BrauerTable( tbl, 2 );
true
gap> libtbl:= CharacterTable( "M" );
CharacterTable( "M" )
gap> BrauerTable( libtbl, 2 );
fail
gap> CharacterTable( "Symmetric", 4 );
CharacterTable( "Sym(4)" )
gap> ComputedBrauerTables( tbl );
[ , BrauerTable( Sym( [ 1 .. 4 ] ), 2 ) ]
```

```
i5 : betti(t,Weights=>{1,0})

            0 1  2  3 4
o5 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: . .  4  . 2
         4: . .  .  4 .
         5: . .  2  . .

o5 : BettiTally
i6 : betti(t,Weights=>{0,1})

            0 1  2  3 4
o6 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: . .  4  . 2
         4: . .  .  4 .
         5: . .  2  . .

o6 : BettiTally
i7 : t1 = betti(t,Weights=>{1,1})

            0 1  2  3 4
o7 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . .  .  . .
         2: . .  .  . .
         3: . 2  .  . .
         4: . .  .  . .
         5: . 2  .  . .
         6: . .  1  . .
         7: . .  8  6 .
         8: . .  4  8 4

o7 : BettiTally
i8 : peek t1

o8 = BettiTally{(0, {0, 0}, 0) => 1 }
                (1, {2, 2}, 4) => 2
                (1, {3, 3}, 6) => 2
                (2, {3, 7}, 10) => 2
                (2, {4, 4}, 8) => 1
                (2, {4, 5}, 9) => 4
                (2, {5, 4}, 9) => 4
                (2, {7, 3}, 10) => 2
                (3, {4, 7}, 11) => 4
                (3, {5, 5}, 10) => 6
                (3, {6, 6}, 12) => 4
                (4, {6, 7}, 12) => 2
                (4, {7, 5}, 12) => 2
```

```
ring r1 = 32003,(x,y,z),ds;
int a,b,c,t=11,5,3,0;
poly f = x^a+y^b+z^(3*c)+x^(c+2)*y^(c-1)+x^
         x^(c-2)*y^c*(y^2+t*x)^2;
option(noprot);
timer=1;
ring r2 = 32003,(x,y,z),dp;
poly f=imap(r1,f);
ideal j=jacob(f);
vdim(std(j));
==> 536
vdim(std(j+f));
==> 195
timer=0;  // reset timer
```

Journal of Software for Algebra and Geometry

Tropical geometric tools for machine learning: the TML package

David Barnhill, Ruriko Yoshida, Georgios Aliatimis and Keiji Miura

vol 14 2024

JSAG **14** (2024), 133–174
https://doi.org/10.2140/jsag.2024.14.133

The Journal of Software for
Algebra and Geometry

Tropical geometric tools for machine learning: the TML package

DAVID BARNHILL, RURIKO YOSHIDA, GEORGIOS ALIATIMIS AND KEIJI MIURA

ABSTRACT: In the last decade, developments in tropical geometry have provided a number of uses directly applicable to problems in statistical learning. The **TML** package is the first R package which contains a comprehensive set of tools and methods used for basic computations related to tropical convexity, visualization of tropically convex sets, as well as supervised and unsupervised learning models using the tropical metric under the max-plus algebra over the tropical projective torus. Primarily, the **TML** package employs a Hit-and-Run Markov chain Monte Carlo sampler in conjunction with the tropical metric as its main tool for statistical inference. In addition to basic computation and various applications of the tropical HAR sampler, we also focus on several supervised and unsupervised methods incorporated in the **TML** package including tropical principal component analysis, tropical logistic regression and tropical kernel density estimation.

1. INTRODUCTION. Tropical geometry is a relatively young field which involves examining the characteristics of geometric structures defined by the solution set of a series of a polynomial equations in max-plus, or tropical, algebra. Alternatively, tropical geometry can be described as the piecewise-linear analogue of classical geometry, as discussed in [9; 21]. In general, tropical geometry focuses on structures existing in the *tropical projective torus* defined as the quotient space $\mathbb{R}^e/\mathbb{R}\mathbf{1}$ which is isomorphic to $\mathbb{R}^{e-1}$. To date researchers have focused much attention on the theoretical underpinnings of tropical algebra and geometry; see [9; 13; 21; 30] for a thorough treatment.

As with any mathematical field, it is natural to examine how the ideas of tropical geometry can be used to solve various statistical problems. However, while methods are currently being developed for applied problems associated with tropical geometry, few comprehensive resources exist to handle such problems. In this article we attempt to fill this need by introducing the **TML** package [5], which provides a number of tropical statistical methods developed for use on problems associated with tropical geometry in the R programming language [27]. The **TML** package can be obtained through the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/TML/index.html.

1A. ***Data science and tropical geometry.*** Statistical tools in data science are often classified in terms of supervised and unsupervised learning. In the case of supervised learning, data observations possess a dependent variable in the form of a label or quantity of interest which is used to train models in order to predict outcomes or classify unseen data based on those labels. Unsupervised learning is descriptive in

MSC2020: 14T90, 62G09, 62R01.
Keywords: tropical machine learning, tropical geometry, tropical data science.
TML-SCRIPT-V3.rtf version 3

nature, where data observations possess no predetermined response variable. The goal in unsupervised learning is to better understand the relationships between data observations or intuit the underlying structure of the data. A thorough yet concise summary with a number of open problems related to supervised and unsupervised learning as applied to tropical geometry can be found in [44]. We leverage these terms in this article as well for methods like tropical logistic regression (supervised learning), tropical principal component analysis (unsupervised learning), and tropical kernel density estimation (unsupervised learning). We also give significant focus to Markov chain Monte Carlo methods that do not necessarily fall exclusively in one class of machine learning but can be incorporated into supervised and unsupervised techniques or standalone for statistical inference computations.

While the research to develop tropical geometric statistical tools is nascent, there have been significant developments in a number of areas. One powerful tool in Euclidean space is the Markov chain Monte Carlo (MCMC) sampler, which combines Monte Carlo sampling with Markov chains. Instead of Monte Carlo sampling according to a specific distribution, MCMC methods sample points by building a Markov chain that converges to a desired target distribution [31]. The first Markov chain Monte Carlo Hit-and-Run (HAR) sampling technique for use in the tropical projective torus was introduced in [48]. This method samples points uniformly from polytropes, which are tropical polytopes that are classically convex, as well as full-dimensional elements of any generic tropical simplex (see Definition 1.27). The authors also show how to sample points over the space of ultrametrics using line segments [48]. This method was extended to show how to sample points about a center of mass with a given dispersion parameter in a method akin to Gaussian sampling in Euclidean space in [3]. These aforementioned HAR samplers feature prominently in several aspects of statistical learning as they have been applied in a number of settings. The HAR methods were employed in [47] to execute nonparametric density estimation techniques over the space of ultrametrics which is known to be tropically convex [36]. How to employ HAR methods to estimate the volume of a tropical polytope was shown in [6]. Being able to sample points this way paves the way for approaching statistical problems ranging from integral estimation to optimization in the setting of the tropical projective torus.

In supervised learning, Akian, et al. define in [1] the idea of tropical linear regression as the best approximation of a set of data observations using a tropical hyperplane. They then show the relationship of the tropical hyperplane approximation with mean payoff games. Supervised classification method called *tropical logistic regression* for use over the space of rooted and equidistant phylogenetic trees was introduced in [2]. In this case, phylogenetic trees are defined in terms of ultrametrics and are classified into one of two *species trees* (see Section 1D). Tropical analogues of other supervised methods exist such as tropical support vector machines [10]. Specifically, Gartner and Jaggi define in [10] the notion of the tropical support vector machine (SVM) as a binary classification mechanism. Extensions of tropical SVMs as classifiers are exhibited in both [38] and [46].

Research into tropical unsupervised methods is also burgeoning. The tropical analogue of principal component analysis (PCA), where the n-th order principal component can be represented as the best fit tropical polytope for a set of observations that are ultrametrics is introduced in [45]. It should be

noted that tropical PCA methods have been adapted to use HAR methods to find the best fit polytope. The tropical analogue of k-means and hierarchical clustering over the tropical projective torus as well as focusing on the space of ultrametrics was introduced in [4].

1B. ***The TML package.*** Availability of software tools in tropical geometry is plentiful when it comes to computer algebra. Singular is a computer algebra system for polynomial computations with emphasis on commutative algebra, algebraic geometry, and singularity theory which includes functionality for use in tropical geometry [33]. However, while Singular provides functionality for applications related to tropical geometry, there is no functionality for specific statistical methods. Similarly, polymake is a software used for research in polyhedral geometry which includes tropical geometry [12]. As with Singular, polymake focuses on the geometric and combinatorial analysis of polytopes. Additionally, polymake provides nice visualization options for tropical polytopes. The Open Source Computer Algebra Research (OSCAR) project provides an extensive corpus of tools by combining the functionality of several software environments including both Singular and polymake for use in the Julia programming language [24].

While there exists a significant number of software programs that include a focus on tropical geometry as it relates to computer algebra, few resources exist specifically for statistical computation. And though the **algstat** package for the R programming language specifically focuses on algebra statistics, it provides no tools for the tropical case [16]. In fact, there is no comprehensive suite of tropical statistical tools available. Nonetheless, some functionality exists in a piecemeal manner in the R programming language. Supervised and unsupervised learning methods are available in the **RTropical** package which makes use of tropical SVMs as well as tropical PCA over the space of phylogenetic trees [39]. Basic tropical arithmetic functions are available in the **tropical** package [15].

In this article we introduce the **TML** which serves to provide a comprehensive suite of statistical tools applicable to tropical geometry for use in the R programming language. The package consists of functions and methods ranging from basic tropical arithmetic and linear algebra functions to more complex supervised and unsupervised tropical machine learning techniques. The **TML** package is distributed on the Comprehensive R Archive Network (CRAN) with version control managed through Git on GitHub (https://github.com/barnhilldave/TML).

The organization of this article is as follows. In Section 1C we offer the essential elements of tropical geometry that provide the background for the methods introduced in the **TML** package. In Section 2 we illustrate basic loading and operations of the **TML** package as well as visualization methods. Because MCMC methods feature prominently in the **TML** package, Section 3 focuses on the tropical HAR sampler developed in [48] called *vertex HAR with extrapolation* and illustrates its usage. Section 4 follows with examples for the prominent methods of statistical inference in the **TML** package. These applications include volume estimation of tropical polytopes, tropical logistic regression, tropical PCA, and tropical kernel density estimation. We finish the article with concluding remarks and potential future developments.

1C. ***Tropical basics.*** In this section we provide the necessary tropical geometric background, notation, and terminology as it relates to functions in the **TML** package. This section is divided into two subsections. The first section focuses on tropical analogues of arithmetic and linear algebraic operations. The second subsection defines several essential concepts associated with tropical polyhedral geometry. Where appropriate we provide examples associated with the definitions in this section.

Tropical arithmetic. In general, research in tropical geometry studies the properties of the space defined by the *tropical semiring* represented by the triplet $(\mathbb{R}\cup\{-\infty\}, \oplus, \odot)$. Here, classical addition is replaced by *max* $(\oplus)$ and classical multiplication is replaced by classical addition $(\odot)$ [9; 21]. This space is known as the *tropical projective torus* represented by $\mathbb{R}^e/\mathbb{R}\mathbf{1}$, where $\mathbf{1} := (1, 1, \ldots, 1)$ is the vector with all ones in $\mathbb{R}^e$. This requires that if $v := (v_1, \ldots, v_e) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$,

$$(v_1 + c, \ldots, v_e + c) = (v_1, \ldots, v_e) = v. \tag{1}$$

For an extensive treatment, see [13] and [21].

Example 1.1. Suppose we have $e = 3$ and let $v = (1, 2, 3) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$. Then

$$v = (1, 2, 3) = (0, 1, 2).$$

Definition 1.2 (tropical arithmetic operations). Under the tropical semiring with the tropical, or *max-plus algebra* $(\mathbb{R}\cup\{-\infty\}, \oplus, \odot)$, we have the arithmetic operations of addition and multiplication defined as

$$x \oplus y := \max\{x, y\}, \quad x \odot y := x + y \quad \text{where } x, y \in \mathbb{R}\cup\{-\infty\}.$$

Here $-\infty$ is the identity element under addition $\oplus$ and 0 is the identity element under multiplication $\odot$.

We may also define the tropical semiring with the *min-plus algebra* $(\mathbb{R}\cup\{\infty\}, \boxplus, \odot)$, where arithmetic operations of addition and multiplication defined as:

$$x \boxplus y := \min\{x, y\}, \quad x \odot y := x + y \quad \text{where } x, y \in \mathbb{R}\cup\{\infty\}.$$

Remark 1.3. Note that the min-plus algebra and max-plus algebra are related by the classical multiplication by (-1) since the min operation and the max operation are homogeneous.

Example 1.4. Let $x = 1$, $y = -1$. Then we have

$$1 \oplus (-1) = \max\{1, -1\} = 1, \tag{2}$$

$$1 \odot (-1) = 1 + (-1) = 0, \tag{3}$$

$$1 \boxplus (-1) = \min\{1, -1\} = -1. \tag{4}$$

Definition 1.5 (tropical scalar multiplication and vector addition). For any $x, y \in \mathbb{R}\cup\{-\infty\}$ and for any $v = (v_1, \ldots, v_e)$, $w = (w_1, \ldots, w_e) \in (\mathbb{R}\cup\{-\infty\})^e$, we have tropical scalar multiplication and tropical vector addition defined as:

$$x \odot v \oplus y \odot w := (\max\{x + v_1, y + w_1\}, \ldots, \max\{x + v_e, y + w_e\}).$$

Example 1.6. Suppose we have $e = 3$ and let $v = (0, 1, 2)$, $w = (0, 0, 0) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$. Then

$$\begin{aligned} 1 \odot v \oplus (-1) \odot w &= (\max\{1+0, (-1)+0\}, \max\{1+1, (-1)+0\}, \max\{1+2, (-1)+0\}) \\ &= (1, 2, 3). \end{aligned}$$

Definition 1.7 (generalized Hilbert projective metric). For any tropical points $v := (v_1, \ldots, v_e)$, $w := (w_1, \ldots, w_e) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ where $[e] := \{1, \ldots, e\}$, the *tropical distance* (also known as *tropical metric*) d_{tr} between v and w is defined as

$$d_{\text{tr}}(v, w) := \max_{i \in [e]}\{v_i - w_i\} - \min_{i \in [e]}\{v_i - w_i\}.$$

For any two points $v, w \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$, the tropical distance between v and w assumes equation (1) holds. Otherwise the tropical distance is not a metric.

Example 1.8. Suppose we have $e = 3$ and let $v = (0, 1, 2)$, $w = (0, 0, 0) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$. Then

$$d_{\text{tr}}(v, w) = \max\{0-0, 1-0, 2-0\} - \min\{0-0, 1-0, 2-0\} = 2 - 0 = 2.$$

It is worth to note that d_{tr} is not a metric over $\mathbb{R}^d$. For example, if we take $u' = (3, 3, 3)$, $w' = (0, 0, 0) \in \mathbb{R}^3$. Then, we have

$$d_{\text{tr}}(u, w) = \max\{3-0, 3-0, 3-0\} - \min\{3-0, 3-0, 3-0\} = 3 - 3 = 0.$$

But, $u' \neq w'$ over $\mathbb{R}^3$. Therefore d_{tr} is not a metric over $\mathbb{R}^3$. However,

$$(3, 3, 3) = (0, 0, 0)$$

over $\mathbb{R}^3/\mathbb{R}\mathbf{1}$. Therefore, d_{tr} is metric over $\mathbb{R}^3/\mathbb{R}\mathbf{1}$.

Definition 1.9 (tropical determinant [13]). Let w be a positive integer. For any square tropical matrix B of size $w \times w$ with entries in $\mathbb{R} \cup \{-\infty\}$, we say the tropical determinant of B as follows:

$$\text{tdet}(B) := \max_{\sigma \in S_w}\{B_{\sigma(1),1} + B_{\sigma(2),2} + \cdots + B_{\sigma(w),w}\}, \tag{5}$$

where we denote the (i, j)-th entry of B as $B_{i,j}$ and S_w represents every permutation of $[w] := \{1, \ldots, w\}$.

Example 1.10. Suppose we have a 3×3 tropical matrix

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}.$$

Then

$$\text{tdet}(B) = B_{21} + B_{12} + B_{33} = 2 + 2 + 2 = 6.$$

Tropical geometric structures. The definitions that follow describe and clarify characteristics of polytopes and hyperplanes in the tropical projective torus. By convention, we focus on tropical geometry defined by a max-plus algebra. However, all geometric structures may be described in terms of min-plus algebra by replacing the max function with the min function. As stated in Remark 1.3, the min-plus algebra and max-plus algebra are related by the classical multiplication by (-1).

Definition 1.11 (tropical line segment from [21]). Given two points u, v, a tropical line segment between u, v denoted as $\Gamma_{u,v}$, consists of the concatenation of at most $e-1$ Euclidean line segments. A point in the collection of points $\boldsymbol{b}$, defining the end points of each Euclidean line segment is called a *bend point* of $\Gamma_{u,v}$. Including u and v, $\Gamma_{u,v}$ consists of at most e bend points. We show how to compute the set $\boldsymbol{b}$ in (6).

Example 1.12. Suppose $u = (0, 0, 0)$ and $v = (0, 3, 1)$. Then the tropical line segment $\Gamma_{u,v}$ consists of the concatenation of two line segments: one from $u = (0, 0, 0)$ to $(0, 2, 0)$ and the other one from $(0, 2, 0)$ to $(0, 3, 1)$.

Maclagan and Sturmfels show in [21] how to construct a tropical line segment between two vectors in the proof of their Proposition 5.2.5. For a pair of vectors $u =: (u_1, \ldots, u_e)$, $v := (v_1, \ldots, v_e) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$, the tropical line segment can be constructed as follows: Without loss of generality, assume that $(v_1 - u_1) \geq \cdots \geq (v_{e-1} - u_{e-1}) \geq (v_e - u_e) = 0$ over the tropical projective torus $\mathbb{R}^e/\mathbb{R}\mathbf{1}$ once the coordinates of $v - u$ have been permuted. The tropical line segment $\Gamma_{u,v}$ from v to u is given by $(d-1)$ connected line segments with breakpoints such that

$$\begin{aligned}
(v_e - u_e) \odot u \oplus v &= v \\
(v_{e-1} - u_{e-1}) \odot u \oplus v &= (v_1, v_2, v_3, \ldots, v_{e-1}, v_{e-1} - u_{e-1} + u_e) \\
&\vdots \\
(v_2 - u_2) \odot u \oplus v &= (v_1, v_2, v_2 - u_2 + u_3, \ldots, v_2 - u_2 + u_e) \\
(v_1 - u_1) \odot u \oplus v &= u.
\end{aligned} \tag{6}$$

Definition 1.13 (tropical polytopes [13]). Suppose we have $S \subset \mathbb{R}^e/\mathbb{R}\mathbf{1}$. If

$$x \odot v \oplus y \odot w \in S$$

for any x, $y \in \mathbb{R}$ and for any v, $w \in S$, then S is called *tropically convex*. Suppose $V = \{v^1, \ldots, v^s\} \subset \mathbb{R}^e/\mathbb{R}\mathbf{1}$. The smallest tropically convex subset containing V is called the *tropical convex hull* or *tropical polytope* of V which can be written as the set of all tropical linear combinations of V

$$\operatorname{tconv}(V) = \{a_1 \odot v^1 \oplus a_2 \odot v^2 \oplus \cdots \oplus a_s \odot v^s \mid a_1, \ldots, a_s \in \mathbb{R}\}.$$

The smallest subset $V' \subseteq V$ such that

$$\operatorname{tconv}(V') = \operatorname{tconv}(V)$$

is called a minimum, or generating set, with $|V'|$ being the cardinality of V'. For $P = \operatorname{tconv}(V')$ the boundary of P is denoted ∂P.

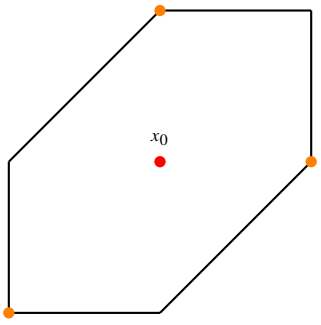

Figure 1. Tropical ball, $B_l(x_0) \in \mathbb{R}^3/\mathbb{R}\mathbf{1}$ with radius l. Center point is indicated in red with the generating set V' shown in orange [3].

Remark 1.14. A tropical polytope of two points $u, v \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ is called a tropical line segment, denoted $\Gamma_{u,v}$.

We use the term *max-tropical polytope* if a tropical polytope is defined in terms of the max-plus algebra. Conversely, we use the term *min-tropical polytope* if a tropical polytope is defined in terms of the min-plus algebra. When the context is clear we use the term tropical polytope.

Definition 1.15 (polytropes [14]). A classically convex tropical polytope is called a *polytrope*.

Definition 1.16 (tropical simplex). A tropical simplex is a tropical polytope that possesses a minimum vertex, or generating, set V' such that $|V'| = e$. A tropical simplex is denoted P_Δ. All polytropes are tropical simplices. The converse is not true.

An important type of polytrope is a *tropical ball*. A tropical ball is the analogue of a Euclidean ball which is defined as

$$B_r(x) = \{y \in \mathbb{R}^e \mid \|x - y\|_2 \leq r\}$$

and indicates the set of all points falling within a distance r of a point x where distance is calculated by the L_2-norm. In the case of a tropical ball, distance is defined in terms of the tropical metric shown in Definition 1.7.

Definition 1.17 (tropical ball). A tropical ball, $B_l(x_0)$, around $x_0 \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ with a radius $l > 0$ is defined as follows:

$$B_l(x_0) = \{y \in \mathbb{R}^e/\mathbb{R}\mathbf{1} \mid d_{\text{tr}}(x_0, y) \leq l\}.$$

The minimum generating set V' of a tropical ball consists of exactly e vertices in which case a tropical ball is a tropical simplex [8, Section 2]. Figure 1 provides the generic structure of a tropical ball.

In many situations, we are interested in the projection of a point onto a tropical polytope. This projection can be represented by using Formula 5.2.3 in [21] and is shown below.

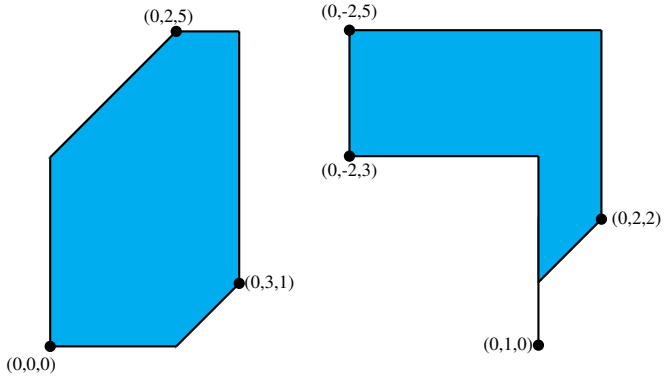

Figure 2. Tropical polytopes in $\mathbb{R}^3/\mathbb{R}\mathbf{1}$. The tropical polytope on the left is a polytrope and therefore a tropical simplex [3].

Proposition 1.18 (tropical projection [21, Formula 5.2.3]). *Let* $V := \{v^1, \ldots, v^s\} \subset \mathbb{R}^e/\mathbb{R}\mathbf{1}$ *and let* $\mathcal{P} = \text{tconv}(v^1, \ldots, v^s) \subseteq \mathbb{R}^e/\mathbb{R}\mathbf{1}$ *be a tropical polytope with its vertex set* V. *For* $\boldsymbol{x} \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$, *let*

$$\pi_{\mathcal{P}}(\boldsymbol{x}) := \bigoplus_{l=1}^{s} \lambda_l \odot v^l, \quad \text{where } \lambda_l = \min\{\boldsymbol{x} - v^l\}. \tag{7}$$

Then

$$d_{\text{tr}}(\boldsymbol{x}, \pi_{\mathcal{P}}(\boldsymbol{x})) \leq d_{\text{tr}}(\boldsymbol{x}, \boldsymbol{y})$$

for all $\boldsymbol{y} \in \mathcal{P}$.

Now we turn our attention to definitions which exhibit the relationship between min-tropical hyperplanes and max-tropical polytopes.

Definition 1.19 (tropical hyperplane [30]). For any $\omega := (\omega_1, \ldots, \omega_e) \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$, the max-tropical hyperplane defined by ω, denoted as $H_\omega^{\max}$, is the set of points $x \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ such that

$$\max_{i\in[e]}\{\omega_i + x_i\} \tag{8}$$

is attained at least twice, i.e., the function is not linear. Similarly, a min-tropical hyperplane denoted as $H_\omega^{\min}$, is the set of points $x \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ such that

$$\min_{i\in[e]}\{\omega_i + x_i\} \tag{9}$$

is attained at least twice. If it is clear from context, we simply denote H_ω as a tropical hyperplane in terms of the min-plus or max-plus algebra where ω is the *normal vector* of H_ω. The point $-\omega$ represents a point contained in H_ω where the maximum or minimum is attained e times. This point is called the *apex* of H_ω.

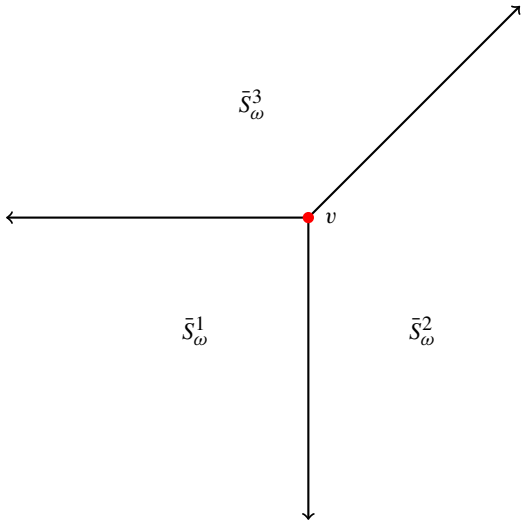

Figure 3. Generic tropical hyperplane $H^{\max}_\omega$ in $\mathbb{R}^3/\mathbb{R}\mathbf{1}$ with associated closed sectors. The red point represents the apex, v with $\omega = -v$ [3].

Definition 1.20 (sectors from Section 5.2 in [13]). Every tropical hyperplane, H_ω, divides the tropical projective torus, $\mathbb{R}^e/\mathbb{R}\mathbf{1}$ into e connected components, which are *open sectors*

$$S^i_\omega := \{x \in \mathbb{R}^e/\mathbb{R}\mathbf{1} \mid \omega_i + x_i > \omega_j + x_j, \forall j \neq i\}, i = [e].$$

These *closed sectors* are defined as

$$\overline{S}^i_\omega := \{x \in \mathbb{R}^e/\mathbb{R}\mathbf{1} \mid \omega_i + x_i \geq \omega_j + x_j, \forall j \neq i\}, i = [e].$$

Lemma 1.21 (distance to a tropical hyperplane H_ω [10]). *The tropical distance from a point $v \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ to a max-tropical hyperplane $H^{\max}_0$ where ω represents the normal vector of zeros, is given by the difference between the maximum and second maximum of v. That is*

$$d_{\text{tr}}(H_0, v) = \max(v) - 2nd\max(v). \tag{10}$$

For a min-tropical hyperplane $H^{\min}_0$, the tropical distance from a point v is

$$d_{\text{tr}}(H_0, v) = 2nd\min(v) - \min(v). \tag{11}$$

For a generic tropical hyperplane H_ω,

$$d_{\text{tr}}(H_\omega, v) = d_{\text{tr}}(H_0, v + \omega). \tag{12}$$

Figure 3 illustrates the construction of $H^{\max}_\omega$ with apex at $v = (0, 4, 7)$ and $\omega = -v$. Additionally, we observe the point $u = (0, 3, 1)$ in relation to $H^{\max}_\omega$ along with each sector $\bar{S}^i_\omega$ as shown in Definition 1.20.

Definition 1.22 (tropical hyperplane arrangements). For a given set of points, $V = \{v^1, \ldots, v^s\}$, tropical hyperplanes with apices at each $v^i \in V$ represent the *tropical hyperplane arrangement* of V, $\mathcal{A}(V)$, where

$$\mathcal{A}(V) := \{H_{-v^1}, \ldots, H_{-v^s}\}.$$

If we consider a collection of tropical hyperplanes defined in terms of the max-plus algebra, we call this arrangement a *max-tropical hyperplane arrangement* denoted $\mathcal{A}^{\max}(V)$. Likewise, considering tropical hyperplanes defined in terms of the min-plus algebra is called a *min-tropical hyperplane arrangement* denoted $\mathcal{A}^{\min}(V)$.

Definition 1.23 (cells). For a given hyperplane arrangement, $\mathcal{A}(V)$, a *cell* is defined as the intersection of a finite number of closed sectors. Cells may be *bounded* or *unbounded*. Bounded cells are polytropes.

Definition 1.24 (bounded subcomplex [9]). For a vertex set, V, $\mathcal{A}(V)$ defines a collection of bounded and unbounded cells which is known as a *cell decomposition*. The union of bounded cells defines the *bounded subcomplex*, $\mathcal{K}(V)$.

Theorem 1.25 [13, Corollary 6.17]. *A* max*-tropical polytope*, P, *is the union of cells in* $\mathcal{K}(V)$ *of the* ***cell decomposition*** *of the tropical projective torus induced by* $\mathcal{A}^{\min}(V)$.

Theorem 1.25 describes $\mathcal{K}(V)$ as a collection of bounded cells induced by some $\mathcal{A}(V)$. Figure 4 represents a tropical polytope in terms of its vertex set (left) and hyperplane arrangement (right).

Throughout this paper we are interested in sampling the union of $(e-1)$-dimensional polytropes belonging to $\mathcal{K}(V)$. The union of $(e-1)$-dimensional polytropes is described in the following definition.

Definition 1.26 (dimension of a tropical polytope). The dimension of a tropical polytope, $P \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$, is defined by the bounded cell of maximal dimension in $\mathcal{C}_P$ and is denoted as $\dim(P)$.

Definition 1.27 (*i-trunk* and *i-tentacles* [20, Definition 2.1]). Let P be a tropical polytope and let $i \in [e-1] := \{1, \ldots, e-1\}$. Let $\mathcal{F}_P$ be the set of relatively open tropical polytopes in $\mathcal{C}_P$. For any $T \in \mathcal{F}_P$, T is called an *i-tentacle element* of $\mathcal{F}_P$ if it is not contained in the closure of any $(i+1)$-dimensional tropical polytope in $\mathcal{F}_P$ where the dimension of T less than or equal to i. The *i-trunk* of P, is defined as

$$\mathrm{Tr}_i(P) := \bigcup \{F \in \mathcal{F}_P : \exists G \in \mathcal{F}_P \text{ with } \dim(G) \geq i \text{ such that } F \subseteq G\}$$

where $\dim(G)$ is the dimension of $G \subset \mathcal{F}_P$. The $\mathrm{Tr}_i(P)$ represents the portion of the $\mathcal{K}(V)$ with $(i-1)$-tentacles removed. The minimum enclosing ball containing only $\mathrm{Tr}_i(P) \subseteq P$ is denoted $B_k(\mathrm{Tr}_i(P))$.

Example 1.28. Consider the tropical polytope, $P = \{(0,0,0), (0,-1,1), (0,2,2), (0,1,-1)\}$. The $\mathrm{Tr}_2(P)$ is the gray portion shown in Figure 5.

For many statistical problems, a first step in statistical inference is finding a center of mass for some given data. This is no less true when handling data in the tropical projective torus. To that end, we concern ourselves with finding the Fermat–Weber point in the tropical projective torus, or simply the tropical Fermat–Weber point.

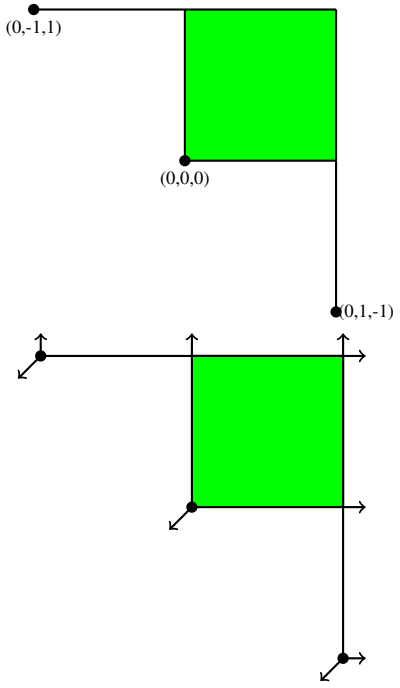

Figure 4. Tropical polytope expressed in terms of its vertex set V (left) and $\mathcal{A}^{\min}(V)$ (right). In the right figure, pseudovertices not belonging to V are defined by the intersection of min-tropical hyperplanes with apices at each $v^i \in V$ [3].

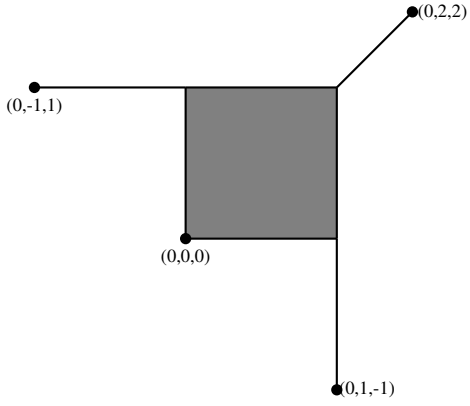

Figure 5. A tropical polytope, P, in $\mathbb{R}^3/\mathbb{R}\mathbf{1}$ defined by four vertices. The $\mathrm{Tr}_2(P)$ is the portion in gray [6].

Definition 1.29 (tropical Fermat–Weber points [19]). For a given set of points $V = \{v^1, \ldots, v^s\}$ in a metric space $\mathcal{M}$, the Fermat–Weber point for the set V is

$$\arg\min_y \sum_{i=1}^{s} d(y, v_i), \tag{13}$$

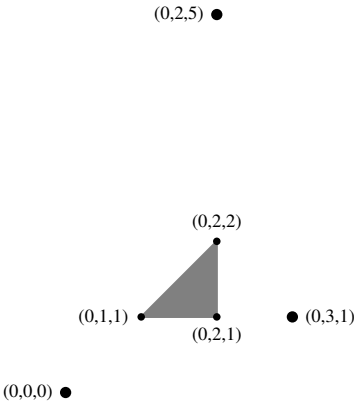

Figure 6. Fermat–Weber region defined by three points. Points in the gray triangle satisfies (14) [4; 19].

where $d(\cdot)$ represents the function defining a distance metric in $\mathcal{M}$ and the point $y \in \mathcal{M}$ represents a center of mass. A *tropical Fermat–Weber point* is similarly defined but replacing $d(\cdot)$ with the tropical metric $d_{\text{tr}}(\cdot)$

$$\arg\min_{y} \sum_{i=1}^{s} d_{\text{tr}}(y, v_i). \tag{14}$$

Remark 1.30. The tropical Fermat–Weber point is not guaranteed to be unique; see [19].

In this paper we introduce a number of ways to obtain a tropical FW point and then apply the resulting point, or center of mass, to problems of statistical inference.

1D. ***Tropical geometry and phylogenetic trees.*** One area of interest where tropical geometry is directly applicable is in the biological science of phylogenomics. Phylogenomics is a discipline focusing on reconstructing the evolutionary history of organisms. One method of representing this evolutionary history is through the use of phylogenetic trees. Phylogenetic trees are weighted unrooted or rooted trees whose internal nodes do not have labels but external nodes have labels. They serve as data structures representing the evolution of genes from related, and usually present-day, species or some other taxa. In terms of graph theory, phylogenetic trees represent rooted out-trees consisting of a root node, unlabeled internal nodes, and external leaf nodes. The root node of the tree represents a common evolutionary ancestor among several taxa, internal nodes represent speciation events over time, and the external, leaf, nodes represent the present-day taxa.

Example 1.31. Figure 7 shows an example of a rooted phylogenetic tree with four leaves $\{a, b, c, d\}$.

Of particular interest, as it relates to tropical geometry, are those phylogenetic trees that are *equidistant* trees. Equidistant trees are unrooted phylogenetic trees where the distance from the root node to each leaf node is the same. It is shown in [7] that equidistant trees can be represented as ultrametrics which are described in Definition 1.33 and illustrated in Example 1.34.

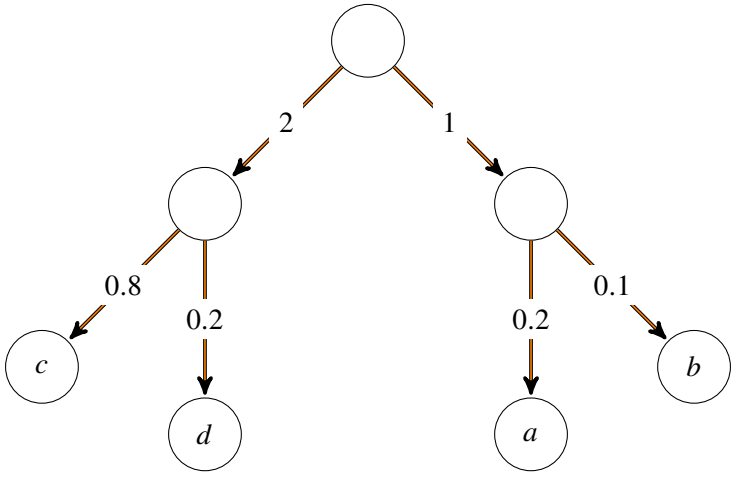

Figure 7. Example for a rooted phylogenetic tree of four leaves with labels $\{a, b, c, d\}$. Each number in each edge represents a weight on the edge which represents an evolutionary time and mutation rates.

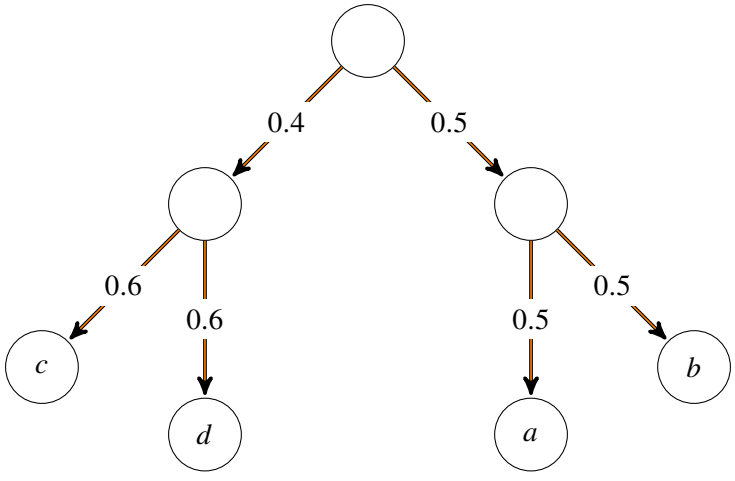

Figure 8. Example for an equidistant tree of four leaves with labels $\{a, b, c, d\}$. The total weights on the path from its root to each leaf is 1.

Example 1.32. Figure 8 shows an example of an equidistant tree with four leaves $\{a, b, c, d\}$. Note that the total weights on the path from its root to each leaf is 1 for all leaves.

Definition 1.33 (ultrametric). Let $[m] := \{1, \ldots, m\}$ and define the distance function $d : [m] \times [m] \to \mathbb{R}$ to be a metric over $[m]$. Then if

$$\max\{d(i, j), d(i, k), d(j, k)\}$$

is attained at least twice, i.e., is not unique, for any $i, j, k \in [m]$, d is an *ultrametric*.

Example 1.34. Suppose $m = 3$. Let d be a metric on $[m] := \{1, 2, 3\}$ such that

$$d(1, 2) = 2, d(1, 3) = 2, d(2, 3) = 1.$$

Since the maximum is achieved twice, d is an ultrametric.

Specifically, for any equidistant tree $T \in \mathcal{U}_m$ where $\mathcal{U}_m$ represents the space of ultrametrics on m leaves, the vector representing the pairwise distances between leaf nodes is an ultrametric. Further, it was shown in [35] that $\mathcal{U}_m$ is a tropical Grassmanian and is therefore tropically convex. This affords us the opportunity to apply statistical methods based on tropical geometry to the space of equidistant phylogenetic trees. We leverage this characteristic of phylogenetic trees throughout this paper by applying the methods of the **TML** package to equidistant phylogenetic trees.

The coalescent model. One way to evaluate how well supervised and unsupervised learning techniques perform when empirical data is unavailable is by using simulated data sets. One method to obtain simulated data is by using the *coalescent model*. The coalescent model was first described in [17] and, for a given sample of taxa, represents a stochastic genealogical process illustrating coalescing events which shows the ancestral lineage of those related taxa. The coalescent model uses species tree to represent the overall lineage of related species with phylogenetic trees showing the lineage of specific genes of the related species in the species tree. We can think of phylogenetic trees emanating from a species tree [28]. For a thorough treatment of the coalescent model, see [32]. Using the coalescent model we can simulate samples of phylogenetic trees that come from specific species trees.

The coalescent is a common model used in a wide range of software. For the simulated data used in this article and included in the **TML** package, we employ the software called Mesquite [22]. Mesquite takes the arguments of species depth, SD, and effective population, denoted N_e. The SD indicates the number of epochs between the common ancestor (root node) and taxa of the present day (leaf nodes). We define the gene trees in terms of the ratio

$$R = \frac{SD}{N_e}.$$

If we set $N_e = 100000$ and $SD = 500000$, then we have $R = 5$. Notably, the Mesquite software does not represent equidistant trees as ultrametrics. However, the output of equidistant trees taken from Mesquite can be manipulated into ultrametric form using tools from the **phytools** `R` package [29]. A total of twelve data sets, each representing 1000 simulated phylogenetic trees on ten leaves with R taking the values of 0.25, 0.5, 1, 2, 5, and 10, coming from two different species trees were constructed from Mesquite are included in the **TML** package. Using **phytools**, these twelve data sets were manipulated into ultrametrics and are included as `Sim_Trees1` for the six datasets coming from species tree one and `Sim_Trees2` from those coming from species tree two.

2. BASIC OPERATIONS.

This section describes the basic functionality of **TML** and how to execute some simple computations.

2A. ***Loading TML and basic operations.*** The **TML** package is loaded as all R packages are loaded:

```
R> library(''TML'')
```

Once **TML** is loaded all functionality is available. We note that except in specific cases, all functions and examples are based on linear algebraic operations in terms of this max-plus algebra. However, all basic functions and some applications may also be employed in terms of min-plus algebra. This is accomplished by assigning the built-in `max()` or `min()` to the `tadd` (tropical addition) argument. The default in each case is `tadd=max`. By assigning `tadd=min` in the associated function, the function is executed in terms of min-plus algebra.

Most inputs into functions in **TML** involve vectors or matrices. For example if we wish to find the tropical distance between two points in $\mathbb{R}^3/\mathbb{R}\mathbf{1}$ we input the following:

```
R> u <- c(0,1,2)
R> v <- c(0,4,7)
R> trop.dist(u,v)

[1] 5
```

As is shown in (1), adding a scalar c to each element of the vector in $\mathbb{R}^e/\mathbb{R}\mathbf{1}$ represents the same vector. This means that for a given vector, we can normalize the vector by adding the additive inverse of the value of any element of the vector. For example:

```
R> u <- c(2,3,4)
R> normaliz.vector(u)

[1] 0 1 2
```

In practice we can choose any element of the vector but by convention **TML** forces the scalar value in the first position to be zero. As we observe, if we instead have unnormalized vectors we still obtain the same tropical distance between points obviating the need to normalize vectors prior to using the function.

```
R> v <- c(3,7,10)
R> trop.dist(u,v)

[1] 5
```

We can also apply a normalizing operation to a set of vectors. In general, we consider a set of vectors as defining a tropical polytope where each row vector consists of a point in the polytope. Importantly, the collection of points need not be the minimum generating set of the tropical polytope. The code snippet below illustrates the use of the `normaliz.polytope()` function on a set of vectors that defines a tropical polytope. We can say that both matrices represent the same polytope by using the fact that $v = v + c\mathbf{1}$, for any $v \in \mathbb{R}^d/\mathbb{R}\mathbf{1}$ and any $c \in \mathbb{R}$, in equation (1).

```
R> (P<-rbind(c(3,3,3),c(4,6,9),c(2,5,3)))

    [,1] [,2] [,3]
[1,]    3    3    3
[2,]    4    6    9
[3,]    2    5    3

R> (P_normalized <- normaliz.polytope(P))

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    2    5
[3,]    0    3    1
```

As with classical linear algebra we can find the tropical analogue of the determinant of a matrix as shown in equation (5). This is equivalent to a linear assignment problem optimization problem as discussed in [13]. In the example that follows, we intentionally show setting `tadd=max` argument despite the fact that this argument defaults to `max`. Throughout the rest of the paper we ignore this argument for max-plus algebraic functions. Again, the input is a matrix where rows are the points in $\mathbb{R}/\mathbb{R}\mathbf{1}$.

```
R> tdets(P_normalized, tadd=max)

[[1]]
[1] 8

[[2]]
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    3    1
[3,]    0    2    5
```

This output comes in the form of a list where the first element represents the value of the tropical determinant and the second element is a matrix of the same points reordered such that the elements of each row vector contributing to the value of the determinant are on the diagonal.

It is common to work with phylogenetic trees, but the statistical methods of tropical geometry use vectorized input. A vector representing a phylogenetic tree consists of components expressing the pairwise distances between the m leaves as the sum of branch lengths connecting those leaves. Consequently, the vector has dimension $e = \binom{m}{2}$. The following example illustrates how a tree with 4 leaves is converted to a vector in $\mathbb{R}^6$ using the function `tree.to.vector()`. The names of the components of the output vector correspond to pairs of leaves. There is also another function (`vector.to.equidistant.tree()`)

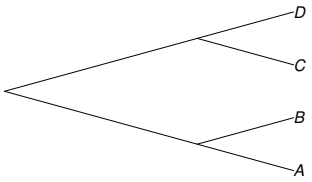

Figure 9. Plot of the phylogenetic tree resulting from the previous code block.

that takes the vector of pairwise distances and converts them back to an equidistance tree. This is done by applying hierarchical clustering to the distance vector. The output of this function is always an equidistant tree, even when the vector is not ultrametric. Therefore, the functions `tree.to.vector()` and `vector.to.equidistant.tree()` are inverse only in the space of equidistant trees as shown in the code chunk below. A nonequidistant tree can be converted to an approximate equidistant tree by using the composite function `vector.to.equidistant.tree` ∘ `tree.to.vector`. For equidistant trees, the composite function returns the original equidistant tree as illustrated in the example below.

In the code chunk below, `read.tree()` calls the function `read.tree()` after loading the **ape** package. Then by using a piping function, `|>` we apply several sequential functions eventually giving a plot of the phylogenetic tree as shown in Figure 9.

```
R> library(ape)
R> tree <- read.tree(text='((A:1, B:1):2, (C:1, D:1):2);')
R> (vector <- tree.to.vector(tree,normalization = FALSE))
 "A-B A-C A-D B-C B-D C-D
  2   6   6   6   6   2"
R> tree |>
  tree.to.vector(normalization = FALSE) |>
  vector.to.equidistant.tree() |>
  plot(type = "c")
R> tree_from_vector <- vector.to.equidistant.tree(vector)
R> all.equal(tree,tree_from_vector)
[1] TRUE
```

The final line in the code block illustrates that we obtain the original tree using the function `vector.to.equidistant.tree()`.

2B. ***Tropical line segments, hyperplanes, polytopes, and projections.*** This section focuses on the functions in **TML** related to tropical geometric structures such as tropical line segments, polytopes, hyperplanes, and projections with associated computations. In each case we can swap to min-plus geometry through use of the `add` argument.

Many of these functions involve constructing visualizations of these geometric structures. The **TML** package relies on the **rgl** package, which provides methods for three-dimensional visualization using OpenGL [23; 43]. However, if OpenGL is not available, loading the **rgl** package may fail. If **rgl** fails to load, the visualization functions in **TML** may fail. To fix this problem the user should reference **rgl** at https://cran.r-project.org/web/packages/rgl/readme/README.html.

Using equation (6) we can construct the tropical line segment by calculating the associated bend points.

```
R> u <- c(0,1,2)
R> v <- c(0,4,7)
R> TLineSeg(u,v)

[[1]]
[1] 0 4 7

[[2]]
[1] 3 4 7

[[3]]
[1] 5 6 7
```

The previous example represents a line segment from the vector $v = (0, 4, 7)$ to $u = (0, 1, 2)$. Note that in this case the second and third bend points are not normalized (i.e., the first value of each vector equal to zero). This can easily be accomplished using the `lapply()` function in conjunction with the `normaliz.vector()` function from **TML**.

```
R> TLineSeg(u,v,tadd=max) |> lapply(normaliz.vector)

[[1]]
[1] 0 4 7

[[2]]
[1] 0 1 4

[[3]]
[1] 0 1 2
```

It should be noted also that for two vectors in $\mathbb{R}^e/\mathbb{R}\mathbf{1}$ the `TLineSeg()` function output list consists of e bend points, but some bend points may be redundant. This is an indication that the line segment consists of fewer than e bend points.

Tropical hyperplanes, as defined in equation (8) are defined simply by the point in the tropical projective torus that serves as the hyperplane apex. The **TML** package provides functions to visualize tropical

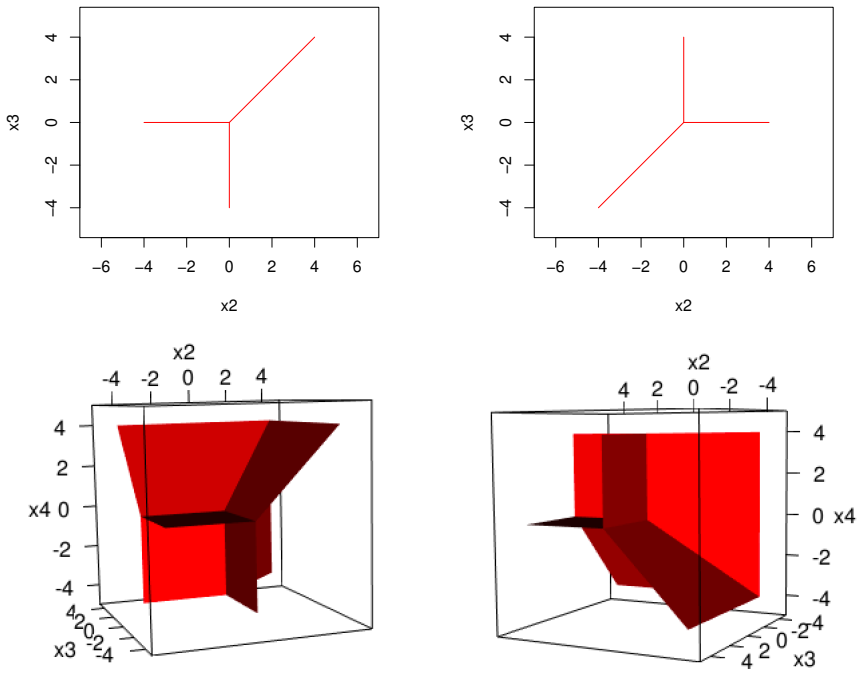

Figure 10. Max-tropical (left) and min-tropical hyperplanes (right) in two and three dimensions using the `hyper3d()` function.

hyperplanes in two and three dimensions as well as functions to measure the tropical distance from a point in the tropical projective torus to the nearest point on the hyperplane. The argument `ext` is a scalar value that defines are far the hyperplane should be extended. The `min.ax` and `max.ax` define the minimum and maximum limits of the axes of the plot.

```
R> D <- c(0,0,0); E <- c(0,0,0,0)
R> ext <- 4
R> max.ax <- -5
R> min.ax <- 5
R> draw.thyper(D, ext, min.ax, max.ax, plot=TRUE)
R> draw.thyper(D, ext, min.ax, max.ax, plot=TRUE, tadd=min)
R> draw.thyper(E, ext, min.ax, max.ax, plot=TRUE)
R> draw.thyper(E, ext, min.ax, max.ax, plot=TRUE, tadd=min)
```

Visualizations from the output of the previous code is shown in Figure 10. Max-tropical hyperplanes in both two dimensions and three dimensions are shown on the left with min-tropical hyperplanes shown on the right.

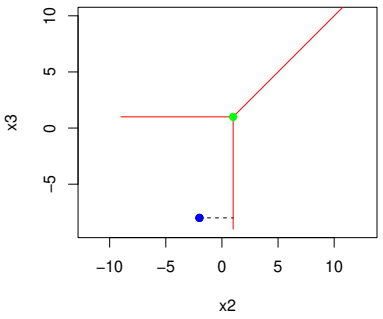

Figure 11. Tropical distance from a point $(0, -2, -8)$ (blue) to the max-tropical hyperplane defined by the normal vector $\omega = (0, -1, -1)$. Note that ω corresponds with the apex of the tropical hyperplane $(0, 1, 1)$ (green).

The statistical methods introduced in this article utilize the tropical distance from a point to a tropical hyperplane. The distance from a point to a tropical hyperplane is shown in Lemma 1.21 for both the max-tropical and min-tropical case. The function `trop.hyper.dist()` takes three inputs: the normal vector associated with the apex of the tropical hyperplane, any generic point in the tropical projective torus, and assigning `tadd=max` or `tadd=min` indicating whether to use max- or min-plus algebra.

```
"this is new"
R> O <- c(0,-1,-1)
R> x0 <- c(0,-2,-8)
R> trop.hyper.dist(O,x0)

[1] 3

R> trop.hyper.dist(O,x0,tadd=min)

[1] 6
```

Recall from Definition 1.19 the normal vector is the same as changing the sign of the point in the tropical projective torus representing the apex of the tropical hyperplane. Figure 11 shows a point on the tropical hyperplane that is closest to the point $(0, -2, -8)$. The tropical distance to this point is shown in the code above. Notably, unlike using a Euclidean distance, the point on the tropical hyperplane that is closest to the point of interest may not be unique, but for points drawn from continuous distributions the projections will be with probability 1.

We now turn our attention to tropical polytopes as tropical polytopes are the primary geometric structures that are used in most functions in the **TML** package. The primary focus here is to illustrate

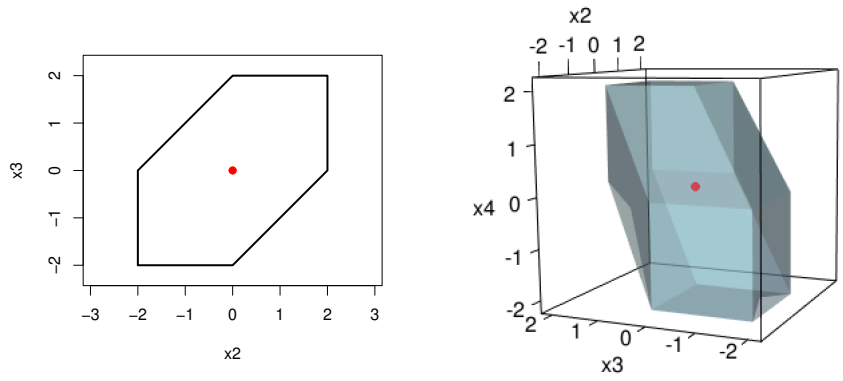

Figure 12. Tropical balls in two and three dimensions each with a radius of two, using the `Trop_ball()` function. The red point in each figure represents the center point.

how we visualize different tropical polytopes. Visualization is combinatorially challenging even in lower dimensions. Here we show a couple of examples of visualizations of two-dimensional and three-dimensional tropical polytopes. As stated in Section 1C, a tropical ball is an important polytrope in tropical geometry. The function `Trop_ball()` allows us to render a two- or three-dimensional tropical ball.

```
R> d <- 2
R> Trop_ball(D, d, a=1, cls='white', fil=TRUE, cent.col='red')
R> Trop_ball(E, d, a=.5, cls='lightblue',
      fil=TRUE, cent.col='red')
```

The code above takes several inputs, the first being the center of the tropical ball, the radius of the tropical ball in terms of tropical distance, and several other inputs involving the transparency and color options. Figure 12 provides the output of two tropical balls from the example above in two dimensions (left) and three dimensions (right).

Visualization of generic tropical polytopes can be accomplished in two or three dimensions using the `draw.tpolytope.2d()` and `draw.tpolytope.3d()` functions. These functions take four inputs: a matrix of points in the tropical projective torus, a color argument for the polytope itself, a color argument for the vertices of the polytope, and assigning `tadd=max` or `tadd=min` indicating whether to use max- or min-plus algebra.

```
R> P <- rbind(c(0,-2,2),c(0,-2,5),c(0,2,1),c(0,1,-1))
R> draw.tpolytope.2d(P, "darkgreen", "black")
R> P <- rbind(c(0,0,0,0),c(0,1,2,5),c(0,1,3,1),c(0,2,5,10))
R> draw.tpolytope.3d(P, "darkgreen", "black")
```

Figures 13 and 14 show the output from the previous code. Note that in each case the function draws line segments between each vertex (and more points in the three-dimensional case) in the polytope. This is

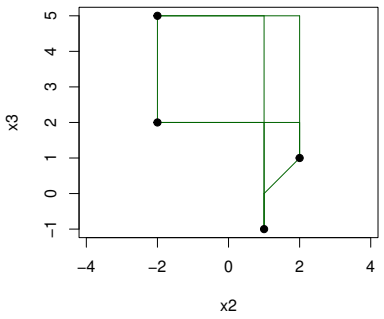

Figure 13. 2-D rendering of a tropical polytope using the `draw.tpolytope.2d()` function. Black points represent the vertices of the tropical polytope.

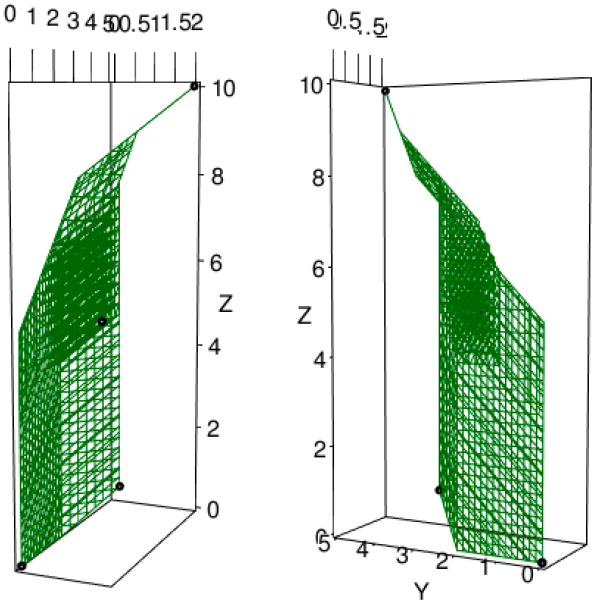

Figure 14. 3-D rendering of a tropical polytope using the `draw.tpolytope.3d()` function. Black points represent the vertices of the tropical polytope.

due to the combinatorial challenge with forming this polytopes. The tropical polytopes themselves are defined by the boundary.

The last specific function we address in this section is the `project.pi()` function. The projection of a point x onto a tropical polytope $\mathcal{P}$ denoted $\pi(x)$ is the point in $\mathcal{P}$ that is closest to x in terms of tropical distance. It must be noted that unlike Euclidean distance, tropical projections are not necessarily unique. In fact oftentimes there is an interval of points satisfying equation (7). For a thorough discussion of projections, see [3].

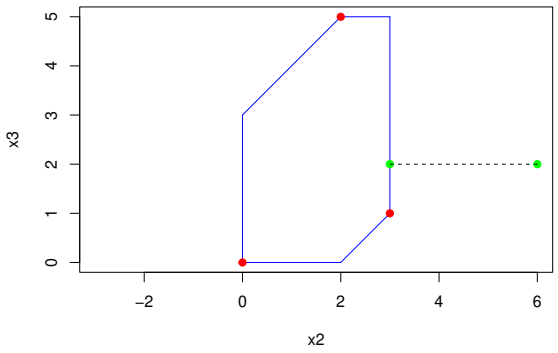

Figure 15. Projection of the point $x = (0, 6, 2)$ onto the tropical polytope $\mathcal{P}$, where $\mathcal{P} := \{(0, 0, 0), (0, 3, 1), (0, 2, 5)\}$. Here $\pi(x) = (0, 3, 2)$ but any point on the boundary of $\mathcal{P}$ from $(0, 3, 2)$ to $(0, 3, 1)$ could serve as $\pi(x)$.

```
R> P <-rbind(c(0,0,0),c(0,2,5),c(0,3,1))
R> x <- c(0,6,2)
R> project.pi(P,x)

[1] 0 3 2
```

The output above provides one of perhaps an infinite number of points that can serve as $\pi(x)$. As in previous functions, `project.pi()` defaults to using max-plus algebra but the addition of `tadd=min` allows for projections in terms of min-plus algebra. Continuing from the previous example, we can illustrate how the projection of a point onto $\mathcal{P}$ appears visually.

```
R> pi_x <- project.pi(P, x)
R> draw.tpolytope.2d(P, "blue", "red")
R> lines(c(x[2], pi_x[2]), c(x[3], pi_x[3]), lty = "dashed")
R> points(c(x[2], pi_x[2]), c(x[3], pi_x[3]), pch = 19,
    col = "green")
```

The functions introduced in this section serve as standalone functions but they also provide the basis of other functions in the **TML** package. In the sections that follow, we introduce the statistical tools that leverage these basic functions.

2C. ***Calculating a tropical Fermat–Weber point.*** In this section we introduce several functions that allow us to calculate tropical Fermat–Weber points. The output of these functions are incorporated in a variety of statistical methods included in the **TML** package, some of which are described below. Here we introduce two methods of finding the tropical FW point. The first is a method that uses linear programming while the second employs a fast gradient-based method.

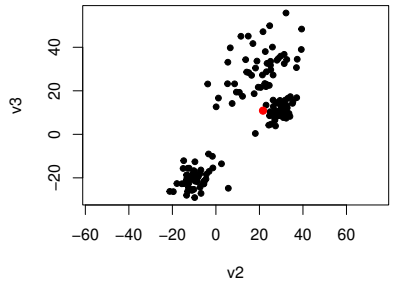

Figure 16. Tropical Fermat–Weber point calculated using the `Trop_FW()` function. Black points indicated the data. The red point is the tropical FW point.

The first method to find a tropical FW point for a given set of data is introduced in the `trop.FW()` function. This function employs a linear programming approach to find a tropical FW point for a given set of data $V = \{v^1, \ldots, v^s\}$ by solving the constrained optimization problem

$$\min_y \sum_{i=1}^{s} d_{\text{tr}}(v^i, y) \tag{15}$$

$$y_j - y_k - v^i_j + v^i_k \leq -d_{\text{tr}}(v^i, y) \qquad \forall i \in [s] \text{ and } 1 \leq j, k \leq e \tag{16}$$

$$y_j - y_k - v^i_j + v^i_k \geq d_{\text{tr}}(v^i, y) \qquad \forall i \in [s] \text{ and } 1 \leq j, k \leq e. \tag{17}$$

The example below shows how this is employed by using simulated data set consisting of 150 normalized points in the tropical projective torus which is found in the **TML** package as `Sim_points`. Note that `trop.FW()` takes as input points which are normalized using the `normaliz.polytope()` function. The `Sim_points` is already normalized so normalization is not required. A plot of the output for the code below is shown in Figure 16.

```
R> set.seed(23)
R> V <- Sim_points
R> FW <- trop.FW(V)
R> plot(V[,2],V[,3],pch=19,cex=.8,xlab="v2",ylab="v3",asp=1)
R> points(FW[2],FW[3],pch=19,col='red')
```

While a FW point can be found directly using linear programming, gradient-based numerical methods like those developed and employed in [2] are much faster. Of particular interest in [2] is inferring the class of species tree associated with a set of gene tree. Because it is proven that the set of tropical FW points asymptotically converges to the maximum likelihood estimate (MLE) vectorized tree under this model this task can be reduced to finding a tropical FW point associated with the set of gene trees. FW points are used in [2] in lieu of MLE trees because the former is faster to compute numerically and it enjoys optimality sufficiency condition.

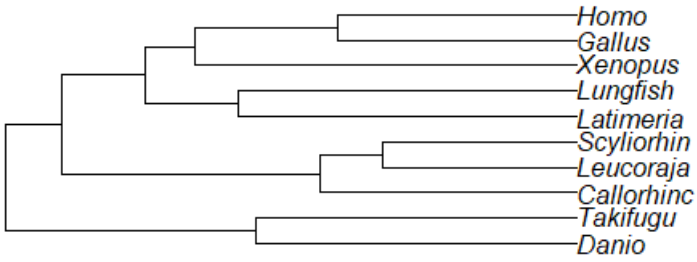

Figure 17. Equidistant tree representation of a Fermat–Weber point of 1290 gene trees from the lungfish data [18].

The following example illustrates how to compute a FW point using the gradient method of 1290 gene trees associated with a lungfish data set used in [18] by employing the function `FWpoint.numerical()`. This data set is accessible in the **TML** package which includes a dissimilarity matrix when `lung_fish` is called. The leaf labels are given by the column names of the `lung_fish` data and are extracted inside the `vector.to.equidistant.tree()` function as shown below.

```
R> lung_fish |>
    FWpoint.numerical() |>
    vector.to.equidistant.tree() |>
    plot()
```

The functions introduced in this section serve as standalone functions but they also provide the basis of other functions in the **TML** package. In the sections that follow, we introduce the statistical tools that leverage these basic functions.

3. TROPICAL HAR AS A MAIN TOOL FOR INFERENCE. Markov chain Monte Carlo (MCMC) methods are an extremely important tool in statistical inference. Since their development in the first half of the twentieth century, MCMC methods have proven effective in a broad spectrum of scientific disciplines. Among the most flexible and easily constructed MCMC methods is the hit-and-run (HAR) sampler. Like all MCMC samplers, HAR samplers sample points according to a target distribution by moving from one point to another by defining a subset of a state space in terms of line segments. Both the current point and the possible next points fall on this line segment [34].

Up until recently, no MCMC samplers existed to sample points from a state space that could be defined as tropically convex. With the introduction of the sampler *vertex HAR with extrapolation* (*VHE*) as shown in [48], tropically convex sets can be sampled more effectively. VHE samples points from tropical line segments where tropical line segments are defined by the projection of the current point onto two tropical polytopes defined by subsets of a vertex set, V, which defines a tropical simplex, $\mathcal{P}$. A single iteration takes as input the current point, x_0 and the vertex set, V. The vertex set is then randomly divided into

two subsets, $V^1, V^2 \subset V$, such that $\text{tconv}(V^i) = \mathcal{P}^i$ and the cardinality of each subset is $|V^1| = e - 1$ and $|V^2| = 1$, respectively. Next, the *extrapolation* step occurs where $\pi_{\mathcal{P}^1}(x_0)$ and $\pi_{\mathcal{P}^2}(x_0)$ are calculated using equation (7). Yoshida et al. in [48] show that x_0 is contained in the tropical line segment defined by $\pi_{\mathcal{P}^1}(x_0)$ and $\pi_{\mathcal{P}^2}(x_0)$ and that this tropical line segment is intrinsic to $\mathcal{P}$. These properties ensure that the Markov property is preserved and that any point sampled from this tropical line segment remains in $\mathcal{P}$. All MCMC methods discussed in this research are variations of VHE. For more detail, see [48].

As most methods in **TML** rely on this novel HAR sampler, we devote this section to providing an in depth examination of its basic implementation as well as a number of variations. Of note, when using MCMC samplers one would typically use different starting points for each iteration. However, for ease of demonstration, we often use the piping operator to collect sampled points in the following examples requiring the starting point to remain the same.

3A. ***HAR sampling from a tropical line segment.*** We begin by showing how we sample from a tropical line segment as the line segment is the basic geometric structure used in HAR sampling. There are two variations we illustrate here. The first shows how we sample points uniformly from a tropical line segment as shown in [48]. This method is implemented using the `HAR.TLineSeg()` function. For any two points in the tropical projective torus, we can define a tropical line segment and then sample from the line segment. The code that follows shows how to sample a single point from a tropical line segment.

```
R> set.seed(1)
R> u <- c(0,3,1)
R> v <- c(0,0,0)
R> e <- length(u)
R> bend_pts <- TLineSeg(u, v) |> lapply(normaliz.vector)
R> G_uv <- do.call("rbind", args = bend_pts)
R> draw.tpolytope.2d(G_uv,'black','black')
R> pt <- HAR.TLineSeg(G_uv[1, ], G_uv[e, ]) |> normaliz.vector()
R> points(pt[2],pt[3],pch=19,col='red')
```

To sample multiple points from a tropical line segment we can employ `HAR.TLineSeg()` in a `for()` loop in the following chunk. Figure 18 shows the building of the line segment, sampling a single point, and then sampling 200 points from the line segment.

```
R> pts <- HAR.TLineSeg(u, v) |>
    normaliz.vector() |> replicate(200, expr = _) |> t()
R> points(pts[, 2], pts[, 3], pch = 19, cex = .75, col='red')
```

The **TML** package also gives the option to sample points from a tropical line segment about a point representing a center of mass as shown in [3]. In practice, this is similar to Gaussian sampling about a point μ with a standard deviation σ_{trop} which controls dispersion in terms of the tropical distance. The `HAR.TLineSeg.centroid()` function performs this calculation. Continuing from the previous example

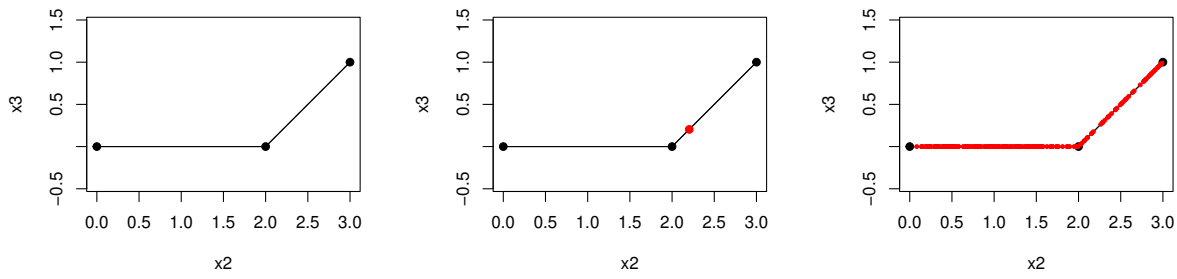

Figure 18. A tropical line segment (top) with the blue points indicating the break points and end points, a single point sampled from the tropical line segment in green (center), and 200 points sampled from the tropical line segment.

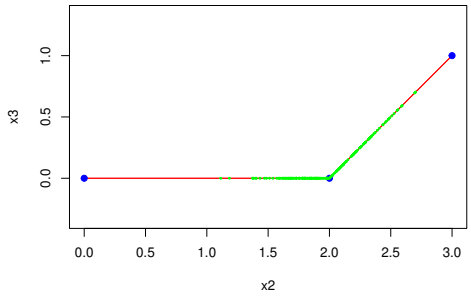

Figure 19. Sampling 200 points (green) about a center of mass $\mu = (0, 2, 0)$ with scale parameter $\sigma_{\text{trop}} = 0.2$.

we show the results of sampling 200 points about a center of mass represented by the point $\mu = (0, 2, 0)$ with a scale parameter $\sigma_{\text{trop}} = 0.2$.

```
R> set.seed(1)
R> m <- c(0, 2, 0); s <- .2
R> pts <- HAR.TLineSeg.centroid(u, v, m, s) |>
    replicate(200, expr = _) |> t()
R> draw.tpolytope.2d(G_uv,'red','blue')
R> points(pts[,2],pts[,3],pch=19,cex=.3,col='green')
```

Comparing the distance of the sampled points from the center of mass μ we can determine the quantiles associated with the tropical distance.

```
R> pts |> apply(1,trop.dist, D1=m) |> quantile()
      0%          25%          50%          75%         100%
0.0002291895 0.0636676116 0.1309480012 0.2323379661 0.7342599864
```

The previous code chunk illustrates quantiles associated with the tropical distance for points sampled about the center of mass, μ, according to a scale parameter $\sigma_{\text{trop}} = 0.2$. As shown in [48], each point on a tropical line segment defined by two points $u, v \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ can be mapped to a scalar value, $\ell \in [0, d_{\text{tr}}(u, v)]$. This means μ corresponds to some scalar $\ell_0 \in [0, d_{\text{tr}}(u, v)]$. In the function `HAR.TLineSeg.centroid()`, scalar values are randomly chosen such that $\ell \sim N(0, \sigma_{\text{trop}})$ distribution. Then, using classical addition, ℓ is added to ℓ_0. In this way, we define the next sampled point y as

$$y = (\ell + \ell_0) \odot u \oplus v$$

as shown in (6). Because ℓ_0 corresponds to the center of mass μ, ℓ_0 remains constant so each sampled point is expressed in terms of the tropical distance from μ. This results in points being sampled according to a location-scale distribution with location parameter μ and scale parameter σ_{trop}. For more detail, see [3].

3B. ***HAR sampling from a tropical polytope.*** Using the line sampling methods described above we can sample from tropical polytopes (see Definition 1.13). This is accomplished using the `VE.HAR()` function. This function allows the user to sample points uniformly from the $(e-1)$-trunk of a tropical simplex (Definition 1.27). Arguments for the `VE.HAR()` function include a matrix defined as the vertices of the tropical simplex, an initial point, and a scalar value indicating the number of intermediate points to sample between the initial state and the final state in the Markov chain. The state in the chain represents the sampled point. Figure 20 shows the results of sampling from a polytrope (top) and a generic tropical simplex (bottom).

```
R> set.seed(1)
R> P <- rbind(c(0, 0, 0), c(0, 2, 5), c(0, 3, 1))
R> x0 <- c(0, 2.5, 3.2)
R> draw.tpolytope.2d(P, "blue", "red")

R> pts <- VE.HAR(P, x0, I = 50) |>
    replicate(1000, expr = _) |> t()
R> points(pts[, 2], pts[, 3], xlab = "x2", ylab = "x3",
    asp = 1, pch = 19, cex = .3)

R> K <- rbind(c(0, -1, 1), c(0, 0, 0), c(0, 1, -1))
R> x0 <- c(0, 0.5, 0.5)
R> draw.tpolytope.2d(K, "blue", "red")

R> pts <- VE.HAR(K, x0, I = 50) |>
    replicate(1000, expr = _) |> t()
R> points(pts[, 2], pts[, 3], xlab = "x2", ylab = "x3",
    asp = 1, pch = 19, cex = .3)
```

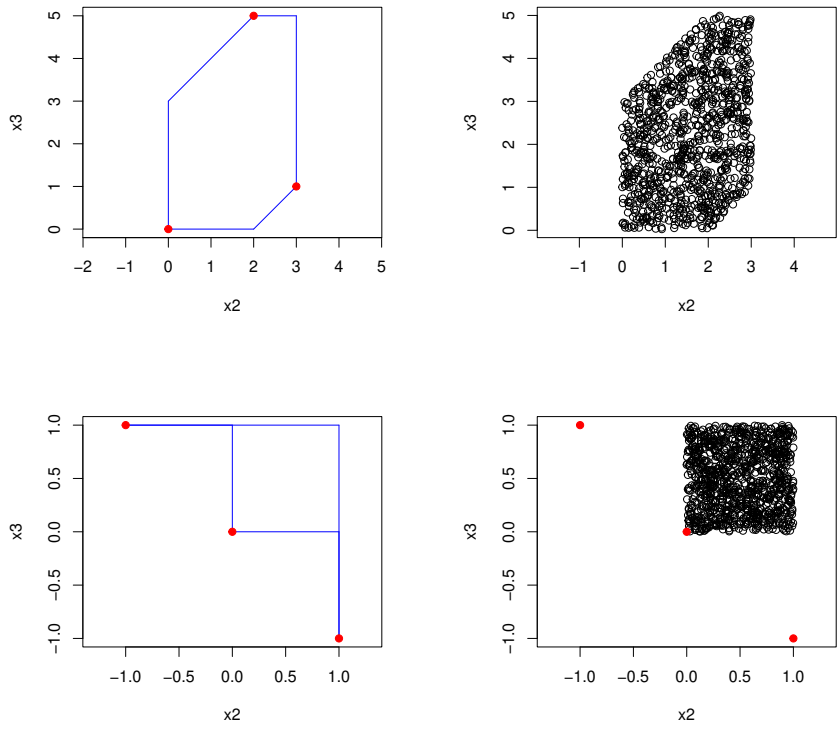

Figure 20. Two tropical polytropes, (left) with vertices indicated in red and results after using `VE.HAR()` to sample 1000 points from each polytope.

In addition, we can also sample points about a center of mass (location parameter, μ) with dispersion controlled by a scale parameter, σ_{trop}, which is defined in terms of the tropical metric. This method is described in detail in [3]. In Euclidean space some HAR methods that sample points according to a Gaussian distribution do so by projecting the center of mass onto the generated line and then sample about the projection according to $N \sim (0, \sigma)$ where σ is a fixed standard deviation. The centroid-based sampler in the **TML** package leverages the mechanics of the basic VHE sampler described previously. However, once the line segment is constructed in the extrapolation step, the point μ representing the location parameter, is projected onto this line segment. However, in the tropical projective torus, the projection of a point onto a line segment is not usually unique. So the algorithm employs a bisection search to approximate the interval of points which provides all valid projections of the location parameter onto the tropical line segment. From this interval, is selected uniformly using the `HAR.TLineSeg()` function. If we do not define and then sample from the entire interval of possible projections, sampling biases towards the projection of μ defined by equation (7). Once selected, this sampled point represents the projection of the location parameter onto the tropical line segment and we sample about this point according to a $N(0, \sigma_{\text{trop}})$ distribution. For detail on how this is accomplished, see Chapter 2 in [3].

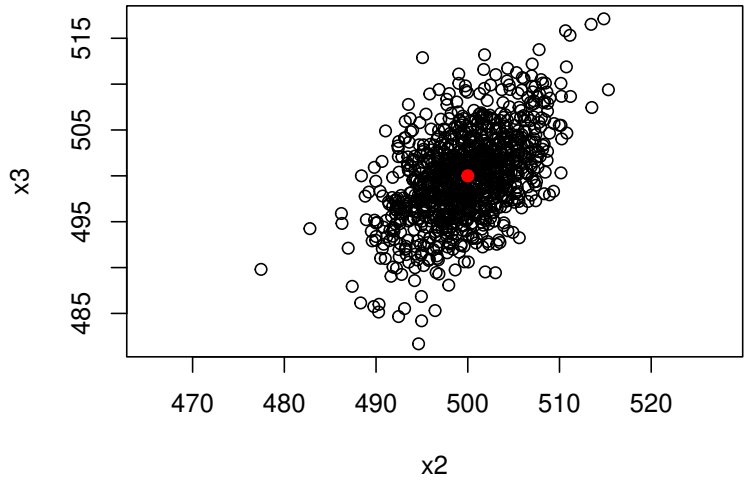

Figure 21. Sampling 1000 points about a center of mass using the `VE.HAR.centroid()` function.

In the **TML** package, the `VE.HAR.centroid()` function executes this method. The function `VE.HAR.centroid()` takes as inputs the vertices of a tropical simplex, a starting point, and a scalar value determining the length of the Markov chain. In addition, a point serving as a center of mass representing the location parameter, and a scale parameter in the form of a scalar to control dispersion are also used. Figure 21 shows an output of sampling about a center of mass.

```
R> P <- rbind(c(0, 0, 0), c(0, 0, 1000), c(0, 1000, 0))
R> x0 <- c(0, 2.5, 3.2)
R> m <- c(0, 500, 500)
R> pts <- VE.HAR.centroid(P, x0, I = 50, m, s = 4) |>
     replicate(1000, expr = _) |> t()

R> plot(pts[, 2], pts[, 3], xlab = "x2", ylab = "x3", asp = 1)
R> points(m[2], m[3], pch = 19, col = "red")
```

The previous example treats the state space as a large, ordinary square, with the center of mass being the center of the square. The intent of the exercise is to show that we can sample points about a center of mass according to a centroid-based distribution which incorporates the tropical metric. The code snippet below only provides a visualization for the scale parameter equal to 0.05 though Figure 22 provides visualization for each scale parameter value. The code block only provides the code for the points sampled using a scale parameter of $\sigma_{\text{trop}} = 0.05$.

```
R> P <- matrix(c(0,0,0,0,2,5,0,3,1),3,3,TRUE)
R> y_star <- c(0,2.1,3.6)
R> m <- c(0,1,1)
R> sds <- c(.05,.1,.5,1)
R> N <- 1000
R> poins <- replicate(4,matrix(NA,0,dim(P)[2],TRUE))
R> for (j in 1:length(poins)){
    y <- y_star
    for (i in 1:N){
        y <- VE.HAR.centroid(P,y,I=50,m,sds[j])
        poins[[j]] <- rbind(poins[[j]],y)
      }
    }
R> draw.tpolytope.2d(P,'lightblue','red',plot=TRUE)
R> points(poins[[1]][,2],poins[[1]][,3],pch=3,cex=.4)
R> points(m[2],m[3],col='orange',pch=19,cex=.4)
```

In Figure 22, we observe that as the scale parameter increases, the sampling biases towards the boundaries because the state space is a polytope and therefore bounded.

The functions in this section illustrate the basics of HAR sampling over the tropical projective torus. In the next section we provide specific data science applications using the methods in the **TML** package.

4. TML FOR TROPICAL STATISTICS. In this section we provide several applications of the methods available in the **TML** package. We begin with an application to show how to estimate the volume of a tropical polytope as described in [6], which is a NP-hard problem [11]. Next, we provide a supervised learning method involving tropical logistic regression applied to classifying phylogenetic trees as introduced in [2]. Then we show how to apply unsupervised learning in the form of tropical principal component analysis, again, applied to phylogenetic trees [26]. Finally, we show how to implement a nonparametric tropical kernel density estimation method as a way to identify outliers related to phylogenetic trees on $[m]$ leaves [47].

4A. ***Application 1: Volume estimation of a tropical polytope.*** A challenging problem in polyhedral geometry is estimating the volume of polytopes and is no less challenging in the tropical setting. In this section, we follow the methods devised and illustrated in [6]. In general, for a given tropical polytope, $\mathcal{P}$, this involves finding a minimum enclosing tropical ball, denoted $B_r(\mathcal{P})$. By sampling from $B_r(\mathcal{P})$, which is of known volume, we can estimate the volume of $\mathcal{P}$ by multiplying the volume of $B_r(\mathcal{P})$ by the proportion of sampled points with membership in $\mathcal{P}$.

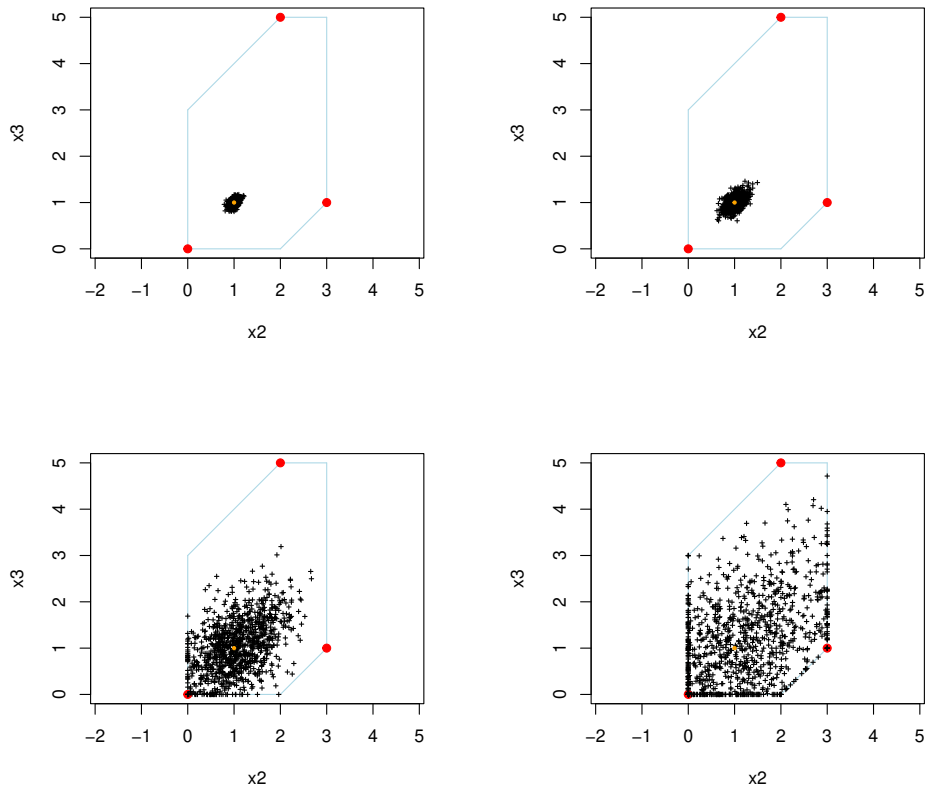

Figure 22. Sampling 1000 points from $\mathcal{P}$, where $\mathcal{P} := \{(0,0,0), (0,3,1), (0,2,5)\}$ about the center of mass $\mu = (0,1,1)$ (in orange) using the `VE.HAR.centroid()` function for four different scale parameters 0.05 (top left), .1 (top right), .5 (bottom left), 1 (bottom right).

This application begins with computing $B_r(\mathcal{P})$ for a given $\mathcal{P}$ using the `min_enc.ball()` function. The output of the function is a two element list with the first element representing the center point of the ball and the second element representing the radius of the tropical ball in terms of tropical distance.

```
set.seed(1)
R> P <- rbind(c(0,0,0),c(0,3,1),c(0,2,5))
R> (B <- P |> min_enc.ball())
$Center
[1] 0.0 2.0 2.5

$Radius
[1] 2.5
```

Next we use the `trop.bal.vert()` to obtain the points in the minimum generating set of $B_r(\mathcal{P})$. The output is a matrix with rows representing the points in the minimum generating set V' of $B_r(\mathcal{P})$.

```
R> (BR <- trop.bal.vert(B[[1]],B[[2]]))

      [,1] [,2] [,3]
[1,]    0  -0.5  0.0
[2,]    0   4.5  2.5
[3,]    0   2.0  5.0
```

Using the output of points from the `trop.bal.vert()` function and the original tropical polytope, $\mathcal{P}$, we can estimate the volume of $\mathcal{P}$. This is accomplished through the use of the `trop.Volume()` function. Inputs include an the matrix representing the tropical points defining the tropical ball, a matrix representing the points defining the original tropical polytope, an initial point, the number of points to sample, a scalar value representing the length of each Markov chain, and the radius, r, of $B_r(\mathcal{P})$.

```
R> x0 <- c(0,1.5,.4); s <- 200; I <- 50; r <- B[[2]]
R> trop.Volume(BR,P,x0,s,I,r)
$Ratio
[1] 0.67

$Vol_Ball
[1] 18.75

$Vol_Poly
[1] 12.5625
```

The output of the `trop.Volume()` function is a list containing three elements. The first element represents the proportion of points falling in the polytope of interest. The second element represents the volume of $B_r(\mathcal{P})$ and the third represents the volume estimate of the tropical polytope.

4B. ***Application 2: Tropical logistic regression.*** We now introduce the supervised learning method of tropical logistic regression as applied to phylogenetics. In [2], tropical logistic regression is introduced and shown to outperform classical logistic regression when applied to phylogenetic trees. Specifically, tropical logistic regression is a binary classification method applied to a given set of phylogenetic trees. The classifications represent membership into one of two species trees.

Next, we turn to using tropical logistic regression to the problem of classifying gene trees according to the species tree that generated them. For this application, we use the two sets of 1000 phylogenetic trees, `Sim_Trees11` and `Sim_Trees21` (or simply $T1$ and $T2$ where the ratio $R = SD/N_e = 1$) where each phylogenetic tree has ten leaves and is represented as an ultrametric. Using $T1$ and $T2$, the task is now to classify unseen gene trees.

The tropical logistic regression model first infers the species tree that generated the corresponding trees of each class. Under the coalescent model which is explained in Section 1D, the species trees and gene trees are equidistant and so the corresponding vectors are ultrametric. However, a Fermat–Weber point, which is used in lieu of the MLE tree, may not be an ultrametric. Ideally, we would like to have an additional constraint, requiring that the species tree be an ultrametric. By adding a regularization term that penalizes deviations from the space of ultrametrics, we instead consider the modified Fermat–Weber point

$$\arg\min_{\omega\in\mathbb{R}^e}\left\{\sum_{i=1}^{N} d_{\mathrm{tr}}(x_i,\omega)+\lambda\|\omega-\pi(\omega)\|^2\right\},$$

where $x_i \in \mathbb{R}^e/\mathbb{R}\mathbf{1}$ is the i-th vectorized gene tree, π is a projection onto the space of ultrametrics and λ is the regularization rate in terms of the l_2-norm. This method is employed in the standalone function `FWpoint.num.w.reg()` which is incorporated in the tropical logistic regression method defined by `trop.logistic.regression()`. However, by default there is no regularization i.e., λ=0. With the default setting, the function is identical to `FWpoint.numerical()`. It has been observed that the AUCs of tropical logistic regression are lower when regularization is applied, but the inferred species trees are closer to the true species trees. Therefore, the modified Fermat–Weber point is more useful for inference than classification. Once the inferred species trees have been computed in `trop.logistic.regression()`, the two scaling parameters are computed by solving an optimization problem using the conjugate gradient method, with the MLE estimate for the scale parameters being used as an initial guess for the optimizer. The following example shows how tropical logistic regression can be employed.

```
R> library("ROCR")
R> D <- rbind(Sim_Trees11, Sim_Trees21)
R> Y <- c(rep(0, dim(Sim_Trees11)[1]),
    rep(1, dim(Sim_Trees21)[1]))
R> N <- length(Y)
R> set.seed(1)
R> train_set <- sample(N, floor(0.8 * N)) ## 80/20 train-test split
R> pars <- trop.logistic.regression(D[train_set, ], Y[train_set],
    penalty = 1e4)
R> test_set <- (1:N)[-train_set]
R> Y.hat <- rep(0, length(test_set))
R> for (i in 1:length(test_set))
    Y.hat[i] <- prob.class(pars, D[test_set[i], ])

R> prediction(Y.hat, Y[test_set]) |>
    performance(measure = "tpr", x.measure = "fpr") |>
    plot(lwd = 2,
    main = "ROC Curve for Logistic Regression Model")
```

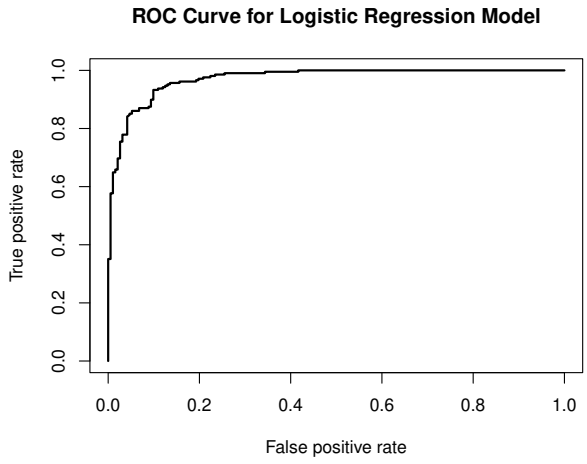

Figure 23. ROC curve produced by the code snippet above.

```
R> prediction(Y.hat, Y[test_set]) |>
    performance(measure = "auc") |>
    slot("y.values")
[[1]]
[1] 0.970966
```

From the example above we see that the area under the curve associated with the receiver operator characteristic (ROC) curve is close to one. This indicates a near-perfect classification for the given example. Figure 23 provides a visual representation of the ROC curve associated with the example above.

4C. ***Application 3: Tropical PCA.*** Principal component analysis (PCA) is an unsupervised learning technique used for dimension reduction. Tropical PCA is no different but focuses on finding a best-fit tropical polytope for some data in the tropical projective torus. As in the previous section, we focus on the space of equidistant trees on $[m]$ leaves. As shown in the previous section, an equidistant tree can be defined as an ultrametric. In the **TML** package, tropical principal component analysis focuses on the tree space defined as the space of ultrametrics on $[m]$ leaves. This method was first introduced in [26] and extended in [45].

In the examples below we instead use simulated data where each point resides in the tropical projective torus. The best-fit polytope, specifically a tropical triangle, is calculated through the use of the `tropical.PCA.Polytope()` function. This function takes an iterative approach to finding the vertices of the best-fit tropical triangle by incorporating vertex HAR with extrapolation which was shown in Section 3B. The primary purpose is to visualize the data along with the associated tropical triangle which is shown through the code that follows.

```
R> set.seed(1)

R> s <- 3 # number of vertices.
R> d <- 3 # dimension
R> N <- 100 # sample size

R> V <- rbind(
      c(100, 0, 0),
      c(0, 100, 0),
      c(0, 0, 100),
      c(-100, 0, 0),
      c(0, -100, 0),
      c(0, 0, -100)
    )

R> D <- cbind(
      rnorm(N, mean =  5, sd = 5),
      rnorm(N, mean = -5, sd = 5),
      rnorm(N, mean =  0, sd = 5)
    )

R> index <- sample(1:N, s)
R> S <- D[index, ]
R> DD <- pre.pplot.pro(S, D) |> apply(2, normaliz.vector)
R> res <- tropical.PCA.Polytope(S, D, V, I = 1000, 50)
R> DD <- pre.pplot.pro(res[[2]], res[[3]])
R> trop.tri.plot.w.pts(normaliz.ultrametrics(res[[2]]), DD)
```

The output of this code provides the tropical triangle shown in Figure 24. Of note, the best fit tropical polytope is an approximation and users should expect to get a different tropical triangle that is different than the one pictured in Figure 24.

4D. ***Application 4: Tropical kernel density estimation.*** A kernel density estimator (KDE) is a nonparametric density estimation method which uses kernel functions. This is a useful method in determining a number of data characteristics when the distribution of the data is unknown. This technique uses a kernel function, $\kappa(\cdot)$ which is simply a nonnegative, smooth function and conjunction with a bandwidth parameter [37; 40]. Like any density function however, there also must exist some normalizing constant, C such that $\frac{1}{C}\kappa(\cdot)$ integrates to one. A common kernel density estimator is the Gaussian kernel with

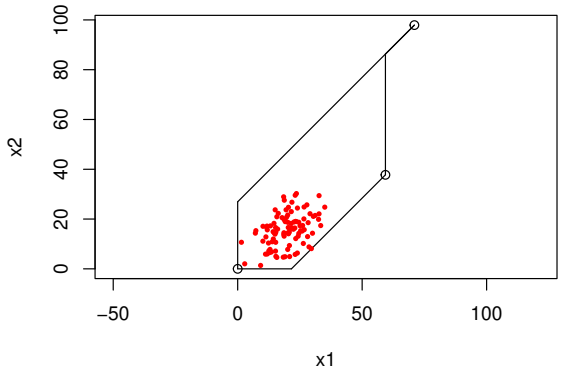

Figure 24. Best-fit tropical triangle found using `tropical.PCA.Polytope()`.

mean zero and standard deviation equal to one

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right). \tag{18}$$

In (18), the normalizing constant is the fraction prior to the exponential function. The bandwidth, or dispersion, parameter is represented by the standard deviation in the case of the Gaussian kernel. Importantly, a reasonable density estimate requires an appropriate choice for bandwidth.

Kernel density estimation over the tropical projective torus has previously been investigated by Weyenberg et al. in [41], which specifically focused on kernel density estimation over the space of phylogenetic trees using what is called the BHV metric. One of the more challenging aspects of their method was that the location of the center of the kernel function causes the value of the normalizing constant to vary. This requires a recalculation of the normalizing constant for each data point.

As an alternative to this method in [47], Yoshida et al. introduced the notion of kernel density estimation over the treespace represented as the space of ultrametrics on $[m]$ leaves using the tropical metric in conjunction with a Laplacian kernel function. Conducted experiments suggested that the normalizing constant remains constant regardless of the center of the function with $m \geq 5$. Bandwidth, which is based on the tropical metric, is chosen using a "nearest neighbor" approach as in [42] meaning that the bandwidth parameter is equal to the tropical distance to the closest other data point.

The **TML** package provides tropical kernel density estimation in the form of the `tropical.KDE()` function. The method leverages two functions from the **KDETrees** package called `pw.trop.dist()` and `bw.nn()` to first calculate the pairwise tropical distance between each data point and then find the bandwidth parameter for each data point [42]. In [42], Weyenberg et al. show how, using their BHV metric-based approach, to identify outliers in a set of gene trees. In this case, an outlier tree is a tree that falls in the tail of the distribution of trees.

Yoshida et al. conducted a similar experiment using the method developed in [47] on a the space of ultrametrics on $m = 10$ leaves to identify such outliers with fixed effective populations but differing species depths (SD). The species depth indicates the number of epochs between the common ancestor of all species (root node) and present day (leaf nodes). For this experiment, we consider an effective population of $N_e = 100000$ and varying species depths such that we obtain a sequence of ratios, R, of SD to N_e equal to 0.25, 0.5, 1, 2, 5, and 10.

The example below consists, again, of using the same two sets of 1000 simulated gene trees, $T1$ and $T2$, with ten leaves where $R = 5$ (represented as `Sim_Trees1` and `Sim_Trees2`). We then provide the cumulative results for each R in the plots of the receiver operator characteristic (ROC) curves for each R that follow.

In order to determine how well we can identify outliers using the `tropical.KDE()` function, we examine each tree in $T2$ as it is appended to the set of trees in $T1$. Using the pairwise tropical distance function, `pw.trop.dist()`, and `bw.nn()`, we find the bandwidth value for each tree in the set. Then, we calculate the density value using `tropical.KDE()` function. We determine how well the method identifies outliers by examining the receiver operator characteristic (ROC) curve. The larger the value of the area under the ROC curve, the better the method is at identifying outliers. In the code chunk below, we assume the density estimate on the final trial for all trees with original membership in $T1$ is representative of density estimates from previous trials. Therefore, when calculating values for the ROC curve in the code chunk below, we only use those density estimates. For demonstration, the following code chunk only shows the code necessary for $R = 5$ scenario.

```
R> set.seed(1)
R> I <- 1000 ## The number of trials
R> D1 <- Sim_Trees15; D2 <- Sim_Trees25; Q5 <- rep(0, I)
R> N1 <- nrow(D1)
R> for(i in 1:I){
       D <- rbind(D1, D2[i,])
       k <- dim(D)[1]
       P5 <- rep(0, k)
       X <- 1:k
       sigma <- D |> pw.trop.dist(D) |> bw.nn()
       P5 <- tropical.KDE(D, n, sigma, h = 2)
       Q5[i] <- P5[k]
   }
R> y <- c(rep(1, N1), rep(0, I))
R> predProbKDE5 <- c(P5[1:N1], Q5)
R> KDE5.ROC <- performance(prediction(predProbKDE5, y),
    measure="tpr", x.measure="fpr")
```

In general, we see that as SD, and therefore R, increases, so does the AUC value as shown below. When we reach the experiment representing $R = 10$, we see perfect classification, indicated by the AUC being equal to one. Figure 25 shows the associated ROC curves for each value of R. This provides us with a visual representation of the AUC values below.

```
R> (KDE025.AUC <- performance(prediction(predProbKDE025, y),
                          measure=‘‘auc’’)@y.values)

[[1]]
[1] 0.563876

R> (KDE05.AUC <- performance(prediction(predProbKDE05, y),
                          measure="auc")@y.values)

[[1]]
[1] 0.630703

R> (KDE1.AUC <- performance(prediction(predProbKDE1, y),
                          measure="auc")@y.values)

[[1]]
[1] 0.697034

R> (KDE2.AUC <- performance(prediction(predProbKDE2, y),
                          measure="auc")@y.values)

[[1]]
[1] 0.87902

R> (KDE5.AUC <- performance(prediction(predProbKDE5, y),
                          measure="auc")@y.values)

[[1]]
[1] 0.998542

R> (KDE10.AUC <- performance(prediction(predProbKDE10, y),
                          measure="auc")@y.values)

[[1]]
[1] 1
```

5. CONCLUSION. This paper provides a basic description of the tropical machine learning methods and functionality of the **TML** package in R. While we provide a thorough descriptions of most available

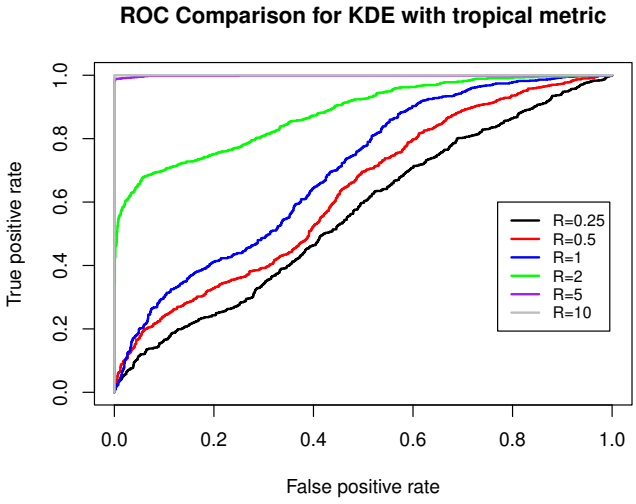

Figure 25. ROC curve for outlier detection when comparing two sets of gene trees with varying SD. Note that an increasing R indicates an increasing SD.

methods in the **TML** package we cannot cover everything. One important unsupervised method not covered are clustering methods over the tropical projective torus. We recommend to read [4] for a thorough treatment.

As shown in [25], all Euclidean statistical models can be described in terms of tropical algebra. With this in mind, we anticipate that **TML** package will continue to mature as new tropical data science methods are developed. We are already observing alternative methods of employing tropical support vector machines using HAR methods and neural networks in terms of tropical algebra. We encourage collaborators to provide input and recommendations via the **TML** GitHub page at https://github.com/barnhilldave/TML/issues.

ACKNOWLEDGMENTS. The authors would like to thank David Kahle for is input on the development of the **TML** package. RY and DB are partially supported from NSF DMS 1916037. GA is funded by EPSRC through the STOR-i Centre for Doctoral Training under grant EP/L015692/1. KM is partially supported by JSPS KAKENHI Grant Numbers JP22K19816, JP22H02364.

SUPPLEMENT. The online supplement contains version 3 of TML-SCRIPT-V3.rtf.

REFERENCES.

[1] M. Akian, S. Gaubert, Y. Qi, and O. Saadi, "Tropical linear regression and mean payoff games: or, how to measure the distance to equilibria", *SIAM J. Discrete Math.* **37**:2 (2023), 632–674. MR Zbl

[2] G. Aliatimis, R. Yoshida, B. Boyacı, and J. A. Grant, "Tropical logistic regression model on space of phylogenetic trees", *Bull. Math. Biol.* **86**:8 (2024), art. id. 99. MR

[3] D. Barnhill, *Markov chain Monte Carlo sampling of tropically convex sets*, Ph.D. thesis, Naval Postgraduate School, 2024. Zbl

[4] D. Barnhill and R. Yoshida, "Clustering methods over the tropical projective torus", *Mathematics* **11**:15 (2023), art. id. 3433. Zbl

[5] D. Barnhill, R. Yoshida, G. Aliatimis, and K. Miura, "TML: Tropical geometry tools for machine learning", 2023. R package version 1.1.0.

[6] D. Barnhill, R. Yoshida, and K. Miura, "Maximum inscribed and minimum enclosing tropical balls of tropical polytopes and applications to volume estimation and uniform sampling", preprint, 2023. Zbl arXiv 2303.02539

[7] P. Buneman, "A note on the metric properties of trees", *J. Combinatorial Theory Ser. B* **17** (1974), 48–50. MR Zbl

[8] F. Criado, M. Joswig, and F. Santos, "Tropical bisectors and Voronoi diagrams", *Found. Comput. Math.* **22**:6 (2022), 1923–1960. MR Zbl

[9] M. Develin and B. Sturmfels, "Tropical convexity", *Doc. Math.* **9** (2004), 1–27. MR Zbl

[10] B. Gärtner and M. Jaggi, "Tropical support vector machines", technical report ACS-TR-362502-01, ACS, 2006. Zbl

[11] S. Gaubert and M. MacCaig, "Approximating the volume of tropical polytopes is difficult", *Internat. J. Algebra Comput.* **29**:2 (2019), 357–389. MR Zbl

[12] E. Gawrilow and M. Joswig, "polymake: a framework for analyzing convex polytopes", pp. 43–73 in *Polytopes—combinatorics and computation* (Oberwolfach, 1997), edited by G. Kalai and G. M. Ziegler, DMV Sem. **29**, Birkhäuser, Basel, 2000. MR Zbl

[13] M. Joswig, *Essentials of tropical combinatorics*, Graduate Studies in Mathematics **219**, American Mathematical Society, Providence, RI, [2021] ©2021. MR Zbl

[14] M. Joswig and K. Kulas, "Tropical and ordinary convexity combined", *Adv. Geom.* **10**:2 (2010), 333–352. MR Zbl

[15] D. Kahle, "Tropical geometry in R", 2015. R package version 0.0.0.900.

[16] D. Kahle, L. Garcia-Puente, and R. Yoshida, "algstat: Algebraic statistics in R, 2017", 2017. R package version 0.1.1.

[17] J. F. C. Kingman, "The coalescent", *Stochastic Process. Appl.* **13**:3 (1982), 235–248. MR Zbl

[18] D. Liang, X. X. Shen, and P. Zhang, "One thousand two hundred ninety nuclear genes from a genome-Wide survey support lungfishes as the sister group of tetrapods", *Molecular Biology and Evolution* **30**:8 (2013), 1803–1807.

[19] B. Lin, B. Sturmfels, X. Tang, and R. Yoshida, "Convexity in tree spaces", *SIAM J. Discrete Math.* **31**:3 (2017), 2015–2038. MR Zbl

[20] G. Loho and M. Schymura, "Tropical Ehrhart theory and tropical volume", *Res. Math. Sci.* **7**:4 (2020), art. it. 30. MR Zbl

[21] D. Maclagan and B. Sturmfels, *Introduction to tropical geometry*, Graduate Studies in Mathematics **161**, American Mathematical Society, Providence, RI, 2015. MR Zbl

[22] W. P. Maddison and D. Maddison, "Mesquite: a modular system for evolutionary analysis", 2009, available at http://mesquiteproject.org. Version 2.72.

[23] D. Murdoch and D. Adler, "rgl: 3D Visualization Using OpenGL", 2023. Version 1.2.1.

[24] OSCAR, "Oscar — open source computer algebra research system", 2023. Version 0.12.2-dev.

[25] L. Pachter and B. Sturmfels, "Tropical geometry of statistical models", *Proc. Natl. Acad. Sci. USA* **101**:46 (2004), 16132–16137. MR Zbl

[26] R. Page, R. Yoshida, and L. Zhang, "Tropical principal component analysis on the space of phylogenetic trees", *Bioinformatics* **36**:17 (2020), 4590–4598. Zbl

[27] R Core Team, "R: a language and environment for statistical computing", 2014, available at http://www.R-project.org. R Foundation for Statistical Computing, Vienna.

[28] B. Rannala and Z. Yang, "Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci", *Genetics* **164**:4 (2023), 1645–1656. Zbl

[29] L. J. Revell, "phytools: an R package for phylogenetic comparative biology (and other things)", *Methods in Ecology and Evolution* **3**:2 (2012), 217–223. Zbl

[30] J. Richter-Gebert, B. Sturmfels, and T. Theobald, "First steps in tropical geometry", pp. 289–317 in *Idempotent mathematics and mathematical physics*, edited by G. L. Litvinov and V. P. Maslov, Contemp. Math. **377**, Amer. Math. Soc., Providence, RI, 2005. MR Zbl

[31] C. P. Robert and G. Casella, *Monte Carlo statistical methods*, 2nd ed., Springer, 2004. MR Zbl

[32] J. Sigwart, "Coalescent theory: An introduction", *Systematic Biology* **58**:1 (03 2009), 162–165. Zbl

[33] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "Singular 4-3-0—A computer algebra system for polynomial computations", software, available at http://www.singular.uni-kl.de. Zbl

[34] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions", *Oper. Res.* **32**:6 (1984), 1296–1308. MR Zbl

[35] D. Speyer and B. Sturmfels, "The tropical Grassmannian", *Adv. Geom.* **4**:3 (2004), 389–411. MR Zbl

[36] D. Speyer and B. Sturmfels, "Tropical mathematics", *Math. Mag.* **82**:3 (2009), 163–173. MR Zbl

[37] P. Sprent, *Applied nonparametric statistical methods*, Chapman & Hall, London, 1989. MR Zbl

[38] X. Tang, H. Wang, and R. Yoshida, "Tropical support vector machine and its applications to phylogenomics", *Algebr. Stat.* **14**:2 (2023), 133–165. MR Zbl

[39] H. Wang, K. Wang, G. Weyenberg, X. Tang, and R. Yoshida, "Rtropical: Data analysis tools over space of phylogenetic trees using tropical geometry", 2021. R package version 1.2.1.

[40] L. Wasserman, *All of nonparametric statistics*, Springer, 2006. MR Zbl

[41] G. Weyenberg, P. M. Huggins, C. L. Schardl, D. K. Howe, and R. Yoshida, "kdetrees: non-parametric estimation of phylogenetic tree distributions", *Bioinformatics* **30**:16 (04 2014), 2280–2287. Zbl

[42] G. Weyenberg, R. Yoshida, and D. Howe, "Normalizing kernels in the Billera–Holmes–Vogtmann treespace", *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **14**:6 (2017), 1359–1365. Zbl

[43] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL programming guide*: *the official guide to learning OpenGL*, *version* 1.2, Addison-Wesley, 1999. Zbl

[44] R. Yoshida, "Tropical data science", preprint, 2020. Zbl arXiv 2005.06586

[45] R. Yoshida, L. Zhang, and X. Zhang, "Tropical principal component analysis and its application to phylogenetics", *Bull. Math. Biol.* **81**:2 (2019), 568–597. MR Zbl

[46] R. Yoshida, M. Takamori, H. Matsumoto, and K. Miura, "Tropical support vector machines: Evaluations and extension to function spaces", preprint, 2021. Zbl arXiv 2101.11531

[47] R. Yoshida, D. Barnhill, K. Miura, and D. Howe, "Tropical density estimation of phylogenetic trees", preprint, 2023. Zbl arXiv 2206.04206

[48] R. Yoshida, K. Miura, and D. Barnhill, "Hit and run sampling from tropically convex sets", *Algebr. Stat.* **14**:1 (2023), 37–69. MR Zbl

RECEIVED: 24 Sep 2023 REVISED: 1 Jun 2024 ACCEPTED: 16 Jul 2024

DAVID BARNHILL:

david.barnhill@nps.edu
Mathematics Department, US Naval Academy, Annapolis, MD, United States

RURIKO YOSHIDA:

ryoshida@nps.edu
Department of Operations Research, Naval Postgraduate School, Monterey, CA, United States

GEORGIOS ALIATIMIS:

g.aliatimis@lancaster.ac.uk
STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, United Kingdom

KEIJI MIURA:

miura@kwansei.ac.jp
Kwansei Gakuin University, Sanda, Japan