2017

A Kernel Based Approach to Determine Atypicality

Austin O'Brien
South Dakota State University

A KERNEL BASED APPROACH TO DETERMINE ATYPICALITY

BY

AUSTIN O'BRIEN

A dissertation submitted in partial fulfillment of the requirements for the

Doctor of Philosophy

Major in Computational Science & Statistics

South Dakota State University

2017

A KERNEL BASED APPROACH TO DETERMINE ATYPICALITY

This dissertation is approved as a creditable and independent investigation by a candidate for the Doctor of Philosophy degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department.

Christopher Saunders, Ph.D. Date
Dissertation Advisor

Kurt Cogswell, Ph.D. Date
Head, Department of Mathematics and Statistics

Dean, Graduate School Date

To my mother, who always supported my pursuit of education. To my father, who encouraged me to choose my own interests. To my friends and siblings who remind me to enjoy all that life has to offer. To my wife, Rose, for keeping me level in these rough waters. And to my son Sean and yet-to-be-born twins, for making every late night worth it.

ACKNOWLEDGMENTS

I am incredibly grateful to Dr. Chris Saunders for the constant guidance, advice, and inspiration. Without his patience and open door, I never would have been able to finish this dissertation. I also need to thank my other committee members: Dr. Kurt Cogswell for his mentorship, feedback, and many hours discussing the material with me. And Dr. Cedric Neumann for taking the time out of his schedule to help me when needed.

Going back to my masters days, I need to thank Dr. Sung Shin, Dr. Ali Salehnia, and Dr. Manki Min from the Computer Science Department for helping me in my initial pursuits in graduate school. Dr. George Hamer, thank you for giving me the chance to teach, which opened the door and instilled my love of teaching, which is a primary factor in pursuing this degree. And finally, Mike O'Conner from the University of Montana for giving a freshman a second chance.

Finally, I would like to thank my family for always being supportive of me. My parents never wavered in their belief of my abilities. Thank you Rosie, my life's love, for consistently lifting me up when I felt my lowest. Without you, I never would have finished this work. And my son Sean; your unflappable happiness and joy would melt away any hardship I encountered in this process. I have no doubt that your unconditional love for me helped me face any difficulty head on with confidence.

TABLE OF CONTENTS

ABBREVIATIONS

SVM	Support Vector Machine
Bootstrap	Monte Carlo Bootstrap Simulation
MMH	Maximum Margin Hyperplane
LDA	Linear Discriminant Analysis
RBF	Radial Basis Function
SVDD	Support Vector Domain Description

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A KERNEL BASED APPROACH TO DETERMINE ATYPICALITY

AUSTIN O'BRIEN

2017

This dissertation outlines the development and use for a new probabilistic measure for categorization, referred to as atypicality. Given a set of known source objects, we can create a corresponding set of similarity scores between them. Assuming the set of scores has a normal distribution, we can estimate its parameters. Then, we can introduce new trace objects to the problem, and compute similarity scores for them. The main goal of the atypicality score is to determine if the new trace objects are similar to the source objects. To do this, we bootstrap many new scores using the estimated parameters (from the source scores), and compare the likelihood of these new scores to the scores belonging to the trace objects. We then make note of how often the trace objects have a higher likelihood value. The bootstrap result will be a number between zero and one, with smaller values indicating that the trace objects are not similar to the source objects. This is the atypicality value. We can use this as a p-value in a hypothesis test where the null hypothesis states that the trace objects are similar to the source objects. This can be used in a variety of applications, especially where we have multiple trace objects in multi-dimensional space. This dissertation will outline the development and use of the atypicality measure, show the results when the objects and scores are not normal, discuss the power of atypicality, and provide a comparison to support vector machines.

1. INTRODUCTION TO ATYPICALITY FOR CLASSIFICATION

In basic terms, atypicality can be thought of as a way to answer the question: "Do two independent simple random samples from a given known source?" The answer to this question can help people make decisions about the origin of objects from an unknown origin. This can be useful in many fields, particularly where classification is needed. An area of focus in this paper is the forensic sciences. An example application could be comparing shards of glass collected from clothing of a suspected thief. If we collected other glass shards from a broken window (a known source of origin), we would like to determine how atypical the glass shards are in reference to the glass shards from the window.

Other authors have developed their own definitions for atypicality, which we will review below.This paper proposes a new development and interpretation for atypicality, in the form of a hypothesis test. The hypothesis test is

H_0 : Two independent simple random samples are from a given known source.

H_1 : Not H_0.

In order to determine whether to reject or fail to reject the null hypothesis, a p-value must be calculated. This p-value is the result of the atypicality algorithm presented in this paper. It can be used as any other p-value, where most researchers would reject the null hypothesis if the p-value is smaller than some threshold, commonly 0.05.

Before proceeding with the literature review of atypicality, let us first walk through a toy example of atypicality in an informal matter. This will help us lock in a solid understanding of what atypicality is, how it is calculated, and how it is applied. The formal write-up of each step can be found within the pages that follow,

but let us follow each bulleted item as a critical step in the development of atypicality.

- First, we make the assumption that the samples are i.i.d. with an unknown distribution. We then state our hypothesis, that the samples are from a known given source. A significance criterion must then be set; 0.05 being an example. A We then collect samples from a known source. These objects will be used in a similar fashion as training data. We also collect the objects from the unknown source. This can be a single object, or several objects. Feature data is then measured from these samples.

- Next, we compute pairwise similarity scores between the objects using some method that is appropriate for the type of data we have. For simple toy examples, we can use the euclidean distance between two objects. For other types of data, there may be a well developed method of computing a similarity score between the objects. By using the scores, any dimensionality in the original data space is reduced to a vector where the length is directly related to the number of objects we have, including both the known objects and the objects of unknown origin. There is one important caveat about the scores: we must assume that this score vector is a multivariate normal random variable. The reason we must make this assumption is because the process of computing the p-value for the hypothesis test to come involves estimating the parameters for this multivariate normal vector of scores. If the vector of scores is not a multivariate normal random vector, steps may be taken to transform the data to become one.

- As mentioned before, the parameters for the multivariate normal score vector are estimated. These parameters include the covariance matrix and the mean vector. Only the scores that relate to the pairwise comparison between the

known origin objects are used. This is similar as using the known origin objects as training data. The trick is, how do we estimate these parameters with a single multivariate score observation. Well, the mean vector is estimated by simply calculating the mean of the scores that make up the score vector. This value is repeated in a vector that is the same dimension as the score vector. The covariance matrix is developed in a rigorous process that can be found starting at lemma 1 on page 21, that continues on until page 33.

- Developing the covariance matrix is easily one of the most difficult aspects of this dissertation. Reducing the process into digestible portions is no small feat, but here is a best attempt. We first define a few different covariance and variance terms for the different types of scores. If a_i and a_j are objects from the original dimensional space, then s_{ij} is the score between them. The variance term for any s_{ij} is defined as simply σ^2. To find the variance of a score, we take the variance of the model on page 20. This yields a variance $\sigma^2 = 2\sigma_a^2 + \sigma_e^2$, where σ_e^2 is the variance for the error term. The covariance term between two scores $Cov\left(s_{ij}, s_{ij'}\right) = \sigma_a^2$, where $j' \neq i, j$. We then construct a projection matrix where the scores are represented down the rows and across columns as demonstrated on page 24. The process of then determining estimates for σ_a^2 and σ_e^2 involves a lengthly eigen decomposition that allows us to solve for a series of sums of squares. This process spans pages 26 through 33. These sums of squares can be used as estimates for the Mean Square values, which in turn, can be used formulas to estimate the variance and covariance terms mentioned earlier; as shown on page 33.

- The dimensionality of these parameters is important to note. Both the estimated mean vector and the covariance matrix are conditional, as proposed by Izenman [29] (as seen on page 38). Given the vector of scores that relate to

the relationships between the test observations, the dimension of the conditional mean vector will be the same length. The conditional covariance matrix will be a square matrix with matching column and row counts.

- Once we have the estimated parameters, applying them with the multivariate normal equation gives us a probability density function. Essentially, we are going to apply these estimated parameters to calculate the probability assigned for the objects given these parameter values. We don't, however, simply compare these assumed probabilities for the objects of known and unknown origin and compare them, but rather we perform a Monte Carlo bootstrap simulation.

- A bootstrap simulation is a special case of Monte Carlo simulations. With repeated sampling, we sample a random multivariate vector using the parameters estimated in the step before. These conditional parameters used to sample this new observation will remain fixed throughout the bootstrap process; no new parameters are estimated. We then compare the assigned probability of this sample vector to that of the objects of unknown origin. Because this is a bootstrap simulation, we do this many times. If $f(\cdot, \theta)$ is the pdf with estimated parameters θ, and $\vec{s}$ is the complete vector of scores; then the comparison of the probability of the sampled score vector $\vec{S^*}$ and that of the unknown source score vector $\vec{s_0}$ would be shown as $f\left(\vec{S^*}|\vec{s}, \vec{\theta}\right) \leq f\left(\vec{s_0}|\vec{s}, \vec{\theta}\right)$. Every occurrence of this condition being true is summed to a number. This number is divided by the total number of bootstraps that occurred. This will always give a number between 0 and 1. This number is then used as the p-value in the hypothesis test.

- A researcher can set any threshold for the p-value that would lead them to reject the null hypothesis that they deem appropriate for their experiment. As

mentioned before, a common number is 0.05. Looking back at the comparison of the likelihoods between the sampled score vector and the score vector relating to the unknown source objects, we can see that if the objects from the unknown source are atypical to the objects from the known source, then the p-value will be small. These small values would lead a researcher to reject the null hypothesis that states that they are typical.

- This last bullet isn't so much a step as it is a explanation of the reasoning for atypicality. We were able to avoid the curse of dimensionality by using pairwise comparison between the objects. Also, we have a probability based metric that can be used to make decisions on the atypicality of objects of unknown origins, when compared to those of a known source. It's also important to note that using conditional parameters, we can consider several unknown source objects at once instead of having to consider them individually. This differs from machine learning methods that simply yield a classification for a single object without considering any other unknown source objects collected at the same time; nor is there much room for interpretation of the model itself. This makes the atypicality classification method described above a robust option for any researcher.

To help summarize this, below is a step-by-step pseudocode version of the process 1:

1.1 Related metrics. Atypicality is an important statistical concept that could potentially be applied in many fields. A particular field of interest is forensic science. In its basic terms, atypicality is an index to determine how likely an object came from a population of other objects. This addresses many questions that pertain to forensic science; such as, did a sample with a given characteristic belong to a population believed to be the source of the sample? Before we describe atypicality more directly, we will review the previous research and importance of

```
Data: scores
initialize scores;
scores.known ← partition.known(scores);
scores.unknown ← partition.unknown(scores);
params.scores.known ← param.estimates(scores.known);
CondSig ← construct.conditional.covmat(params.scores.known);
CondMu ← construct.conditional.mean(params.scores.known);
sum ← 0;
for j ← 1 to NumSamples do
    Samples ← get.scores(CondSig, CondMu);
    if f(Samples, CondSig, CondMu) ≤
     f(scores.unknown, CondSig, CondMu) then
        sum ← sum + 1;
    end
end
atypicality ← sum/NumSamples;
```

Algorithm 1: Atypicality Algorithm Pseudocode

matching samples to a given population in forensic science. Before we move forward, it is important to define our usage of the term "match". Here, when we say that a sample is a match to a population, we mean that the probability that the sample belongs to a population is within some threshold.

Trying to match found samples to a given population has been the subject of extensive study in the forensic sciences. Theoretical work can be found in papers such as [51],[52], [35] and [66]. Examples involving characteristics more attuned to real world situations include experiments involving: chips of paint [73] and [43], matching human hair from a sample to an individual [19], matching shoe prints [22], and even dental impressions [1]. More recently, a paper comparing the depth contours of tool mark surfaces has developed an algorithmic matching procedure [41].

This paper describes the development of an atypicality index that uses similarity scores in its construction. Similarity scores have been used in several areas of the forensic sciences and biometrics, including facial recognition [67], [75], [54]

and [61].[1] The higher the similarity score between two objects, the more likely they have come from the a common source. More examples of similarity scores used in forensic applications include: speaker/audio recognition [45], [15], [20], [3], [21]; text/writing recognition [47], [14]; shoe prints [13]; fingerprints [31]; walking patterns [7]; and statistical theory [12], [6].

This paper will now formally describe atypicality theory. Beyond that, we will show how to derive an atypicality statistic from score based pairwise comparison data. A simulation of the power of the statistic is provided, as well as several other examples and simulations. This paper focuses on the conditional atypicality statistic. Future work will likely include work for the unconditional statistic.

1.2 Previous Definitions of Atypicality. Below is a description of a previously defined atypicality metric. This is not the same as the one being presented, but is shown as a review of similar work done. In particular, we present the work of [44], and how it is applied to multidimensional data. Given a set of training data, our goal is to assess how likely it is that the observation x arose from a population in question. This has a similar goal as our atypicality, but is calculated differently.

First, we will start with a summary of the concept of atypicality among objects, as described by [44]. Formally, we have n i.i.d. objects X_i where $i = 1, \ldots, n$. This is shown as

$$X_i \sim MVN(\mu, \mathbf{\Sigma}); \tag{1}$$

where $MVN(\mu, \mathbf{\Sigma})$ denotes a multivariate normal distribution with mean vector μ and covariance matrix Σ. In R^p space, these objects have a multivariate normal distribution, with realized values of x_{ij}, where $i = 1, \ldots, n; j = 1, \ldots, p$. Using an

[1] We can easily convert a dissimilarity score to a similarity score.

atypicality index, we can assign a value from the unit interval $[0, 1]$ to each object that indicates the probability of incorrectly stating that the objects came from different populations, when they actually came from the same one. Values closer to one indicate that the object in question is more typical, and values closer to zero indicate that the object is atypical and may have come from a different population.

From the assumption of multivariate normality, for each object vector $\mathbf{x}_i$, we have the associated tail area a_i under the density curve. This tail area is computed to the right of the normalized squared Mahalanobis distance. The Mahalanobis distance is described at great length in [42]. It is essentially a measure of the distance between a point P and a distribution D. In multi-dimensional space, it is a way of measuring how many standard deviations away P is away from the mean of D along each principal component. It also accounts for other aspects that the Euclidean distance does not. These include the covariance between variables, and the fact that variances in different directions (for each variable) may be different. When the respective axes are scaled to have the unit variance and the variables are uncorrelated, then the Mahalanobis distance is equivalent to the Euclidean distance.

In forming the distance that corresponds to each object, $\mathbf{x}_i$ is first deleted from the sample to prevent it from contaminating the estimates of the mean and covariance matrix. Let $\bar{\mathbf{x}}$ denote the mean vector of all the objects, and $\widehat{\mathbf{\Sigma}}$ be the unbiased estimate of the covariance matrix Σ. Then define $\bar{x}_{(i)}$ and $\widehat{\mathbf{\Sigma}}_{(i)}$ as the resulting values of $\bar{s}$ and $\widehat{\mathbf{\Sigma}}$ after the deletion of $\mathbf{x}_i$ from the data. The squared Mahalanobis distance is then

$$\delta(\mathbf{x}_i|\bar{x}_{(i)}, \widehat{\mathbf{\Sigma}}_{(i)}) = (\mathbf{x}_i - \bar{x}_{(i)})^T \widehat{\mathbf{\Sigma}}_{(i)}^{-1} (\mathbf{x}_i - \bar{x}_{(i)}).$$

Now, to get the tail area a_i to the right of $\mathbf{x}_i$, we multiply the Mahalanobis

distance by the factor

$$c(n,\nu) = \frac{(n-1)\nu}{(np)(\nu+p-1)},$$

where p and $\nu = n - p - 1$ are degrees of freedom. In our case, n is defined as the number of objects that we have, and p is the number of dimensions in each vector.

By multiplying the Mahalanobis distance by this factor, then we have

$$c(n,\nu)\delta(\mathbf{x}_i|\bar{x}_{(i)}, \widehat{\mathbf{\Sigma}}_{(i)}). \tag{2}$$

The above statistic is closely related to Hotelling's T^2 statistic. By Anderson[5], it is known that the squared Mahalanobis distance multiplied by the number of observations is equal to T^2. Anderson continues to show that $T^2/(n-1)$ can be written as the ratio of a noncentral χ^2_ν and another independent χ^2_p. He then goes on to show that $(T^2/(n-1))\left[(n-p+1)/p\right] \sim F_{p,n-p+1}$. To summarize, if $X \sim T_{p,n}$, then $\left[(n-p+1)/pn\right] X \sim F_{p,n-p+1}$. This means that the equation 2 is similar to a T^2 statistic in that it has a similar $F_{p,n-p+1}$ distribution.

Now let us write equation 2 in shorthand as a_i:

$$a_i = c(n,v)\delta(\mathbf{x}_i|\bar{x}_{(i)}, \widehat{\mathbf{\Sigma}}_{(i)}). \tag{3}$$

According to [44], the a_i values will have a uniform distribution on the unit interval $[0,1]$.

To illustrate what we have stated above, here is a simple example. We will randomly sample n objects with p-dimensions and show their atypicality index value to a multivariate normal distribution with known parameters. Specifically, will set

$n = 10$ and $p = 3$. The mean vector will be set as

$$\mu = \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix}.$$

To help make this information more relatable, we could say that these values represent 2"x4"x10' pieces of wood. To get an appropriate positive definite covariance matrix, we can specify the principle components and construct the covariance matrix. We do this in R [57] by modifying the "`genPositiveDefMat`" function in the "`clusterGeneration`" package [56]. The modifications we make remove the random components of the covariance matrix generation. In the R program, the eigenvalues were set as 1 through n. For this example, the known covariance matrix is

$$\Sigma = \begin{bmatrix} 2.0000 & 0.2113 & 0.7887 \\ 0.2113 & 2.2887 & 0.5000 \\ 0.7887 & 0.5000 & 1.7113 \end{bmatrix},$$

with eigenvalues 3, 2, 1. Using R, we obtain the samples found on Table 1 rounded to two decimal places and their corresponding area values a. We remind the reader that when computing the estimates for the mean vector and covariance matrix, we remove the object in question first to prevent contamination of the estimated parameters by that object. That being said, below is the mean vector and covariance matrix for all the samples found in Table 1.

$$\hat{\mu} = \begin{pmatrix} 2.1692 \\ 4.1405 \\ 9.4090 \end{pmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} 3.9037 & 0.0764 & 1.8404 \\ 0.0764 & 1.9876 & 0.4604 \\ 1.8404 & 0.4604 & 2.0957 \end{bmatrix},$$

with eigenvalues 5.0812, 2.0657, 0.8401.

Looking at Table 1, we notice that the object with the smallest atypicality score is object x_5. This may be intuitive by looking at the objects first dimension, which is negative and quite far from the mean. The object with the largest atypicality index is object x_9 with whose values are close to the mean vector values.

We now observe the results when we increase the size of the dimensions, approaching the total number of objects that we have. We are constrained to a minimum of two and a maximum of $n-2$ dimensions. This is due to the limits of computing an atypicality score using the F distribution. We need the degrees of freedom to be larger than zero. The degrees of freedom are computed as $\nu = n - p - 1$, where n is the total sample size and p is the number of dimensions we have. As the number of objects gets larger, we will expect the density of the resulting $\mathbf{a}$ vector to appear uniform on the unit interval. Figure 1 shows the resulting density when we have n equal to one thousand, and the dimensions p increasing in size. We can see that as the number of dimensions increase, the empirical density is further removed from a uniform density and approaches a bell shaped curve; meaning that this can't be used as a p-value.

This fact is extremely important to this paper. We will see further in the paper that we use similarity scores between objects to reduce the dimensionality. This helps to maintain the uniform distribution property that we desire. However, applying McLachlan's method directly with a vector of scores will still have the problem mentioned in the previous paragraph.

From [44], we can see how to use the atypicality index value a_i as a p-value for a test of the compatibility of an object (or a set of objects) with a given population. We will prove this further along in this paper. A feature vector can be assessed as being atypical of a given population if

$$a_i \leq \alpha,$$

Sample	p_1	p_2	p_3	a
1	2.68	6.74	9.84	0.1921
2	3.57	2.75	11.20	0.3200
3	3.05	5.34	10.86	0.3579
4	4.95	4.15	10.52	0.2053
5	-2.15	5.40	8.19	0.0190
6	1.09	2.41	9.64	0.4942
7	0.45	2.51	6.30	0.0196
8	3.40	3.88	8.69	0.9491
9	2.38	3.63	8.97	0.9676
10	2.26	4.59	9.87	0.9490

Table 1: Randomly drawn samples and their corresponding atypicality values.

where α is some predetermined threshold. As stated by McLachlan, although we assume that the population is normal, the measure of atypicality can still be considered useful if the conditional density is at least elliptically symmetric.

We now a different description of atypicality, as described by Aitchison[2]. Working with a vector of similarity scores, if we attempted to implement McLachlan's method; we will have the problem where our atypicality values are less uniform and occur more often on the tails. This is why we must find a better method of determining atypicality indexes. Such a method is to use the index of [2] described below.

Given a multivariate normal density f with observations ω, given parameters θ, the index of atypicality for an object vector $\mathbf{x}$ is defined by

$$a(\mathbf{x}) = 1 - \int_{H(\mathbf{x})} f(\omega; \theta) d\omega, \tag{4}$$

where

$$H(\mathbf{x}) = \{\omega : f(\omega, \theta) > f(\mathbf{x}, \theta)\} \tag{5}$$

is the set of all observations more typical of the group than $\mathbf{x}$. To provide an assessment of $a(\mathbf{x})$, we replace the conditional density $f(\mathbf{x}, \theta)$ with $\hat{f}(\mathbf{x}, \theta)$ in

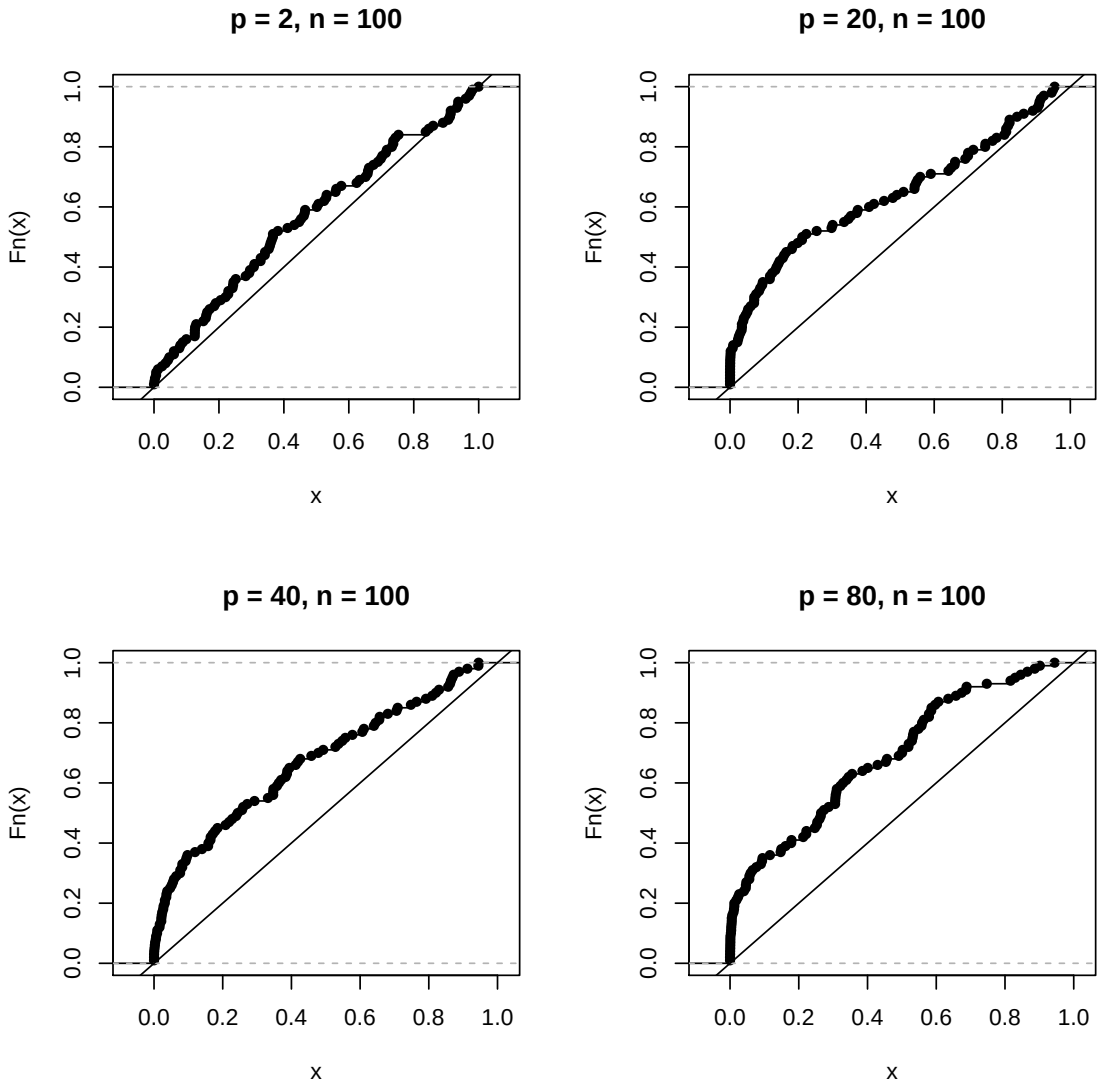


Figure 1: The density of the **a** vector with objects under increasing dimensions.

Equation (4) on page 12 and Equation (5) on page 12. If $\hat{a}(\mathbf{x})$ represents the estimate of $a(\mathbf{x})$, then it can be shown under the normal model that

$$\hat{a}(\mathbf{x}) = a(\mathbf{x}). \tag{6}$$

That is to say $\hat{a}(\mathbf{x})$ can be interpreted, in a frequentist sense, as the level of significance associated with the test of the compatibility of $\mathbf{x}$ with the population of objects.

More specific to our situation, we substitute the integral in Equation (4) on page 12 with

$$a(\mathbf{x}_n) = 1 - \int_{H(\mathbf{x}_n)} f(\omega, \theta | \mathbf{X}_n = \mathbf{x}_n) d\omega$$

where

$$H(\mathbf{x}_n) = \{\omega : f(\omega, \theta | \mathbf{X}_n = \mathbf{x}_n) > f(\mathbf{x}_n, \theta | \mathbf{X}_n = \mathbf{x}_n)\}.$$

In order to use these atypicality values as a p-value, it is important to show that they will have a uniform distribution when the null hypothesis is true. We will show that this is mathematically true for our version of atypicality in a future section.

1.3 Typicality as a P-value. For the purposes of this research, we will apply the atypicality index as a p-value. This p-value can be used to test the the null hypothesis that an object (or set of objects) introduced to a population has a differing distribution than that population, and can therefore be considered atypical. The alternative hypothesis would then be that the population and the newly introduced object do have similar distributions, and the object(s) may be considered typical. This is shown as

H_0 : Two independent simple random samples are from a given known source.

H_1 : Not H_0.

If we estimate the parameters from the known source population of scores, we can then use the resulting multivariate normal likelihood function to compare the density value of the scores pertaining to the unknown object with the scores sampled from the estimated distribution of the known source population. Now, let $f\left(\vec{s_0}|\vec{s}, \vec{\theta}\right)$ be the multivariate normal pdf such that larger values give evidence that

H_0 is true. We will derive the log-likelihood function in a future section of this paper. It will be shown to be

$$-2\log(\ell(\mu_s, \hat{\sigma_a^2}, \hat{\sigma_e^2}|s_i)) = N\ln(2\pi) + \ln(\widetilde{\lambda_1}) + (n-1)\ln(\widetilde{\lambda_2}) + (N-n)\ln(\widetilde{\lambda_3}) + \frac{SS_{\mu_s}}{\widetilde{\lambda_1}} + \frac{SS_a}{\widetilde{\lambda_2}} + \frac{SS_e}{\widetilde{\lambda_3}}. \tag{7}$$

where $\bar{s}, \hat{\sigma_a^2},$ and $\hat{\sigma_e^2}$ are restricted maximum likelihood estimates of the parameters. Let us simply denote the set of parameters to be

$$\theta = \left\{\mu_s, \sigma_a^2, \sigma_e^2\right\}.$$

Now, we define the entire set of scores between objects as $mathbfs$. Here, we will show that when H_0 is true,

Theorem 1. $p(\mathbf{s}) = Pr\{f(\mathbf{S}, \theta) \leq f(\mathbf{s}, \theta)\}$.

Then $p(\mathbf{s})$ can be thought of as a valid p-value for our hypotheses. To show that $p(\mathbf{s})$ is a valid, unbiased p-value, consider the following proof.

Proof. *First, let $F(\omega)$ denote the continuous cumulative distribution function (cdf) of $f(\mathbf{S}, \theta)$. Then we can define*

$$p(\mathbf{s}) = Pr\{f(\mathbf{S}, \theta) \leq f(\mathbf{s}, \theta)\} = F\left(f(\mathbf{s}, \theta)\right).$$

So, we will show that by the probability integral transformation, that $f(\mathbf{S}, \theta)$ has a continuous cdf F and $p(S) = F(f(\mathbf{S}, \theta))$. By definition, F is a non-decreasing function. When F is increasing,

$$F^{-1}\left(p(\mathbf{s})\right) = f(\mathbf{s}, \theta) \Leftrightarrow F\left(f(\mathbf{s}, \theta)\right) = p(\mathbf{s}).$$

Also let

$$F^{-1}(1) = \infty \text{ if } F\left(f(\mathbf{s},\theta)\right) < 1 \text{ for every } f(\mathbf{s},\theta), \text{ and}$$
$$F^{-1}(0) = -\infty \text{ for any } F.$$

Then for $p(\mathbf{S}) = F\left(f(\mathbf{S},\theta)\right)$ *we have for* $0 < p(\mathbf{s}) < 1$,

$$\begin{aligned} Pr\{p(\mathbf{S}) \leq p(\mathbf{s})\} &= Pr\{F\left(f(\mathbf{S},\theta)\right) \leq p(\mathbf{s})\} \\ &= Pr\{F^{-1}\left[F\left(f(\mathbf{S},\theta)\right)\right] \leq F^{-1}\left(p(\mathbf{s})\right)\} \end{aligned}$$

because F^{-1} *is increasing.*

Since F *is strictly increasing, then* $F^{-1}\left[F\left(f(\mathbf{s},\theta)\right)\right] = f(\mathbf{s},\theta)$. *Therefore, we have*

$$\begin{aligned} Pr\{F^{-1}\left[F\left(f(\mathbf{S},\theta)\right)\right] \leq F^{-1}\left(p(\mathbf{s})\right)\} &= Pr\{f(\mathbf{S},\theta) \leq F^{-1}\left(p(\mathbf{s})\right\} \\ &= F\left[F^{-1}\left(p(\mathbf{s})\right)\right] && (\textit{by the definition of } F) \\ &= p(\mathbf{s}) && (\textit{by the continuity of } F). \end{aligned}$$

At the end points, we have

$$Pr\{p(\mathbf{S}) \leq p(\mathbf{s})\} = 1 \text{ for } p(\mathbf{s}) \geq 1 \text{ and}$$
$$Pr\{p(\mathbf{S}) \leq p(\mathbf{s})\} = 0 \text{ for } p(\mathbf{s}) = 0.$$

This implies that $p(\mathbf{S})$ *has a uniform distribution on the unit interval,*

$$p(\mathbf{S}) \sim \textit{Unif}(0,1).$$

This in turn implies that,

$$Pr\{p(\mathbf{S}) \leq p(\mathbf{s})\} = p(\mathbf{s}) \text{ for } 0 < p(\mathbf{s}) < 1.$$

So for every $0 \leq \alpha \leq 1$

$$Pr\{p(\mathbf{S}) \leq \alpha\} \leq \alpha.$$

Since $Pr\{p(\mathbf{s}) \leq \alpha\} \leq Pr\{p(\mathbf{S}) \leq \alpha\} \leq \alpha$ *for every* $0 \leq \alpha \leq 1$*, then we can say that* $p(\mathbf{S})$ *is a valid p-value.*

It is important to note the direction of the inequality sign in the probability statement $Pr\{f(\mathbf{S}, \theta) \leq f(\mathbf{s}, \theta)\}$. This is the reverse of what one might normally see in hypothesis testing, but one can see that by the definition of our hypotheses, this is the correct format. This probability can be stated formally as the probability that the likelihood of the estimated parameters from a known population given a set of scores from a similar distribution is less than or equal to the likelihood of the estimated parameters given the object-in-question's scores. So if we have object(s) in question with an associated set of scores that give a low likelihood value for the estimated parameters compared to the known population scores, then the probability statement decreases, making it more likely that we reject the null hypothesis.

Typically, the integrals to calculate $p(s)$ do not exist in closed form and it will be necessary to use resampling techniques to calculate its value.

In order to show that the distribution of atypicality is uniform when H_0 is true, we can run a simulation where we calculate the p-value many times for a random population of objects with a random new object being tested for atypicality. The algorithm with known parameters is given in Algorithm 2.

In this instance, we use known parameters for the likelihood function. Figure 2

```
Data: scores
Initialize atypicality.vector;
for i ← 1 to NumSims do
    initialize scores;
    scores.known ← partition.known(scores);
    scores.unknown ← partition.unknown(scores);
    params.scores.known ← param.estimates(scores.known);
    CondSig ← construct.conditional.covmat(params.scores.known);
    CondMu ← construct.conditional.mean(params.scores.known);
    sum ← 0;
    for j ← 1 to NumSamples do
        Samples ← get.scores(CondSig, CondMu);
        if f(Samples, CondSig, CondMu) ≤
         f(scores.unknown, CondSig, CondMu) then
            sum ← sum + 1;
        end
    end
    atypicality.vector[i] ← sum/NumSamples;
end
plot(cdf(typicality.vector));
```

Algorithm 2: Uniform P-Value Simulation Pseudocode

is a graph of the cumulative distribution function from one such simulation. We would expect a straight diagonal line from $F(p(\mathbf{S})) = 0$ to $F(p(\mathbf{S})) = 1$ as the p-value also goes from 0 to 1 along the interval. This can be seen in figure 2. The pseudo code for this is found in Algorithm 2. The hardware and software specifications can be found in the appendix.

We will see, that when the parameters are not known and we must use our REML estimates, we will see a distribution that is not exactly uniform. However, as n approaches infinity, this distribution will converge weakly to a uniform distribution. With our estimates denoted with hats, then we can write this as

$$Pr\{f(\mathbf{S}, \hat{\theta}) \leq f(\mathbf{s}, \hat{\theta})\} \rightsquigarrow \text{Unif}(0, 1), n \to \infty.$$

1.4 Parameter Estimation for the Multivariate Normal Likelihood Function for Pairwise Scores. As we have seen in the previous chapter, the atypicality score is

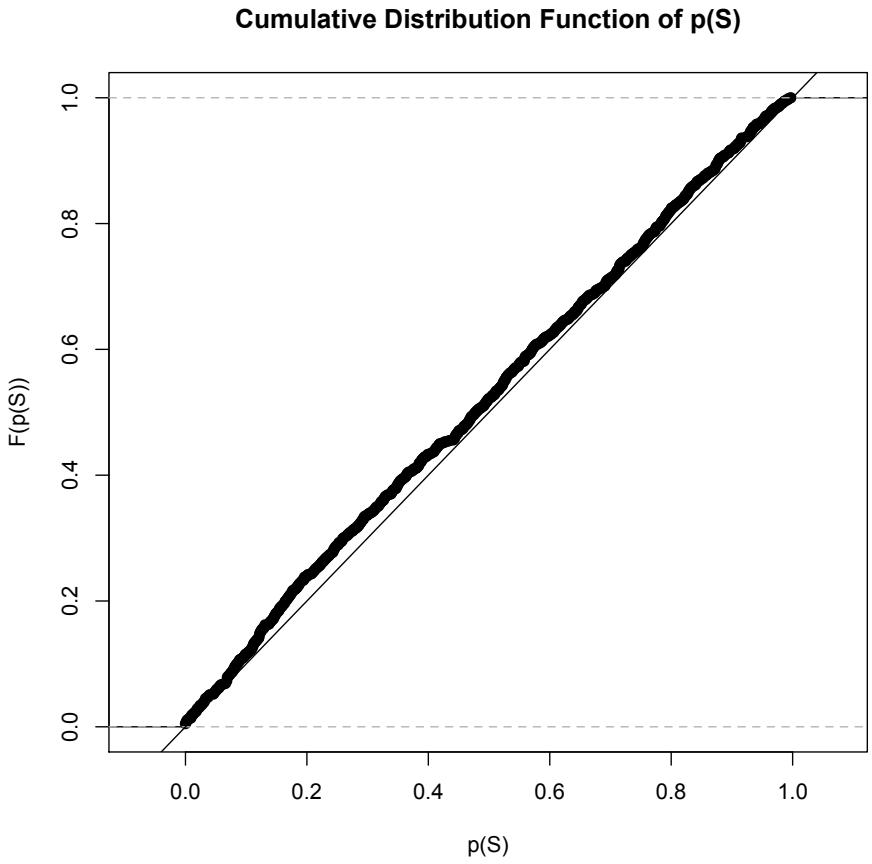


Figure 2: The Empirical Cumulative Distribution Function of Atypicality Measures.

given as the probability that the likelihood of the parameters given the newly bootstrapped scores will be less than or equal to the likelihood of the parameters given our observed scores in question. Since we assume that the scores have a multivariate normal distribution, then these likelihoods are multivariate normal likelihood functions.

If we take the log of the density and multiply by negative two, we have

$$-2\log(l) = N\log(2\pi) + \log(|\mathbf{\Sigma}|) + (\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1}).$$

This chapter deals with finding appropriate values for $|\mathbf{\Sigma}|$ and the derivation of $(\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1})$. Much of the work for finding these necessary parameters

was conducted by [48].

We will start with the following assumptions. We start with a population of n i.i.d. objects $\mathbf{X} = \{X_1, X_2, ..., X_n\}^T$, that have a similar, but arbitrary distribution, shown as:

$$\mathbf{x} \sim F$$

.

The product of the set of objects with itself can be mapped to the real space, which we call a score:

$$s : X \times X \mapsto \Re.$$

Our next assumption is that the variance for each score is finite,

$$\mathrm{Var}\left(s\left(x_i, x_j\right)\right) < \infty.$$

Now consider the parametric model for the scores;

$$s_{ij} = \theta + a_i + a_j + \epsilon_{ij}\ ,\ 1 \leq i < j \leq n, \tag{8}$$

where the a_i's are i.i.d. random variables with $\mathrm{E}a_i = 0$ and $\mathrm{Var}(a_i) = \sigma_a^2$; the ϵ_{ij}'s are also i.i.d. random variables with $\mathrm{E}\epsilon_{ij} = 0$ and $\mathrm{Var}(\epsilon_{ij}) = \sigma_e^2$. We will also assume that these random variables are normally distributed,

$$a_i \sim \mathcal{N}(0, \sigma_a^2),$$
$$\epsilon_{ij} \sim \mathcal{N}(0, \sigma_e^2).$$

From here, we have the necessary information to begin our study of $\mathbf{\Sigma}$.

If we have a number of n X_i objects, then we will have $N = \binom{n}{2}$ total scores in the vector $\mathbf{s}$. Under this model, we would then have the vector of a_i and a_j objects

and the vector of error terms ϵ_{ij} given as $\mathbf{a}$ and $\boldsymbol{\epsilon}$ respectively. This allows to give a vector/matrix representation of the model:

$$\mathbf{s} = \mathbf{P}\mathbf{a} + \boldsymbol{\epsilon} + \theta\mathbf{1},$$
$$\mathrm{E}[\mathbf{s}] = \theta\mathbf{1}.$$

where $\mathbf{P}$ is a design matrix that has the form of:

$$\mathbf{P} = \begin{pmatrix} P_{12} \\ P_{13} \\ \vdots \\ P_{(n-1)n} \end{pmatrix}.$$

where each P_{kl} row is a vector length n that has 1's in the k and l indices of that row and zeros elsewhere. For example,

$$P_{24} = (0, 1, 0, 1, 0 \cdots 0).$$

(We will soon see more on $\mathbf{P}$.)

It can be shown that the covariance for $\mathbf{s}$ takes on the following structure:

Lemma 1. *Cov*$(\mathbf{s}) = \mathbf{P}\mathbf{P}^T\sigma_a^2 + \sigma_e^2\mathbf{I}$

Proof.

$$\begin{aligned} \mathrm{Cov}(\mathbf{s}) &= \mathrm{Cov}(\mathbf{P}\mathbf{a}) + \mathrm{Cov}(\epsilon) \\ &= \mathbf{P}\mathrm{Cov}(\mathbf{a})\mathbf{P}^T + \sigma_e^2\mathbf{I} \\ &= \mathbf{P}\sigma_a^2\mathbf{P}^T + \sigma_e^2\mathbf{I} \\ &= \mathbf{P}\mathbf{P}^T\sigma_a^2 + \sigma_e^2\mathbf{I}. \end{aligned}$$

□

If n is the number of known objects, and m is the number of unknown objects, we use the following covariance matrix for our likelihood functions, developed by [48], where $\text{Cov}(\mathbf{s}) = \mathbf{\Sigma}$ and is shown as

$$\mathbf{\Sigma} = \sigma_e^2 \mathbf{I}_{NM} + \sigma_a^2 \mathbf{P}\mathbf{P}^T.$$

where NM has the value of $\binom{n+m}{2}$. The derivation of $\mathbf{\Sigma}$ follows.

First we look at the covariance between two scores s_{ij} and $s_{ij'}$ where $i = i$, but $j \neq j'$. By conditioning on the common term x_i, we have

Lemma 2. *Cov*$(s_{ij}, s_{ij'}) = \sigma_a^2$.

Proof.

$$\begin{aligned}
\text{Cov}(s_{ij}, s_{ij'}) &= \text{E}\left[\left(s\left(x_i, x_j\right) - \theta\right)\left(s\left(x_i, x_{j'}\right) - \theta\right)\right] \\
&= \text{E}\text{E}_{|X_i = x}\left[\left(s\left(x_i, x_j\right) - \theta\right)\left(s\left(x_i, x_{j'}\right) - \theta\right)\right] \\
&= \text{E}\left[(s_i - \theta)(s_i - \theta)\right] \\
&= \text{Var}(s_i) = \sigma_a^2.
\end{aligned}$$

which is the variance of the scores that pertain to the i^{th} object. □

Now we will define the variance for a given score between objects i and j as $\sigma^2 = \text{Var}(s_{ij}) = 2\sigma_a^2 + \sigma_e^2$. With our assumption that $\sigma_e^2 > 0$, then we also have $\sigma^2 \geq 2\sigma_a^2$.

Next, we will look at the correlation between two scores,

$$\text{Corr}\left(s_{ij}, s_{ij'}\right) = \frac{\text{Cov}(s_{ij}, s_{i'j'})}{\left(\text{E}\left(s_{ij} - \theta\right)^2\right)^{1/2}\left(\text{E}\left(s_{i'j'} - \theta\right)^2\right)^{1/2}}$$

where we will have three cases:

$$\begin{aligned}
\rho &= \frac{\sigma_a^2}{\sigma^2} \quad \text{if } i = i' \text{ and } j \neq j', \\
\rho &= \frac{\sigma^2}{\sigma^2} \quad \text{if } i = i' \text{ and } j = j', \\
\rho &= \frac{0}{\sigma^2} \quad \text{if } i \neq i' \text{ and } j \neq j'.
\end{aligned}$$

Thus, we achieve the following correlation matrix for $\mathbf{s}$:

$$\text{Corr}(\mathbf{s}) = \begin{array}{c} \\ s_{12} \\ \vdots \\ s_{1n} \\ s_{23} \\ \vdots \\ s_{n-1,n} \end{array} \begin{array}{c} \begin{array}{cccccc} s_{12} & \dots & s_{1n} & s_{23} & \dots & s_{n-1,n} \end{array} \\ \begin{pmatrix} 1 & \dots & \frac{\sigma_a^2}{\sigma^2} & \frac{\sigma_a^2}{\sigma^2} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\sigma_a^2}{\sigma^2} & \dots & 1 & 0 & \dots & 0 \\ \frac{\sigma_a^2}{\sigma^2} & \dots & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{pmatrix} \end{array}$$

We can see that the matrix dimensions are $N \times N$. The value for $\text{Corr}(\mathbf{S}_{ij})$ depends on the values of i and j as pointed out directly above. (Note that for the above illustration, we assume that $n > 3$.

Noting that we defined $\sigma^2 = 2\sigma_a^2 + \sigma_e^2$, we can see that the covariance matrix

$\mathbf{\Sigma}$ can be written as

$$\mathbf{\Sigma} = \sigma_a^2 \text{Corr}(\mathbf{s}) = \begin{array}{c} \\ s_{12} \\ \vdots \\ s_{1n} \\ s_{23} \\ \vdots \\ s_{n-1,n} \end{array} \begin{array}{c} \begin{array}{cccccc} s_{12} & \dots & s_{1n} & s_{23} & \dots & s_{n-1,n} \end{array} \\ \begin{pmatrix} \sigma^2 & \dots & \sigma_a^2 & \sigma_a^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sigma_a^2 & \dots & \sigma^2 & 0 & \dots & 0 \\ \sigma_a^2 & \dots & 0 & \sigma^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & \sigma^2 \end{pmatrix} \end{array}$$

If we want to express this in matrix form, then define a matrix $\mathbf{A}$ of dimension $N \times N$ that has zeroes in the same places as the correlation matrix, and 1's everywhere else. Then we can write $\mathbf{\Sigma} = \mathbf{A}\sigma_a^2 + \mathbf{I}(\sigma_a^2 + \sigma_e^2)$.

Now we can see that if we take $\mathbf{PP}^T$, we get a similar structure with values of 2 along the diagonal, and zeros and ones in the same positions as the correlation matrix. An example is shown below. With of $n = 6$, the $\mathbf{PP}^T$ has dimensions of

$N \times N$, or 15×15 and takes the form of

$$\mathbf{PP}^T = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 2 \end{pmatrix}$$

We have found that it is simpler to perform the spectral decomposition of $\mathbf{\Sigma}$ when it takes the form of $\sigma_a^2 \mathbf{PP}^T + \mathbf{I}\sigma_e^2$.

Perhaps a better illustration of the form the covariance matrix takes can be seen using the heat maps in figure 3. The pattern begins to emerge as the number of objects increases. White indicates values of 0, gray indicates values of 1, and black indicates values of 2.

The covariance matrices grow increasingly sparse as the number of objects increases. Table 2 below shows the number of each kind of value and the proportion of the $\mathbf{PP^T}$ matrix that it holds. As the number of objects increases, the number of 'zero' entries increases rapidly. As the number of zeros approaches infinity, the

	0		1		2		total
$n = 5$	30,	30%	60,	60%	10,	10%	100
$n = 10$	1260,	62%	720,	36%	45,	2%	2025
$n = 20$	29070,	81%	6840,	19%	190,	1%	36100
$n = 30$	164430,	87%	24360,	13%	435,	0%	189225
$n = 40$	548340,	90%	59280,	10%	780,	0%	608400
$n = 60$	2925810,	93%	205320,	7%	1770,	0%	3132900
$n = 100$	23527350,	96%	970200,	4%	4950,	0%	24502500

Table 2: Ratio of values in $\mathbf{PP^T}$

proportion of zeros to non-zero values will never reach one hundred percent, but it will continuously approach it.

1.5 Covariance Matrix Eigen Decomposition. The spectral decomposition is as follows: First we start with the property that for a square matrix $\mathbf{A}$, $\mathbf{A}\mathbf{v}_k = \lambda_k \mathbf{v}_k$ where $\mathbf{v}_k$ and λ_k are a unique eigenvector/eigenvalue pair. Since $\mathbf{\Sigma}$ is a linear combination of $\mathbf{PP}^T$ and the identity matrix multiplied by a constant, then the eigenvectors/values found for $\mathbf{PP}^T$ can be used to find the eigenvectors/values for $\mathbf{\Sigma}$ in the following way,

$$\begin{aligned} \mathbf{\Sigma}\mathbf{v}_k &= (\sigma_a^2 \mathbf{PP}^T + \sigma_e^2 \mathbf{I})\mathbf{v}_k \\ &= \sigma_a^2 \lambda_k \mathbf{v}_k + \sigma_e^2 \mathbf{I}\mathbf{v}_k \\ &= \sigma_a^2 \lambda_k \mathbf{v}_k + \sigma_e^2 \mathbf{v}_k \\ &= (\sigma_a^2 \lambda_k + \sigma_e^2)\mathbf{v}_k, \end{aligned} \tag{9}$$

where λ_k is the eigenvalue for the $\mathbf{PP}^T$ matrix. This important result tells us that if we calculate the eigenvalue for the matrix $\mathbf{PP}^T$, then the corresponding eigenvalue for $\mathbf{\Sigma}$ is equal to $\sigma_a^2 \lambda_k + \sigma_e^2$.

We will now show that there are three unique eigenvalues for $\mathbf{\Sigma}$. We make the process simpler by noting that $\mathbf{PP}^T$ and its transpose $\mathbf{P}^T\mathbf{P}$ have the same rank and the same eigenvalue/eigenvalue sets. $\mathbf{P}^T\mathbf{P}$ has the form of $n - 1$ along the diagonal,

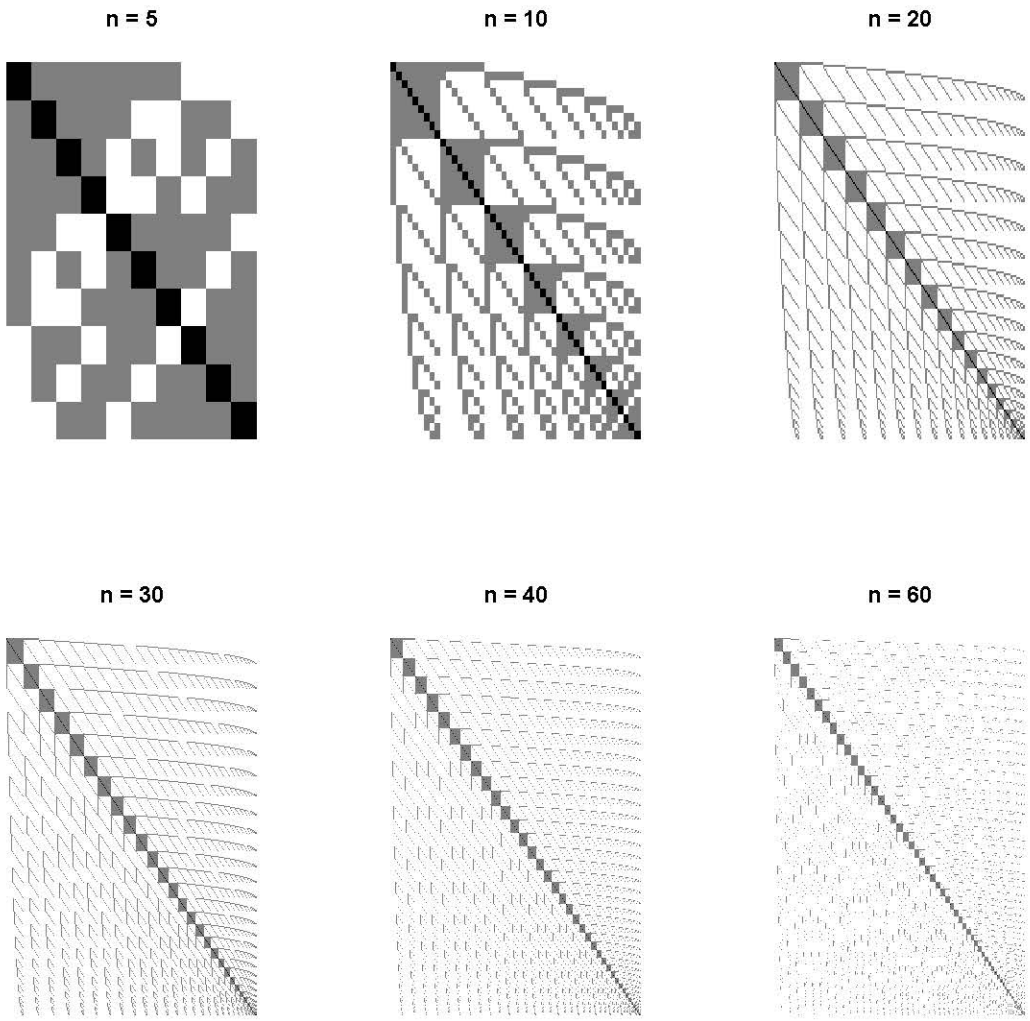


Figure 3: The heat maps of PP^T with n = 5, 10, 20, 30, 40, 60.

and ones everywhere else as shown below.

$$\mathbf{P}^T\mathbf{P} = \begin{pmatrix} n-1 & & 1 \\ & \ddots & \\ 1 & & n-1 \end{pmatrix}$$

We note that $\mathbf{P}^T\mathbf{P}$ has dimensions of $n \times n$. We can write this in matrix form as $(n-2)\mathbf{I}_n + \mathbf{1}_n\mathbf{1}_n^T$ where $\mathbf{1}_n$ is a vector of ones of length n. We have shown before

that the eigenvalues for a matrix in the form of $\mathbf{A} + b\mathbf{I}$ will have eigenvalues equal to $\lambda = \lambda_a + b$. In this situation, the matrix $\mathbf{1}\mathbf{1}^T$ has one eigenvalue equal to n with the matching eigenvector equal to $\frac{\mathbf{1}}{\mathbf{1}^T\mathbf{1}} = \frac{\mathbf{1}}{\sqrt{N}}$. There are $n-1$ remaining eigenvalues of zero with eigenvectors orthogonal to $\mathbf{1}_n$. Therefore, the eigenvalues for $(n-2)\mathbf{I}_n + \mathbf{1}_n\mathbf{1}_n^T$ are

$$\lambda_1 = n + (n-2) = 2(n-1) \text{ with eigenvector } \frac{\mathbf{1}}{\sqrt{N}},$$
$$\lambda_2, \ldots, \lambda_n = 0 + (n-2) = n-2.$$

We note that $\mathbf{P}\mathbf{P}^T$ will have N eigenvectors/eigenvalues; but is not full rank. There are n non-zero eigenvalues and $N-n$ eigenvalues equal to zero with eigenvectors equal to the zero vector due to orthogonality.

Applying the spectral decomposition found in Equation (9) on page 26 to our covariance matrix $\mathbf{\Sigma}$, we get

- Eigenvector $\mathbf{v}_1 = \frac{1}{\sqrt{N}}$ with corresponding eigenvalue $\lambda_1 = \sigma_e^2 + 2(n-1)\sigma_a^2$.
- $n-1$ eigenvectors $\mathbf{v}_2$ through $\mathbf{v}_n$ with matching eigenvalue $\lambda_2 = \sigma_e^2 + (n-2)\sigma_a^2$.
- $N-n$ eigenvectors $\mathbf{v}_{n+1}$ through $\mathbf{v}_N$ with eigenvalue $\lambda_3 = \sigma_e^2$.

Because $\sigma_e^2 > 0$, then $\mathbf{\Sigma}$ will have full rank. Also, because eigenvectors are orthogonal, then we will have $\mathbf{v'}_k\mathbf{1}_N = 0, \text{ for all } k > 1$.

We will find that is useful to work with the sums of the eigenvectors in the form of $\mathbf{v}\mathbf{v}^T$. We continue with the spectral decompositions of $\mathbf{I}$, $\mathbf{P}\mathbf{P}^T$, and $\mathbf{\Sigma}$. Since

the eigenvectors are orthogonal, we have

$$\begin{aligned}
\mathbf{I} &= \mathbf{v}_1\mathbf{v}^T{}_1 + \sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}_k^T + \sum_{l=n+1}^{N}\mathbf{v}_l\mathbf{v}^T{}_l. \\
\mathbf{PP}^T &= 2(n-1)\mathbf{v}_1\mathbf{v}^T{}_1 + (n-2)\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}^T{}_k + 0\cdot\sum_{l=n+1}^{N}\mathbf{v}_l\mathbf{v}^T{}_l \\
&= 2(n-1)\mathbf{v}_1\mathbf{v}^T{}_1 + (n+2)\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}^T{}_k. \\
\mathbf{\Sigma} &= (\sigma_e^2 + 2(n-1)\sigma_a^2)\mathbf{v}_1\mathbf{v}^T{}_1 + (\sigma_e^2 + (n-2)\sigma_a^2)\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}^T{}_k + \sigma_e^2\sum_{l=n+1}^{n}\mathbf{v}_l\mathbf{v}^T{}_l.
\end{aligned}$$

So we know from the preceding work that $\mathbf{PP}^T$ can be used to find

$$\begin{aligned}
\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}'{}_k &= \frac{\mathbf{PP}^T - 2(n-1)\mathbf{v}_1\mathbf{v}_1^T}{n-2} \\
&= \frac{\mathbf{PP}^T - 2(n-1)\frac{1}{N}\mathbf{1}\mathbf{1}^T}{n-2}.
\end{aligned}$$

We know that we can obtain this value since we know the values in the above equation. Subbing this into the spectral decomposition of $\mathbf{I}$, we can find the third sum of eigenvectors as

$$\begin{aligned}
\sum_{l=n+1}^{N}\mathbf{v}_l\mathbf{v}^T{}_l &= \mathbf{I} - \mathbf{v}_1\mathbf{v}^T{}_1 - \sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}_k^T \\
&= \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T - \frac{\mathbf{PP}^T - 2(n-1)\frac{1}{N}\mathbf{1}\mathbf{1}^T}{n-2}
\end{aligned}$$

We'll see how these eigenvector sums will be useful when constructing our multivariate likelihood function.

Because we are assuming that a_i and ϵ_{ij} values are normally distributed, we can write out the log-likelihood function for a multivariate normal density that is

appropriate for our model.

$$\ln(L) = -\frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}(\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1}) - \frac{N}{2}\ln(2\pi).$$

Noting that $|\mathbf{\Sigma}| = \prod_{i=1}^{N}\lambda_i = \lambda_1\lambda_2^{n-1}\lambda_3^{N-n}$ and that

$$\ln(\lambda_1\lambda_2^{n-1}\lambda_3^{N-n}) = \ln(\lambda_1) + (n-1)\ln(\lambda_2) + (N-n)\ln(\lambda_3),$$

we now have

$$-2\ln(L) = \ln(\lambda_1) + (n-1)\ln(\lambda_2) + (N-n)\ln(\lambda_3) + (\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1}) + N\ln(2\pi).$$

Now analyzing $(\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1})$, we have

$$\begin{aligned}(\mathbf{s} - \theta\mathbf{1})^T\mathbf{\Sigma}^{-1}(\mathbf{s} - \theta\mathbf{1}) =& (\mathbf{s} - \theta\mathbf{1})^T\sum_{k=1}^{N}\lambda_k^{-1}\mathbf{v}_k\mathbf{v}_k^T(\mathbf{s} - \theta\mathbf{1}) \\ =& (\mathbf{s} - \theta\mathbf{1})^T\lambda_1^{-1}\mathbf{v}_1\mathbf{v}_1^T(\mathbf{s} - \theta\mathbf{1}) \\ &+ (\mathbf{s} - \theta\mathbf{1})^T\sum_{k=2}^{n}\lambda_k^{-1}\mathbf{v}_k\mathbf{v}_k^T(\mathbf{s} - \theta\mathbf{1}) \\ &+ (\mathbf{s} - \theta\mathbf{1})^T\sum_{l=n+1}^{N}\lambda_l^{-1}\mathbf{v}_l\mathbf{v}_l^T(\mathbf{s} - \theta\mathbf{1}).\end{aligned}$$

First, we will analyze the first term of these sums:

$$
\begin{aligned}
(\mathbf{s}-\theta\mathbf{1})^T\lambda_1^{-1}\mathbf{v}_1\mathbf{v}_1(\mathbf{s}-\theta\mathbf{1}) &= \frac{(\mathbf{s}-\theta\mathbf{1})^T\mathbf{v}_1\mathbf{v}_1(\mathbf{s}-\theta\mathbf{1})}{\lambda_1} \\
&= \frac{\left(\mathbf{s}^T\frac{\mathbf{1}}{\sqrt{N}}-\frac{\theta\mathbf{1}^T\mathbf{1}}{\sqrt{N}}\right)^2}{\lambda_1} \\
&= \frac{\mathbf{s}^T\frac{\mathbf{1}}{\sqrt{N}}\mathbf{s}^T\frac{\mathbf{1}}{\sqrt{N}}-\left(\mathbf{s}^T\frac{\mathbf{1}}{\sqrt{N}}\right)\left(\frac{\theta\mathbf{1}^T\mathbf{1}}{\sqrt{N}}\right)-\left(\frac{\theta\mathbf{1}^T\mathbf{1}}{\sqrt{N}}\right)\left(\mathbf{s}^T\frac{\mathbf{1}}{\sqrt{N}}\right)+\left(\frac{\theta\mathbf{1}^T\mathbf{1}}{\sqrt{N}}\right)\left(\frac{\theta\mathbf{1}^T\mathbf{1}}{\sqrt{N}}\right)}{\lambda_1} \\
&= \frac{\left(\frac{\sum_{i=1}^N s_i}{\sqrt{N}}\right)^2-\left(\frac{\sum_{i=1}^N s_i}{\sqrt{N}}\right)\left(\frac{N\theta}{\sqrt{N}}\right)-\left(\frac{N\theta}{\sqrt{N}}\right)\left(\frac{\sum_{i=1}^N s_i}{\sqrt{N}}\right)+\left(\frac{N\theta}{\sqrt{N}}\right)\left(\frac{N\theta}{\sqrt{N}}\right)}{\lambda_1} \\
&= \frac{\frac{\left(\sum_{i=1}^N s_i\right)^2}{N}-\theta\sum_{i=1}^N s_i-\theta\sum_{i=1}^N s_i+N\theta^2}{\lambda_1} \\
&= \frac{\frac{\left(\sum_{i=1}^N s_i\right)^2}{N}-2\theta\sum_{i=1}^N s_i+N\theta^2}{\lambda_1} \\
&= \frac{N\left(\frac{\left(\sum_{i=1}^N s_i\right)^2}{N^2}-\frac{2\theta\sum_{i=1}^N s_i}{N}+\theta^2\right)}{\lambda_1} \\
&= \frac{N(\bar{s}^2-2\theta\bar{s}+\theta^2)}{\lambda_1} \\
&= \frac{N(\bar{s}-\theta)^2}{\lambda_1} \\
&= \frac{SS_\theta}{\lambda_1}.
\end{aligned}
$$

Regarding the other two terms, we have

$$
\begin{aligned}
\frac{(\mathbf{s}-\theta\mathbf{1})^T\sum_{k=2}^n\mathbf{v}_k\mathbf{v}_k^T(\mathbf{s}-\theta\mathbf{1})}{\lambda_2} &= \frac{\mathbf{s}^T\left(\sum_{k=2}^n\mathbf{v}_k\mathbf{v}_k^T\right)\mathbf{s}}{\lambda_2} = \frac{SS_a}{\lambda_2}, \text{ and} \\
\frac{(\mathbf{s}-\theta\mathbf{1})^T\sum_{l=n+1}^N\mathbf{v}_l\mathbf{v}_l^T(\mathbf{s}-\theta\mathbf{1})}{\lambda_3} &= \frac{\mathbf{s}^T\left(\sum_{l=n+1}^N\mathbf{v}_l\mathbf{v}_l^T\right)\mathbf{s}}{\lambda_3} = \frac{SS_e}{\lambda_3}.
\end{aligned}
$$

So finally, we have our log likelihood as

$$
-2\ln(L) = N\ln(2\pi)+\ln(\lambda_1)+(n-1)\ln(\lambda_2)+(N-n)\lambda_3+\frac{SS_\theta}{\lambda_1}+\frac{SS_a}{\lambda_2}+\frac{SS_e}{\lambda_3}.
$$

By the spectral decomposition of $\mathbf{P}\mathbf{P}^T$, we obtain a useful property.

$$\begin{aligned}\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}^T{}_k &= \frac{\left(\mathbf{P}\mathbf{P}^T - 2(n-1)\mathbf{v}_1\mathbf{v}^T{}_1\right)}{n-2}\\ &= \frac{1}{n-2}\left(\mathbf{P}\mathbf{P}^T - \frac{4}{n}\mathbf{1}_N\mathbf{1}^T{}_N\right)\\ &= \frac{(n-1)^2}{n-2}\cdot\frac{1}{(n-1)^2}\left(\mathbf{P}\mathbf{P}^T - \frac{4}{n}\mathbf{1}_N\mathbf{1}^T{}_N\right)\\ &= \frac{(n-1)^2}{n-2}\left(\frac{1}{(n-1)^2}\mathbf{P}\mathbf{P}^T + \left(\frac{4n}{n^2(n-1)^2} - \frac{8}{n(n-1)^2}\right)\mathbf{1}_N\mathbf{1}^T{}_N\right);\end{aligned}$$

noting that $\frac{n(n-1)}{2} = N$, we continue with

$$\begin{aligned}&\sum_{k=2}^{n}\mathbf{v}_k\mathbf{v}^T{}_k = \frac{(n-1)^2}{n-2}\left(\frac{1}{(n-1)^2}\mathbf{P}\mathbf{P}^T - \frac{4}{N(n-1)}\mathbf{1}_N\mathbf{1}^T{}_N + \frac{n}{N^2}\mathbf{1}_N\mathbf{1}^T{}_N\right)\\ &= \frac{(n-1)^2}{n-2}\left(\frac{1}{(n-1)^2}\mathbf{P}\mathbf{P}^T - \frac{1}{N(n-1)}2\cdot\mathbf{1}_n\mathbf{1}^T{}_N - \frac{1}{N(n-1)}2\cdot\mathbf{1}_N\mathbf{1}^T{}_N + \frac{n}{N^2}\mathbf{1}_N\mathbf{1}^T{}_N\right)\\ &= \frac{(n-1)^2}{(n-2)}\left(\frac{1}{(n-1)^2}\mathbf{P}\mathbf{P}^T - \frac{1}{N(n-1)}\mathbf{P}\mathbf{1}_n\mathbf{1}^T{}_N - \frac{1}{N(n-1)}\mathbf{1}_N\mathbf{1}^T{}_n\mathbf{P}^T + \frac{1}{N^2}\mathbf{1}_N\mathbf{1}^T{}_n\mathbf{1}_n\mathbf{1}^T{}_N\right)\\ &= \frac{(n-1)^2}{n-2}\left(\frac{1}{n-1}\mathbf{P} - \frac{1}{N}\mathbf{1}_N\mathbf{1}^T{}_n\right)\left(\frac{1}{n-1}\mathbf{P}^T - \frac{1}{N}\mathbf{1}_n\mathbf{1}^T{}_N\right).\end{aligned}$$

We can see that if $\bar{s}^{(k)} = \frac{1}{n-1}\sum_{(i \text{ or } j=k)} s_{ij}$, we can get the following vectors

$$\frac{1}{n-1}\mathbf{P}^T\mathbf{s} = \begin{bmatrix}\bar{s}^{(1)}\\ \bar{s}^{(2)}\\ \vdots\\ \bar{s}^{(n)}\end{bmatrix} \text{ and } \frac{1}{N}\mathbf{1}_N^T\mathbf{s} = \bar{s}, \text{ then } \frac{1}{n-1}\mathbf{P}^T\mathbf{s} - \frac{1}{N}\mathbf{1}^T{}_N = \begin{bmatrix}\bar{s}^{(1)} - \bar{s}\\ \bar{s}^{(2)} - \bar{s}\\ \vdots\\ \bar{s}^{(n)} - \bar{s}\end{bmatrix}$$

Therefore, our sum-of-squares SS_a must be

$$SS_a = \frac{(n-1)^2}{n-2}\sum_{k=1}^{n}\left(\bar{s}^{(k)} - \bar{s}\right)^2.$$

Source	df	Sum of Squares	Mean Square	Expected Value of Mean Square
A	$n-1$	SS_a	$MS_a = SS_a/(n-1)$	$\sigma_e^2 + (n-2)\sigma_a^2$
Error	$N-n$	SS_e	$MS_e = SS_e/(N-n)$	σ_e^2
Total	$N-1$	SS_t	$MS_t = SS_t/(N-1)$	

Table 3: ANOVA Table

As we have seen before,

$$\begin{aligned}\sum_{l=n+1}^{N} \mathbf{v}_l \mathbf{v}^T{}_l &= \mathbf{I} - \mathbf{v}_1 \mathbf{v}^T{}_1 - \sum_{k=2}^{n} \mathbf{v}_k \mathbf{v}^T{}_k \\ &= \mathbf{I} - \frac{1}{N} \mathbf{1}_N \mathbf{1}^T{}_N - \sum_{k=2}^{n} \mathbf{v}_k \mathbf{v}^T{}_k.\end{aligned}$$

If we now define the total sum-of-squares to be $SS_t = \mathbf{s}^T \left(\mathbf{I} - \mathbf{v}_1 \mathbf{v}^T{}_1\right) \mathbf{s}$, then we can write the error sum-of-squares as $SS_e = \mathbf{s}^T \sum_{l=n+1}^{N} \mathbf{v}_l \mathbf{v}^T{}_l \mathbf{s} = SS_t - SS_a$.

Because SS_a and SS_e are defined using the characteristic vectors of $\mathbf{\Sigma}$, then by Cochran's Theorem[10], we have independence between SS_a and SS_e with degrees of freedom $n-1$ and $N-n$ respectively. (Note that $SS_t = SS_a + SS_e$). Each of the SS's are independent of the sample mean. It can also be seen that $E(\bar{s}) = \theta, Var(\bar{s}) = \frac{\sigma_e^2}{N} + \frac{4\sigma_a^2}{n}, E(SS_a) = (n-1)(\sigma_e^2 + (n-2)\sigma_a^2),$ and $E(SS_e) = (N-n)\sigma_e^2$. This gives us the ANOVA table shown in Table 3.

This allows us to retrieve REML plug-in unbiased estimators for the necessary parameters in our multivariate normal density function, as derived by [48], shown as:

$$\begin{aligned}\hat{\theta} &= \bar{s}. \\ \hat{\sigma}_a^2 &= MS_a - \frac{MS_e}{n-2}. \\ \hat{\sigma}_e^2 &= MS_e.\end{aligned} \tag{10}$$

1.6 Conditional Likelihood. When computing the atypicality value during the bootstrap, we use a conditional multivariate normal pdf. It is the probability of the sub-vector of scores that are within the unknown objects and between the known and unknown objects (denoted as $\vec{s_0}$), conditional on the scores between the known objects being equal to what they are ($\vec{s_X} = \vec{s_x}$). We can write this probability density function as $f\left(\vec{s_0}|\vec{s_{X=s_x}}\right)$. However, since the parameters θ for this distribution are estimated using only data found in $\vec{s_x}$, we write the pdf as $f\left(\vec{s_0}|\theta\right)$ throughout this paper for simplicity. The following section describes the process of deriving the conditional multivariate normal parameters θ that make up the conditional distribution.

Let the vector of known i.i.d. objects be represented by $\mathbf{X}_n = (X_1, \ldots, X_n)^T$, and $\mathbf{Y}_m = (Y_1, \ldots, Y_m)^T$ represent a different vector of unknown i.i.d objects of size m. We do not assume to know the distribution for either vector of these objects. Now, let the vector of all the objects be given as $Z_{n+m} = (Y_m + X_n)^T$.

Next, we use an appropriate kernel obtain the vector of scores between all objects, shown as $\vec{s}$. We assume $\vec{s}$ has a conditional multivariate normal distribution with unknown parameters. To obtain the conditional parameters, we then discern the score vector into two groups. The first group pertains to the scores that are within the unknown objects Y *and* between the unknown and known objects X. Let this subset of the score vector be known as $\vec{s_0}$. The remaining scores are within the group of known objects. Let's denote this subset as $\vec{s_X}$. For the conditional multivariate normal distribution, we have $f\left(\vec{s_0}|\vec{s_{X=s_x}}\right)$.

As an example, let $Y_2 = (y_1, y_2)$ and $X_3 = (x_3, x_4, x_5)$. Then the vector $Z_5 = (y_1, y_2, x_3, x_4, x_5)$; and the corresponding score vector $\vec{s} = (s_{12}, s_{13}, s_{14}, s_{15}, s_{23}, s_{24}, s_{25}, s_{34}, s_{35}, s_{45})^T$. Then the subset of scores $\vec{s_0} = (s_{12}, s_{13}, s_{14}, s_{15}, s_{23}, s_{24}, s_{25})^T$, and $s_X = (s_{34}, s_{35}, s_{45})$.

The number of scores in $\vec{s_0}$ and $\vec{s_x}$ is dependent on the number of objects in

X_n and Y_m. The number of scores withing the unknown objects is $\binom{m}{2}$. The number of scores between the known and unknown objects is $m \times n$. Since $\vec{s_0}$ is the vector of scores within the unknown objects and the scores between the unknown scores, the number of scores in $\vec{s_0}$ is $\binom{m}{2} + mn$. The number of scores in $\vec{s_x}$ is simply $\binom{n}{2}$. This means that the total number of scores $N = \binom{m}{2} + mn + \binom{n}{2}$.

Now that we have our score vector separated into two groups, we use Izenman's approach to calculate the conditional mean and covariance matrix [29].

To simplify the notation for the following work, let Σ_X denote the covariance matrix for $\vec{s_x}$, Σ_0 be the covariance matrix for $\vec{s_0}$, and Σ_{X0} contain the covariances for the $\vec{s_x}$ and $\vec{s_0}$ sub-vectors.

While Σ_X and Σ_0 are square matrices, Σ_{X0} may not necessarily be so. Let $\Sigma_{0X} = (\Sigma_{X0})^T$. For the derivation of the covariance matrix, we use quadrant's of $\mathbf{\Sigma}$ to get the necessary $\Sigma_X, \Sigma_{X0}, \Sigma_{0X}, \Sigma_0$ values, as specified by [29]. This is shown as

$$\Sigma = \begin{pmatrix} \Sigma_0 & \Sigma_{0X} \\ \Sigma_{X0} & \Sigma_X \end{pmatrix}$$

This means that the dimensionality of the covariance matrix will be $N = \binom{m+n}{2} = \binom{m}{2} + mn + \binom{n}{2}$.

Σ_X is a square $\binom{n}{2}$ matrix in the lower-right quadrant of Σ. Σ_0 is a $\binom{m}{2} + mn$ square matrix that is the top-left quadrant. Σ_{X0} is the lower-left quadrant with $\binom{n}{2}$ rows and $\binom{m}{2} + mn$ columns. Finally, Σ_{0X} is the top right quadrant of the covariance matrix and is simply the transpose of Σ_{X0}.

As an example, let $n = 5, m = 1$. Instead of Σ, we will use $\mathbf{PP}^T$ for the visual

aid to keep the numeric values simple. Below, $\mathbf{PP}^T$ divided into the four quadrants.

$$\mathbf{PP}^T = \left[\begin{array}{ccccc|ccccccccccc} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 2 \end{array}\right]$$

So our quadrants are given as

$$\mathbf{PP}_0^T = \left[\begin{array}{ccccc} 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{array}\right]$$

$$\mathbf{PP}^T_{0X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$\mathbf{PP}^T_{X0} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\mathbf{PP}^T_{X} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

From here, we replace all the 2 values with $2\sigma_a^2 + \sigma_e^2$ and the 1 values with σ_a^2 to get

our appropriate $\mathbf{\Sigma}$ covariance matrix.

From [29], the conditional mean and covariance matrix are shown as

$$\mu_{0|X} = \bar{s_x} + \Sigma_{0X}\Sigma_X^{-1}(\vec{s_x} - \bar{s_x}),$$

$$\Sigma_{0|X} = \Sigma_0 - \Sigma_{0X}\Sigma_X^{-1}\Sigma_{X0}.$$

$$\theta = \{\mu_{0|X}, \Sigma_{0|X}\}$$

A detailed derivation of these parameters can be found in Izenman's text. But we can see that the conditional mean is the mean of $\vec{s_x}$ plus an adjustment. The adjustment involves the covariances between $\vec{s_0}$ and $\vec{s_x}$ and the inverse of the covariance matrix for s_x, and the difference between the $\vec{s_x}$ vector and its mean. Therefore, the conditional mean is a function of $\vec{s_x}$. Typically , one would use $\bar{s_0}$ as the first term in this equation, but $\bar{s_x}$ was chosen instead to make atypicality's hypothesis test more powerful. By using $\bar{s_x}$, probability values given by $f\,(\vec{s_0}|\theta)$ will be smaller as $\vec{s_0}$ is further away from the mean of $\vec{s_x}$, making it harder to fail to reject H_0.

These parameters clearly have the same dimension as $\vec{s_0}$. Also, these parameters are used in the bootstrap to sample a new vector of scores $\vec{S^*}$ each iteration. Therefore, $\vec{S^*}$ and $\vec{s_0}$ will have the same dimensionality, and will use the same conditional parameters in the computation of atypicality $Pr\{f\left(\vec{S^*}|\theta\right) \leq f\,(\vec{s_0}|\theta)\}$. This is an important fact because, as the dimensionality increases, probability values from pdfs tend to decrease in general. This can be seen by looking at Table 4 below. The Table represents taking the pdf of the standard multivariate normal distribution for vectors of zeros of different dimensions.

This means that if one of the pdf calculations has a smaller dimensionality, then it naturally tends to have a higher value, yielding what might be considered skewed atypicality results. By utilizing the conditional multivariate normal

Dimension of zero vector	Pdf Value
1	0.399
2	0.159
3	0.063
4	0.025
5	0.010
6	0.004
7	0.002
8	0.001
9	0.000
10	0.000

Table 4: Pdf values of vectors of zeros. The diminishing pdf value illustrates the natural phenomenon that as dimensions increase, the pdf values decrease in general.

distribution, we can maintain equal dimensionality comparisons. The pseudo code in Algorithm 3 is the bootstrap portion replicated here for the reader's convenience.

```
sum ← 0;
for j ← 1 to NumSamples do
    Samples ← get.scores(CondSig, CondMu);
    if f(Samples, CondSig, CondMu) ≤
     f(scores.unknown, CondSig, CondMu) then
        sum ← sum + 1;
    end
end
atypicality.vector[i] ← sum/NumSamples;
```

Algorithm 3: The bootstrap portion of the algorithm demonstrating the use of the conditional parameters.

1.7 Example of Atypicality Application. Here we will show some examples of finding the atypicality values from fabricated data. Further chapters will use real world data. To start, we will create a population of random objects from a standard normal distribution, and then introduce a new object from a normal distribution with differing mean and variance.

We will start with $n = 8$ known objects sampled from a one dimensional standard normal distribution, and then a new object will be introduced with a value starting at zero, increasing by one for each new simulation. We predict that the

atypicality value should get smaller as the value of the unknown objects moves away from the others. For all the simulations, the eight known objects were sampled as: -0.626, 0.184, -0.836, 1.595, 0.330, -0.820, 0.487, and 0.738. Let the unknown observation be represented by y.

y	0.0	0.5	1.0	1.5	2.0	2.5	3.0
Atypicality	0.942	0.234	0.775	0.227	0.093	0.008	0.000

Table 5: Table showing how atypicality tends to approach zero as the unknown object moves away from the other objects.

We can see that as the value of X_n is moved farther from zero, it's atypicality value tends to decrease. The exception is when the unknown object is equal to 0.5. But, if we were to perform a hypothesis test using atypicality with an alpha set to 0.05, then we wouldn't choose to reject H_0 until the unknown object is equal to 2.5.

The R code for an example using tool mark data is in the Appendix. According to the paper that was published with this data, they found that the 16th object was the most unlike the others by their metric [41]. Below is the atypicality R output.

```
[1] "2Afullt.txt  -  1 : 0.9707"
[1] "2Afullt.txt  -  2 : 0.963"
[1] "2Afullt.txt  -  3 : 0.9007"
[1] "2Afullt.txt  -  4 : 0.9123"
[1] "2Afullt.txt  -  5 : 0.9509"
[1] "2Afullt.txt  -  6 : 0.0023"
[1] "2Afullt.txt  -  7 : 0.8226"
[1] "2Afullt.txt  -  8 : 0.6686"
[1] "2Afullt.txt  -  9 : 0.9854"
[1] "2Afullt.txt  -  10 : 0.879"
[1] "2Afullt.txt  -  11 : 0.9939"
[1] "2Afullt.txt  -  12 : 0.8902"
[1] "2Afullt.txt  -  13 : 0.041"
[1] "2Afullt.txt  -  14 : 0.3598"
[1] "2Afullt.txt  -  15 : 0.7536"
[1] "2Afullt.txt  -  16 : 0"
[1] "2Afullt.txt  -  17 : 0.2511"
[1] "2Afullt.txt  -  18 : 0.5202"
[1] "2Afullt.txt  -  19 : 0.8236"
[1] "2Afullt.txt  -  20 : 0.9756"
```

As we can see, the 16th object has an atypicality value of zero, matching the authors of the data.

2. THE POWER OF ATYPICALITY

This chapter shows the power of the atypicality measure under different conditions. The power of a hypothesis test is the probability that the test correctly rejects the null hypothesis H_0 when the alternative hypothesis H_1 is true. We remind the reader that the hypotheses are:

H_0 : Two independent simple random samples are from a given known source.

H_1 : Not H_0.

Primarily, we will focus on the conditional atypicality score. We found that when generating scores between standard normal objects, we weren't getting scores that were multivariate normal. The discovery was made by rotating the cloud of scores in three dimensional space; showing a non-normal skew depending on the view. See Figure 4 for a visual of this phenomenon.

The cloud was created by first randomly sampling a three-dimensional, standard multivariate normal variable $X \sim MVN\left(\vec{0}_3, I_3\right)$. Then, a score vector was created by squaring the difference between each dimension's value, and dividing by two, yielding a chi-squared variable. We divide by two because that would be the variance of the chi-squared variable in the numerator. By calculating the probability of this chi-squared variable using the cumulative distribution function $F_{\chi^2(1)}$, and then the using that probability as input for the normal quantile function $Q_{N(0,1)}$, we will get a normal variable, which will be used as a score value. The vector of these scores is a standard MVN sample.

We then repeat this process a large number of times to create a large sample of these MVN vectors. Principal component analysis is then performed to obtain the component scores. The scatterplot shown on the left side of Figure 4 is of the component scores of the first and third dimensions. This appears to be a normal

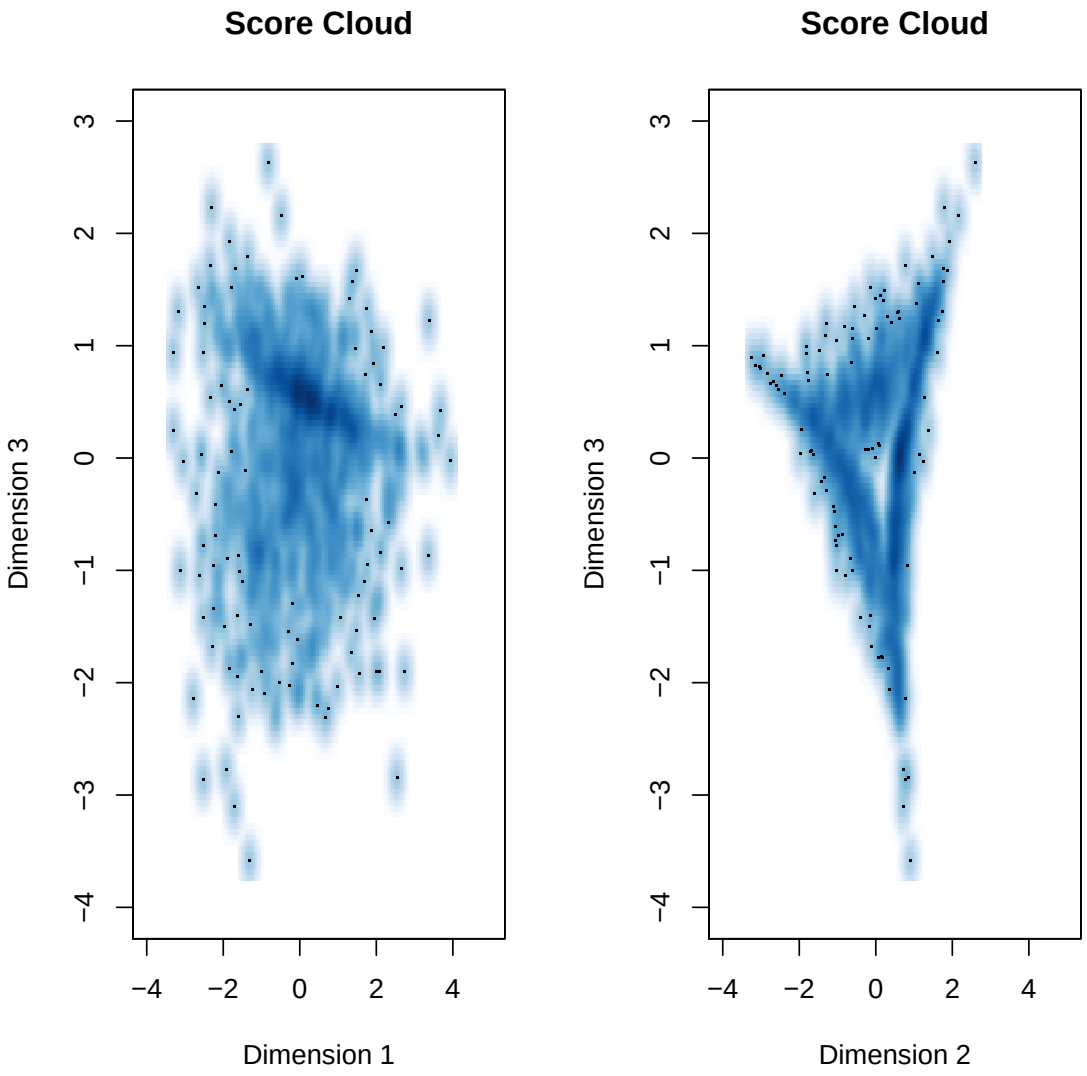


Figure 4: Scatterplots comparing the principal component scores of fabricated MVN vectors. The left plot appears to be multivariate normal, but the right is skewed.

density cloud. The scatterplot on the right is of the component scores for the second and third dimensions. Here, the pattern of the density cloud does not appear normal, but has a triangular pattern with a hollow center.

$$s_{ij} = Q_{N(0,1)}\left(F_{\chi^2(1)}\left(\frac{(X_i - X_j)^2}{2}\right)\right), i < j < 3.$$

$$\vec{s} = \{s_{12}, s_{13}, s_{23}\}$$

In order to alleviate this problem, we increased the dimensionality of the sampled objects to $2n + \binom{n}{2}$. There may be a more specific dimensionality to sample from, but we leave that to be explored in future work.

The Figures 5, 7, and 6 show various situations where we have different numbers of objects from the original known population, and then introduce new objects. We need a minimum of four known population objects and one introduced object in order for the algorithm to run properly. A maximum of forty total objects (both known and introduced) is computationally convenient to run in simulations, so that is the max shown here.

The simulations presented start with a known group of objects and a smaller to equal sized group of introduced objects. The distribution for the known objects will be set as:

$$X \sim N(0, 1).$$

The distribution for the introduced objects will have a set variance of one, but the we will move the mean away with each new simulation. A new set of known and introduced objects will be randomly selected from these distributions and the atypicality value will be computed. We will expect near uniform distributions of atypicality scores when both groups of objects have the same distribution; that is to say, when they have the same mean. We would then expect the atypicality scores to be closer to zero as the we move the mean of the introduced objects further away from zero. The simulations will run one thousand times for each set of distributions, and an empirical CDF and distribution plot will be shown to illustrate the results.

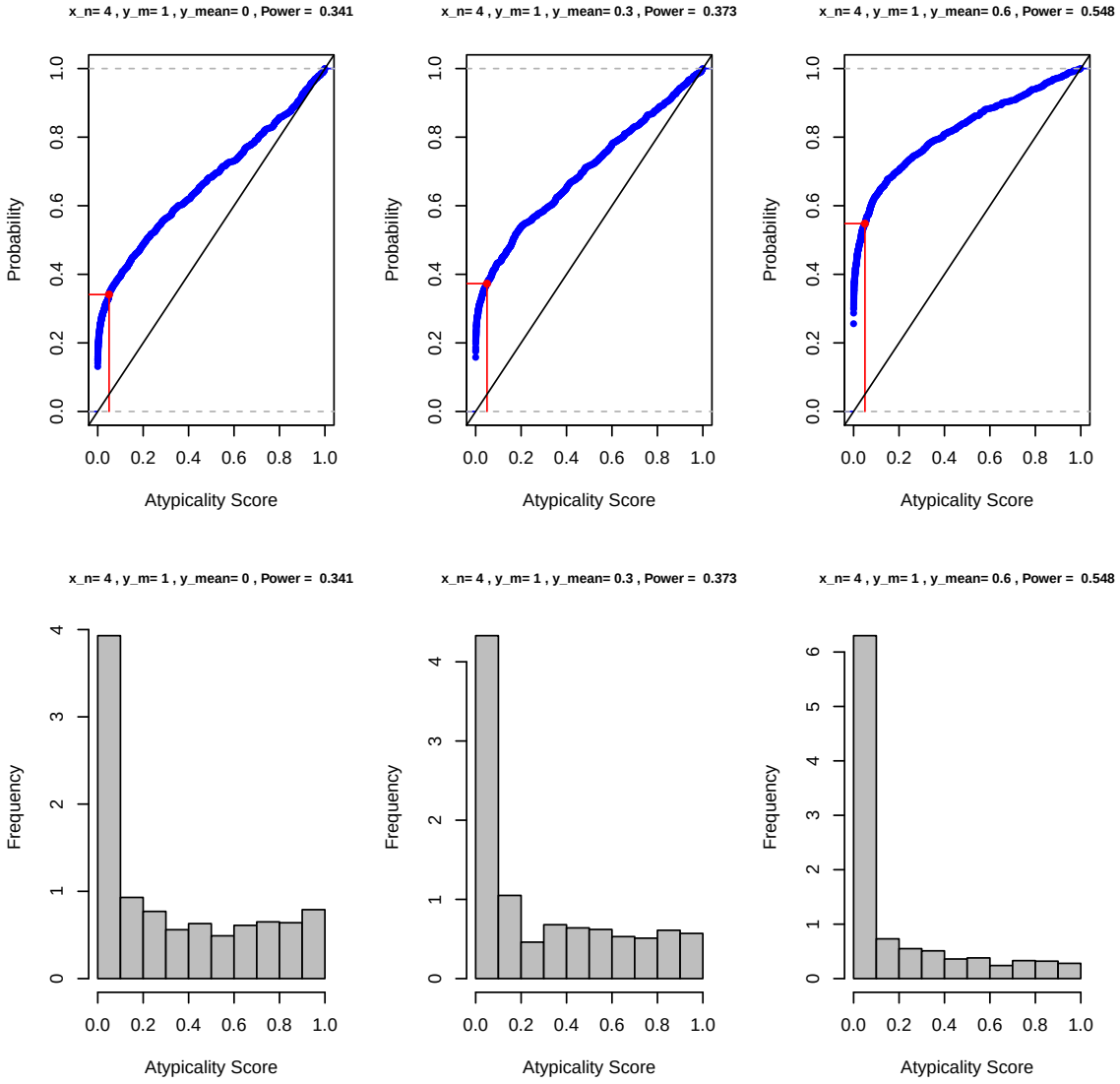


Figure 5: The empirical CDFs and histograms of the Atypicality Scores with four known objects, one introduced object, and changing means.

First, we begin by showing Figure 5 for the smallest sample sizes that we can have, with four known objects and one introduced object. The first plot in Figure 5 shows that the distribution of the atypicality scores when the known objects and the introduced object have the same distribution is not uniform. Due to the skewness of the multivariate score vector, this can be expected. If the scores did not have this skewness, then we would expect a completely uniform distribution of p-values. We will see that in further simulations, as the number of known objects increases, and

the smaller the number of introduced objects in proportion, then the more uniform our simulated distribution of atypicality scores will be. The second and third plots show that as the mean of the introduced object moves further away from the mean of the known objects, we get a higher frequency of low atypicality scores, as expected.

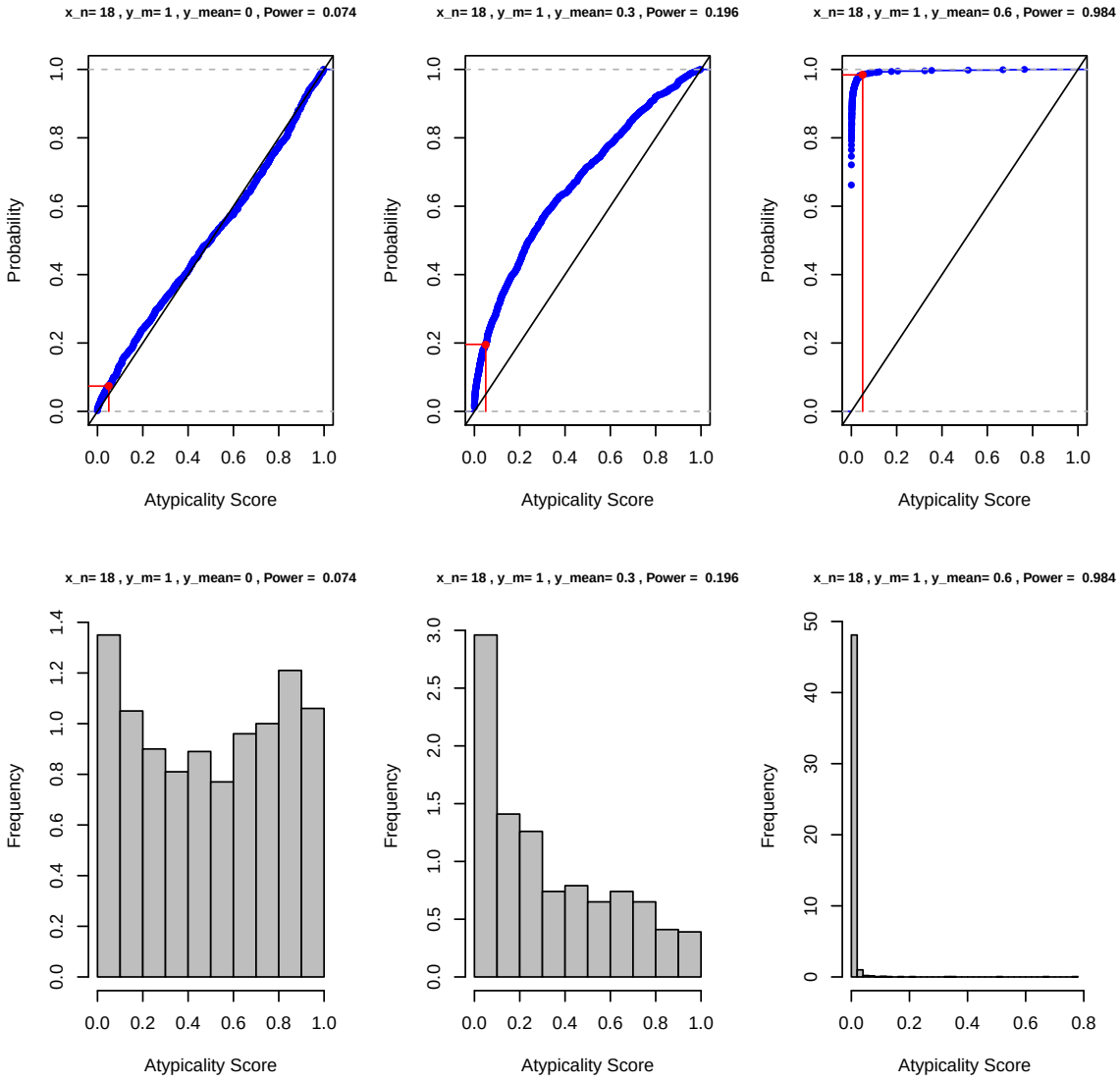


Figure 6: The empirical CDFs and histograms of the Atypicality Scores with eighteen known objects, one introduced object, and changing means.

Figure 6 is similar to Figure 5, but has eighteen known objects. We can see that the top left plot is near uniform. This is because H_0 is true and we have a large

number of known objects. The middle and right plot show the expected results of getting more atypicality values as the unknown object moves away from known objects.

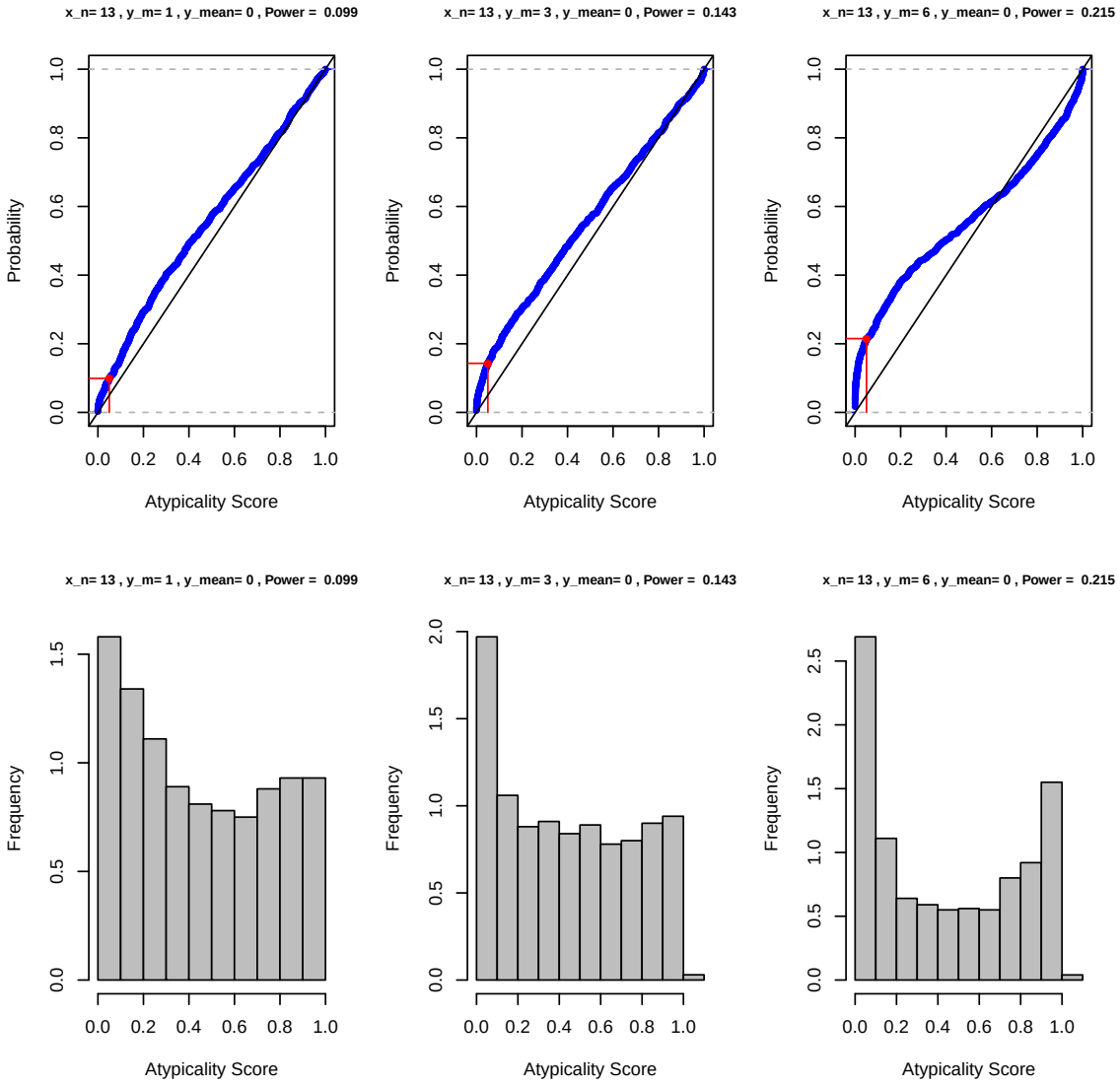
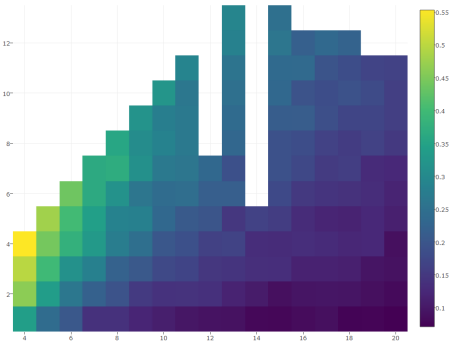


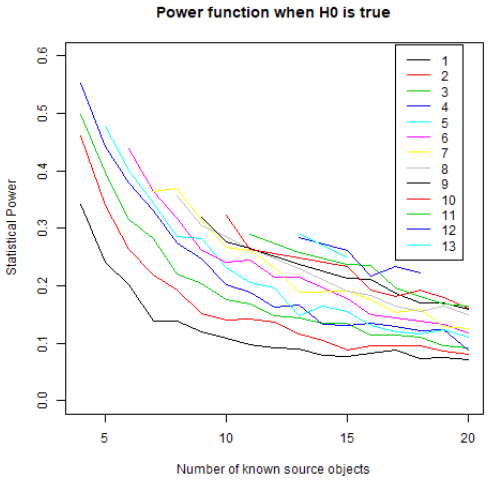
Figure 7: The empirical CDFs and histograms of the Atypicality Scores with thirteen known objects, H_0 is true, with an increasing number of unknown objects.

Figure 7 illustrates how the distribution of the atypicality scores change as increase the number of introduced objects. We see that as the number of introduced objects approaches the number of known objects, we get low atypicality scores more frequently.

To help summarize the results in their entirety, let us first look at the results when the null hypothesis is true. That is, when the known and unknown objects do in-fact come from the same distribution. See the heat map on Figure 8 where the x-axis represents the number of known objects, the y-axis represents the number of unknown objects, and the coloring represents the type I error for that particular simulation. We can see that when the ratio of unknown objects to known is small, then the probability of committing a type I error is closer to 5%. The type I error probability tends to rise as the ratio of the unknown and known objects gets closer to one.



(a) X-axis represents the number of known objects. Y-axis represents the number of unknown objects. The coloring represents the probability of committing a type I error.



(b) Line graph when the null hypothesis is true.

Figure 8: Visual representations of the probability of committing a type I error when using atypicality and the null hypothesis is true.

An exhaustive listing of these results and their plots is readily available by request as supplementary material. Also, the plots of the observed power for different values of known and unknown objects can be found in the appendix.

3. SUPPORT VECTOR MACHINE COMPARISON

Now that we have thoroughly described our atypicality metric, we should compare it to other known methods for sample categorization. One of the most popular categorizing tools used today is the Support Vector Machine.

In this chapter, we will describe what a Support Vector Machine (SVM) is and how it works. After that, its popularity will be discussed, as well as why it was chosen as a good comparison to our atypicality measure. We will then run a SVM on the same data sets that were generated for our atypicality simulations, with the results following. Finally, there will be a discussion comparing the results of the two methods.

3.1 Introduction to Support Vector Machines. There have been many different methods developed to solve the classification problem. Given a set of objects grouped into classes based on their known features, the classification problem involves determining the optimum class membership of a new object. These classes can be predetermined by the user, or developed using a training set of data. The training data contains objects with their corresponding features. In order to train the SVM, some combination of one or more of the object features is used to determine which class the object belongs to. So in this case, the classes of known objects are known before hand, and are used to help determine the class of new objects.

The classification problem itself is claimed to be in the realm of several different disciplines. Computer Science claims it as a part of machine learning. Information system analysts claim classification as a subproblem of data mining. Also, the field of statistics lays claim to the problem. The atypicality measure that we have developed fits into this last category, stating that classification can be done through a probability measure. The fact is, all three have valid claims. Recently, the overlap between these three disciplines has become more clear as researchers from

these areas share their ideas via publications, presentations, and lectures.

In this chapter, we will focus on what is currently one of the more popular methods of solving the classification problem, support vector machines. As such, we will discuss it's origins and derivation in detail.

To help set the idea of what a SVM is, here is a brief description with more rigorous detail to follow. A support vector machine can be interpreted as a surface that creates a boundary between different multidimensional points of data. In other words, this boundary works as a way to separate the different data points into classes in multidimensional space.

The boundary itself is generally a flat hyperplane that attempts to partition the data on either side of the boundary. A hyperplane is a subspace that is one dimension less than the ambient space around it. For example, if we're looking at a two-dimensional space, then the hyperplanes are one-dimensional lines. If we're looking at three-dimensional space, then the hyperplanes are two-dimensional lines. In the case of four-dimensional space, the hyperplane is a three-dimensional enclosure. The concept continues on into higher dimensions. "Similar" data points are on the same side of the boundary, and "differing" points are separated via the hyperplane. The SVMs learning method combines both instance-based nearest neighbor learning and linear regression modeling to develop the hyperplane. This combination of learning is very powerful and allows SVMs to model complex relationships fairly well.

Typically, SVMs are applied in a binary classification system. However, it is possible to use them with multi-class problems [76, 77, 16, 25, 17]. This is by implementing a voting method and combining many binary classification systems. Because the atypicality measure presented in this paper determines the probability that a group of objects is from a similar source as another group of objects on a case by case basis, we will focus on the binary classification methods with SVMs.

The reason SVMs were chosen as a comparison to our atypicality measure is mainly due to its recent rise in popularity. This rise in popularity is likely because of its solid performance in classification. Although the math needed to implement a SVM is quite complex, it has been around for several decades. Because of its stellar performance, many SVM's have been implemented in popular libraries across many different programming languages (including R). This allows them to be adopted by audiences who may not have considered using them before, due to a lack of knowledge on the subject or understanding how to implement them. Without extensive study, comprehending the math needed to implement an SVM can be a daunting task for those not fluent in their behavior. By implementing SVMs in popular and well-maintained libraries, the door to new and diverse applications has been opened for researchers not directly studying the material.

SVMs can be used in nearly any learning problem, including classification and numeric prediction problems, through pattern recognition. Notable applications include: classification of microarray gene expression data
[4, 9, 8, 18, 23, 27, 38, 46, 49, 53, 55, 58, 68, 69], which is useful to identify cancer and other genetic diseases; text categorization
[32, 74, 33, 36, 65, 39, 60, 34, 64, 70, 28, 78], which is useful for language identification on documents or web pages; or the classification of documents by subject matter. SVMs are also good at detecting important, but rare, events. Examples of this include combustion engine failure, security breaches [50, 26, 71], or earthquakes.

3.2 SVMs and Hyperplanes. As you've read, SVMs use hyperplanes to separate data into different classes. There are several different ways to implement these hyperplanes, each with different difficulties and perks. First, we will discuss the simplest case, straight line/surface hyperplanes, and then we'll discuss kernels used to create more ornate hyperplanes.

In the ideal case, we would like our hyperplane to be a straight line/surface that separates the data completely into two different classes. When this occurs, we have linearly separable data.

In both of these examples, the SVM algorithm generates the hyperplane that completely separates the data into two sections using a straight line in the 2D case, and a straight plane in the 3D case. The concept, while not possible to truly visualize, is carried on in higher dimensions.

You can imagine that there are other lines that could separate the circles from the squares. The line/plane that is generated by the SVM algorithm looks for a particular one that is called the maximum margin hyperplane (MMH). This hyperplane has the greatest separation between the two classes. This allows the greatest chance of new data falling on the "correct" side of the hyperplane when classified by the SVM model; even when random noise is present.

In order to generate the MMH, the SVM algorithm identifies the support vectors for each class. The support vectors are the data points that are the closest to the MMH. Each class will have at least one support vector, meaning that there may be more than one. When this is the case, the two support vectors from the same class are the same distance from the MMH. Using just these support vectors alone, one could construct the MMH. This is one of the reasons for SVMs popularity; the support vectors provide a very compact way to store the classification model regardless of the number of features in the data itself. Using vector geometry, the SVM algorithm will find the parallel planes that create the furthest distance from the MMH. When introducing new data, whichever side the point falls on will determine its classification. When points fall on the MMH, generally one of the classes is chosen based on a hard coded decision.

A detailed explanation of constructing the MMH can be found in Cortes's and Vapnik's paper on the subject [11]. A more generic description will be given here via

assistance from Lantz [37]. To begin discussing how the MMH is generated algorithmically, we will start by analyzing the simple case when the data is linearly separable (which may not always be the case, as we will see).

The process to find the MMH is one that searches through the space of all possible hyperplanes to find the two parallel planes that separate the data into their defined classes, and are then as far apart as possible.

First thing to get started is to define a hyperplane in n-dimensional space.

$$\vec{w} \cdot \vec{x} + b = 0 \tag{11}$$

The arrows above the letters indicate that we're dealing with vectors. The $\vec{w}$ vector of weights, $\vec{x}$ is the vector of data points, and b is a scalar value known as the bias. One could think of the bias term in a similar fashion as one would think of the intercept term in the slope-intercept form of a line in simple linear regression.

Let's say that we have data for two classes, where $x_1 \dots x_n$ are the data points and each data point has a corresponding class denoted by 1 or -1, given as $y_1 \dots y_n \in \{1, -1\}$. A particular data point and it's class could then be given as $(x_i, y_i), i = 1 \dots n$. Then we can say that data is linearly separable if there exists a vector $\vec{w}$ and scalar b such that

$$\begin{aligned} \vec{w} \cdot x_i + b &\geq 1 \qquad \text{if } y_i = 1 \\ \vec{w} \cdot x_i + b &\leq -1 \quad \text{if } y_i = -1 \end{aligned} \tag{12}$$

We can rewrite the above inequalities in the form

$$y_i \left(\vec{w} \cdot \vec{x_i} + b\right) \geq 1, i = 1, \dots, n. \tag{13}$$

As noted before, the MMH is the optimal hyperplane that linearly separates the data with a maximal margin. To find this optimal hyperplane, the direction

$w/\left|w\right|$ that will give this maximal margin is determined. The margin itself is a function of $\vec{w}$ and b given as $d\left(\vec{w},b\right)$.

$$d\left(\vec{w},b\right)=\min_{\{x:\ y=1\}}\frac{\vec{x}\cdot\vec{w}}{\left\|\vec{w}\right\|}-\max_{\{x:\ y=-1\}}\frac{\vec{x}\cdot\vec{w}}{\left\|\vec{w}\right\|}. \tag{14}$$

The goal is to find the unique arguments $\vec{w_0}$ and b_0 that maximize the above distance, yielding the hyperplane $(\vec{w_0},b_0)$. It follows from Equation (13) on page 53 and Equation (14) on page 54 that

$$d\left(\vec{w_0},b_0\right)=\frac{2}{\left\|\vec{w_0}\right\|}=\frac{2}{\sqrt{\vec{w_0}\cdot\vec{w_0}}}. \tag{15}$$

This indicates that the MMH is a unique one that will minimize $\vec{w}\cdot\vec{w}$ under the constraints of Equation (13) on page 53. This means that solving for the MMH is a quadratic optimization problem. The data points x_i for which $y_i\left(\vec{w}\cdot\vec{x_i}+b\right)=1$ will be known as the support vectors. Since we want to maximize the distance, we can look at the equation in a different way. From the distance equation, we know that in order to maximize the distance, we need to minimize $\left\|\vec{w}\right\|$ because it is in the denominator. This is expressed then as

$$\min\left(\frac{1}{2}\left\|\vec{w}\right\|^2\right)\ \text{s.t.}\ y_i\left(\vec{w}\cdot\vec{x_i}-b\right)\geq 1,\forall\vec{x_i}. \tag{16}$$

Finding the solution to Equation (16) on page 54 given the constraints of can be a processor-intensive task. As of late, quadratic optimization software with specialized algorithms (outside the scope of this paper) have been developed to ease this burden, even with large datasets.

3.3 Nonlinearly Separable Data. So far we've been working under the assumption that the data itself can be linearly separated. While this is ideal, it is often not the case. There are two approaches to alleviate this problem. The first will employ the

use of slack variables to create a soft margin. This soft margin allows for some points to fall on the incorrect side of the MMH. The second option is to map the problem into a higher dimensional space. Doing so can turn a nonlinear relationship into one that is in the new dimensional space. This is known as the kernel trick.

First, we'll discuss using slack variables to create a soft margin. Imagine, if you will, that we have data where there is a point for each class on the wrong side of the MMH. These points violate the constraints set forth by equation Equation (13) on page 53. To account for this, a cost is associated with these vectors. The cost is in the form of a distance from the vector to the margin separating the data, given as $\xi_i \geq 0, i = 1, \ldots, n$. The goal now turns from finding the maximum margin to linearly separate the classes, to an algorithm that minimizes the total cost of the system, given as

$$\Phi(\xi) = \sum_{i=1}^{n} \xi_i^{\alpha} \text{ where } \alpha > 0. \tag{17}$$

with new constraints

$$y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 - \xi, i = 1, \ldots, n. \tag{18}$$

This changes our original quadratic problem so that we need to minimize

$$\begin{aligned} &\frac{1}{2}\|\vec{w}\|^2 + C\Phi(\xi) \\ &\text{s.t. } y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1 - \xi_i, \forall \vec{x_i}, \xi_i \geq 0, \alpha > 0, \end{aligned} \tag{19}$$

where C is a constant and $\Phi(\xi)$ is a monotonic convex function.

Given a constant C that is sufficiently large, an α that is sufficiently small; the vector $\vec{w_0}$ and constant b_0 that minimize the function Equation (19) on page 55 determines the hyperplane for the system. With this hyperplane, the number of errors in the training set will be minimal, and the rest of the elements that are on the correct side of the margin will be separated by the maximal margin. The process

of minimizing this function can be found in the appendix of Cortes and Vapnik [11].

Let's take a look at a toy example of a SVM and follow the process of categorizing data. The data used was taken from the University of California, Irvine (UCI) Machine Learning Repository [40]. Here is a direct quote regarding the data itself:

> The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We typically train on the first 16000 items and then use the resulting model to predict the letter category for the remaining 4000.

Because there are 16 feature vectors besides the class variable, it can be difficult to visualize the data. If we decide to use a 2D graph to display our SVM results, we need to find the two features that best show the separation between the classes, and then draw the line representing the hyperplane edge in that space. This is not a simple task. To help alleviate this problem, Linear Discriminant Analysis (LDA) is performed on the data set. LDA is a supervised method that computes the directions (a.k.a the linear discriminants) that represent the axes that maximize the separation between multiple classes [30].

Figure 9 on page 57 represents the data on the two linear discriminants that most separate the classes. We can see that horizontal axis explains roughly 31.6% of the between-group variance. We can also see that the character 'Z' class clusters towards the top left corner of our plot, and the character 'M' class clusters towards the bottom right. These will make exceptional candidates for a linearly separable

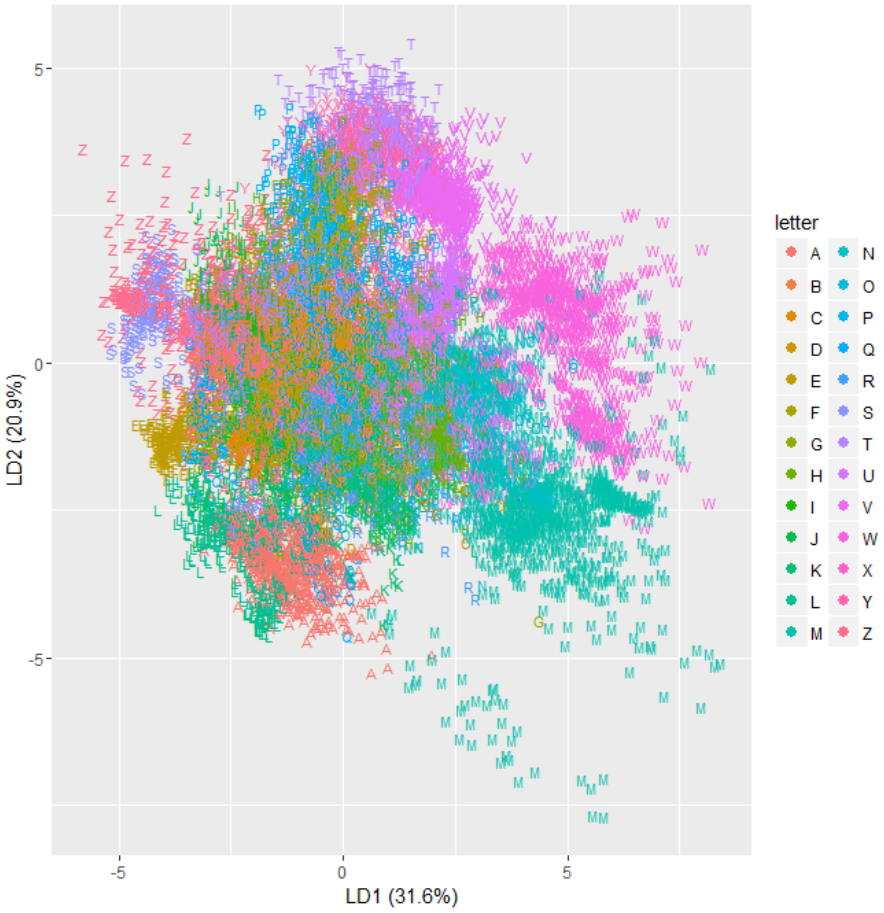


Figure 9: Letter data with LDA axes.

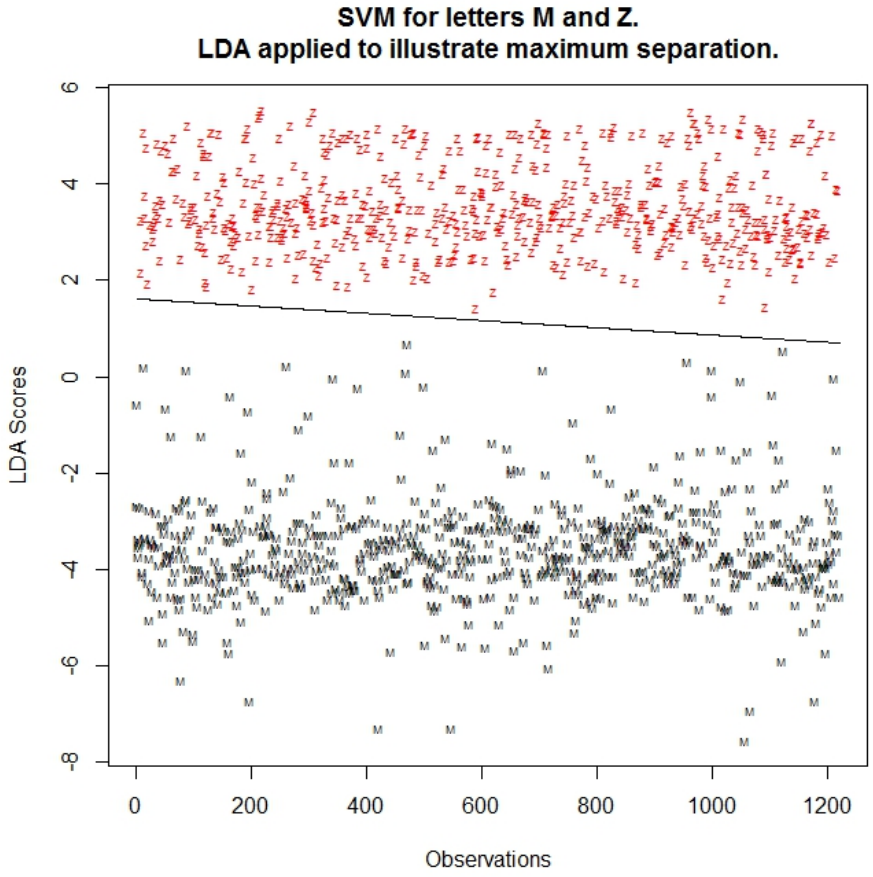


Figure 10: Letters 'M' and 'Z' with LDA axes.

binary classification example.

Figure 10 on page 58 shows the letters 'M' and 'Z' separated by the line generated by applying the SVM algorithm to the data. We can see that the line linearly separates the letters. When using 75% of the data for training the SVM algorithm, the SVM model correctly predicted 100% of the testing data. This shouldn't be too much of a surprise for us since we saw that the data was linearly separable.

Now, what happens if the data is *not* linearly separable. Looking back at Figure 9, we can choose two letters that are similar looking, and therefore have overlapping clusters. Two such letters could be 'D' and 'O'.

Figure 11 on page 59 shows that even with the LDA axes, we do not have

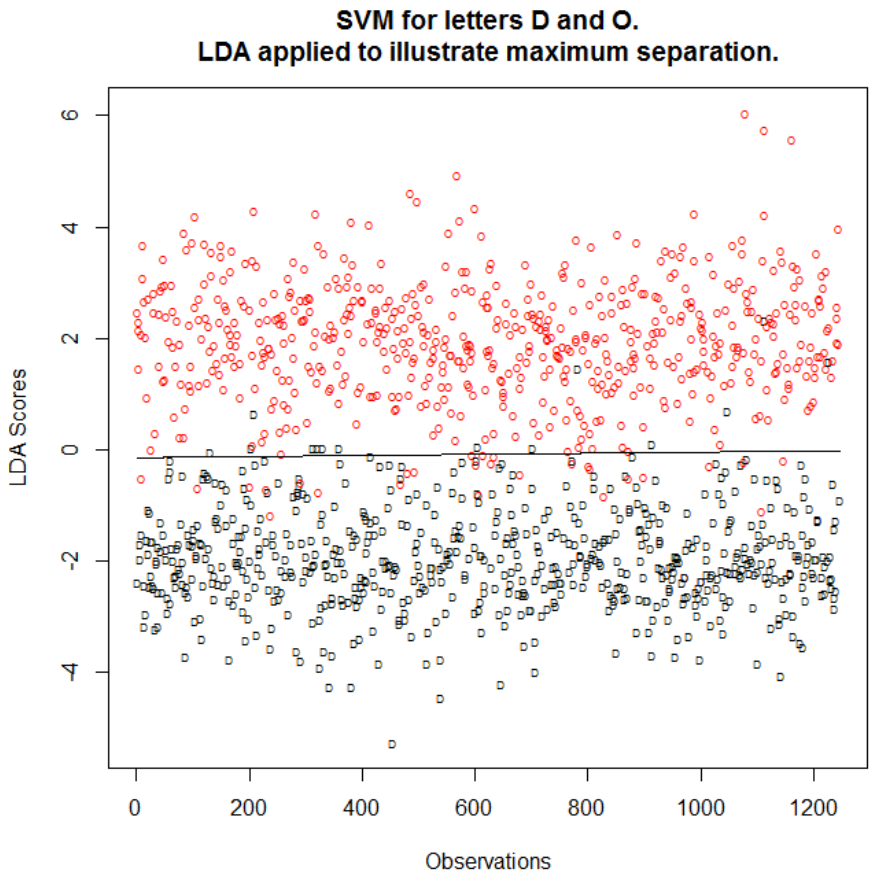


Figure 11: Letters 'D' and 'O' with LDA axes.

completely linearly separable data points. With close observation, we can see there are points on either side of the MMH, creating cost in the SVM model itself. When training the SVM model with 75% of the data, the SVM model correctly predicted roughly 97% of the testing data values. Quite robust indeed.

3.4 Support Vector Machines in Non-Linear Space. it is not always the case that we will end up with data that shares a linear relationship. We just saw when this can occur, and noted how a slack variable with a cost function can give us a means to separate the data as much as possible while still having some incorrectly classified data. Another method to handle data that is not directly separable is to use what is know as the kernel trick. The kernel trick is the process of mapping the data to a higher dimensional space. In this process, we hope to find a relationship that allows linear separation.

A kernel is simply a mathematical mapping (or function) that will transform an n-dimensional input vector $\vec{x}$ into an N-dimensional feature vector. For our purposes, $n < N$. The kernel function itself, denoted as ϕ, is

$$\phi : \Re^n \to \Re^N. \tag{20}$$

Let's now take a look at an example when we do decidedly do not have linearly separated data. Figure 12 shows what data can look like when it is clearly not linearly separable. The two circles represent two classes of data. However, by mapping it to a higher dimension, as seen in Figure 13 we find that it is possible to fit a hyperplane between the data points on this new dimension.

The extra dimensions are not inherently part of the original data set, but are constructed via a mathematical relationship computed by the kernel. This relationship is dependent on the kernel that is chosen, as there are several to choose from. But before we look at the specific kernels themselves, let's look at the general kernel function. This generalized function applies a transformation to the feature

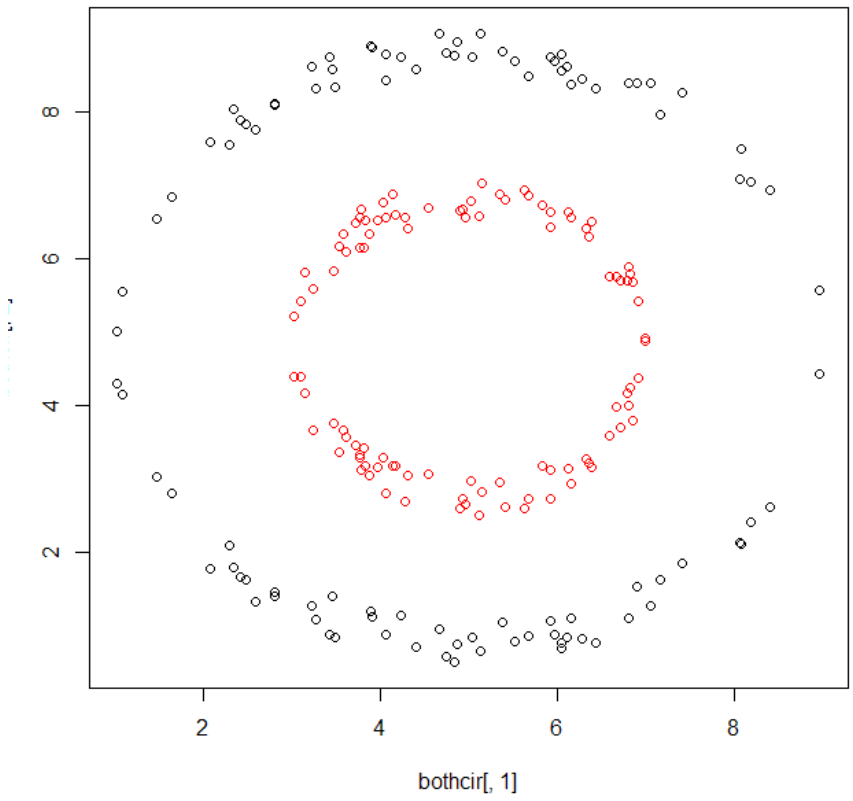


Figure 12: Non-linearly separable data in 2D

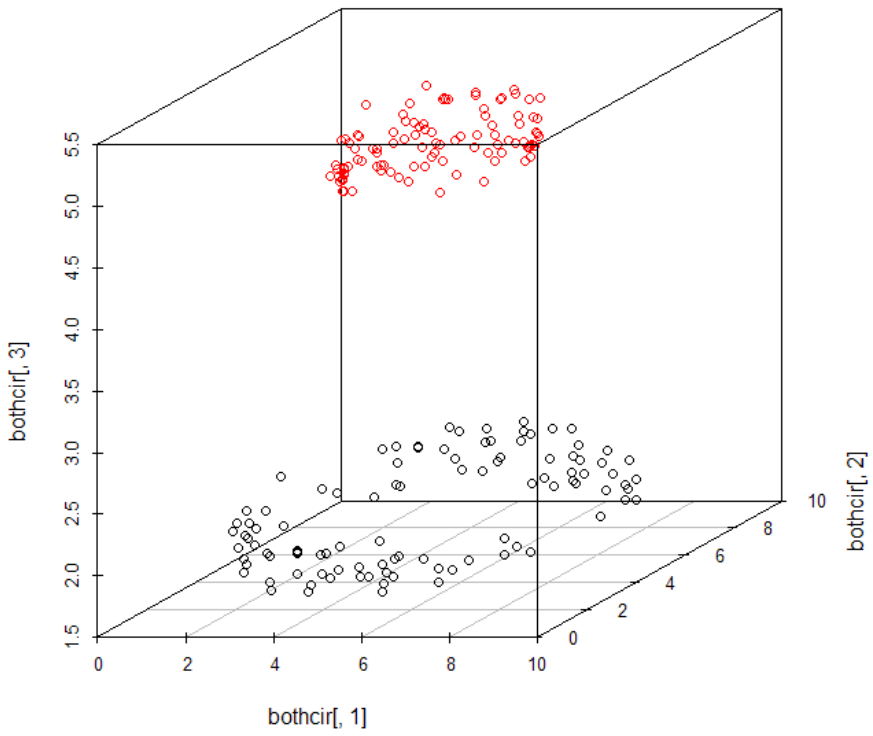


Figure 13: Non-linearly separable data in 3D

vectors $\vec{x_i} \in \Re^n$ to map them into an N-dimensional feature vectors.

$$\phi\left(\vec{x_i}\right) = \phi_1\left(\vec{x_i}\right), \phi_2\left(\vec{x_i}\right), \ldots, \phi_N\left(\vec{x_i}\right), i = 1, \ldots, \ell. \tag{21}$$

To clarify, ℓ is the number of feature vectors that make up the input data, n is the number of dimensions per vector, and N is the newly increased dimensionality of each vector created by a transformation defined by ϕ.

The generalized kernel, denoted by $K\left(\vec{x_i}, \vec{x_j}\right)$, then combines the transformed feature vectors using the pairwise dot product; taking two numbers and returning a single number. This is shown as

$$K\left(\vec{x_i}, \vec{x_j}\right) = \left\langle \phi\left(\vec{x_i}\right), \phi\left(\vec{x_j}\right) \right\rangle_N, \tag{22}$$

where $\langle \cdot, \cdot \rangle_N$ is an inner product of $\Re^N$.

There has been a lot presented in the last page or two, so let's streamline it all into a concise pipeline: For $\vec{x_i}, \vec{x_j} \in \Re^n, K\left(\vec{x_i}, \vec{x_j}\right) = \left\langle \phi\left(\vec{x_i}\right), \phi\left(\vec{x_j}\right) \right\rangle_N, n < N$, and $\phi\left(\vec{x}\right)$ transforms $\vec{x}$ to $\Re^N$ $\left(\phi : \Re^n \to \Re^N\right)$.

With the general form of a SVM kernel given, we should take the time to consider the consequences of its use (and also how to get around them). The biggest caveat to consider is the increase in computational complexity by increasing the dimensionality from n to N. If N grows exponentially (e.g. $O\left(2^n\right)$) with respect to n, then the problem may become infeasible to calculate due to storage space and computational constraints.

This problem is actually avoided due to the fact that the SVM does not explicitly work in the N-dimensional space during the training or testing phase. The training data is only used to compute the dot products mentioned before. Thus the kernel implicitly transforms the given data to a higher dimension during the training phase to train the SVM model. One does not necessarily need to store the

transformed vectors; with the exception of visuals and plotting. There is the computational complexity of calculating K, but so long as the kernels themselves are computationally efficient, then the cost of computation can be minimal.

There are a handful of kernels available. We will discuss three that may be considered the most popular here.

The polynomial classifier [11] of degree d in n-dimensional input space takes the form of

$$K\left(\vec{x_i}, \vec{x_j}\right) = \left(\langle \vec{x_i}, \vec{x_j} \rangle + 1\right)^d, \tag{23}$$

although most packages add more parameters

$$K\left(\vec{x_i}, \vec{x_j}\right) = \left(\gamma \cdot \langle \vec{x_i}, \vec{x_j} \rangle + r\right)^d, \tag{24}$$

where r and γ are set by the user.

Looking at a basic example where the feature vectors originate from $\Re^2$ space, and we set $d = 2$ (yielding a two-degree polynomial kernel) then the implicit transformation will take on the form of

$$[x_1, x_2] = \left[x_1^2, x_2^2, x_1 x_2 \sqrt{2}, x_1 \sqrt{2c}, x_2 \sqrt{2c}, c\right] \tag{25}$$

This essentially mapped $\Re^2 \rightarrow \Re^6$, adding four more dimensions. In general, a d-dimensional kernel will map a dataset in $\Re^n$ space to $\binom{n+d}{d}$-dimensional space. If we were to actually store these transformed vectors instead of implicitly using the calculations from the kernel, these problems could quickly become intractable.

The second kernel we will look at is the Sigmoid kernel [24]. It looks like

$$K\left(\vec{x_i}, \vec{x_j}\right) = \tanh\left(\gamma \left\langle \vec{x_i}, \vec{x_j} \right\rangle + r\right). \tag{26}$$

where, again, r and γ are set by the user. The use of a sigmoid activation function is similar to procedures found in certain neural networks. The sigmoid function (which is a special case of the logistic function) allows the hyperplane to curve at different points to accommodate points that would normally be on the wrong side of a linear hyperplane.

The last kernel we'll look at is the Radial Basis Function (RBF) Kernel. It looks like

$$K\left(\vec{x_i}, \vec{x_j}\right) = \exp\left(-\gamma \cdot \|\vec{x_i} - \vec{x_j}\|^2\right), \text{ where } \gamma > 0. \tag{27}$$

The RBF kernel is a natural choice to start with when selecting one. It behaves similar to RBFs in neural networks as activation functions whose value only depends on a distance.

The biggest problem facing SVMs attempting to model non-linear data is that there are no hard and fast rules as to which kernel to pick. Even after one is selected, choosing the parameters takes trial and error that will yield differing results. While a detailed study of kernel and parameter choice is outside the scope of this paper, an excellent overview can be found in [24].

Before we move on to the next section, let's look back at Figure 12. After training models for straight linear, RBF, polynomial, and sigmoid kernels, the test results had the following accuracy: linear - 56%, polynomial - 56%, sigmoid - 44%; which are all fairly a 50-50 chance, but RDF stood out with 100% accuracy. This makes sense as that RBF transforms the new dimension in such a way that would look similar to Figure 13, creating a separating hyperplane perpendicular to the z axis so to speak. The other kernels try to weave the hyperplane around the feature points, which would work poorly given the circular nature of the data.

One could tinker with the parameters of the different kernels, but the results change little in this example. The code to generate the data and plots given in this

subsection can be found in the appendix.

3.5 One-Class Support Vector Machines. The final topic to cover before we compare SVMs to our atypicality classification method are one-class SVMs. Previously we've seen how hyperplane boundaries are constructed between multiple classes. The reason that we can't use those methods directly in our comparison is because that makes an initial assumption that we have two classes to begin with. Atypicality is currently more focused on the probability that a set of objects belongs to a given distribution. Instead of multi-class classification, we can use one-class support vector machines. These have also been called distribution estimators by computer scientists [63, 72, 62].

Scholkopf et al [62] present a planar method of developing a hypersurface that attempts to encompass the training data to form a boundary. The hypersurface has a maximal distance from the origin in the feature space, separating all the data points from the origin. Any data outside of this boundary would be considered novelty (outlier) data. Tax and Duin [72] have dubbed their method the support vector domain description (SVDD). This is a spherical approach, as opposed to planar. The algorithm obtains a spherical boundary, whose volume is minimized to better help disclose outliers. Since the majority of software packages out there use Scholkopf's method, we will focus on that.

The primary difference in calculating the hypersurfaces comes from the quadratic minimization function found in equation Equation (19) on page 55. In that previous equation, C was used as a smoothness parameter. The updated version here can be shown as

$$\begin{aligned} &\min \left(\frac{1}{2} \left\| \vec{w} \right\|^2 \right) + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i^{\alpha} \\ &\text{s.t. } y_i \left(\vec{w} \cdot \vec{x_i} - b \right) \geq 1 - \xi_i, \forall \vec{x_i}, \xi_i \geq 0, \alpha > 0, \end{aligned} \tag{28}$$

The emphasis here is put on the parameter ν. This characterizes the solution

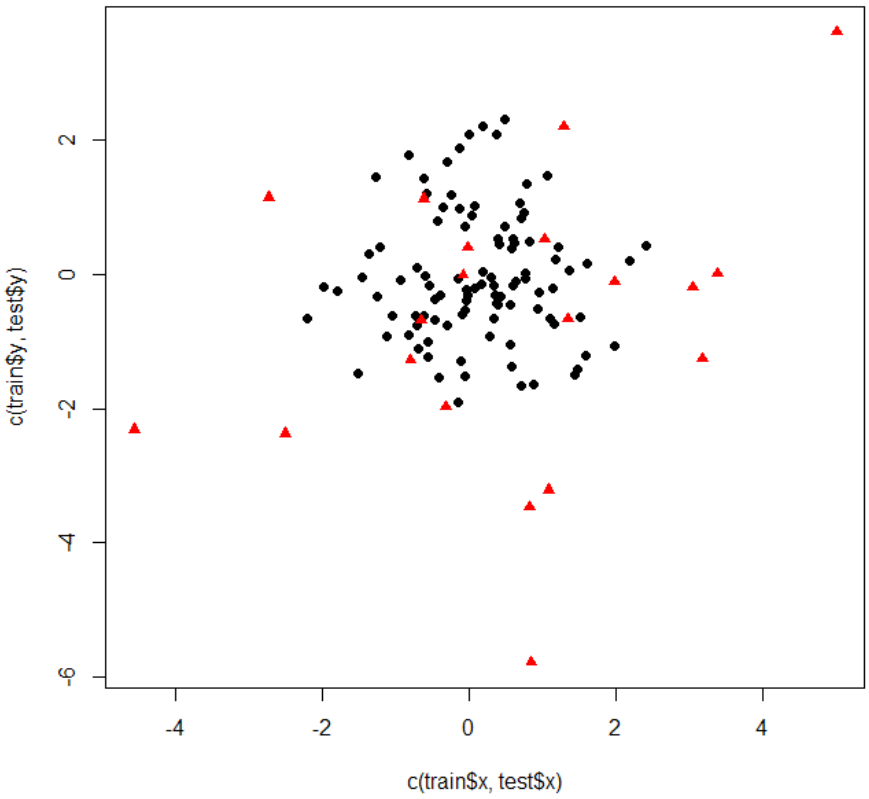


Figure 14: One-Class SVM Toy Example

by setting an upper bound on the fraction of outliers that are allowed when constructing the hypersurface with the training data. This also sets a lower bound on the number of training examples that can be used when training the SVM model. This method of creating the SVM is arguably the most popular and is referred to as a ν-SVM.

Figure 14 illustrates a toy example of a one-class SVM. Here, the black dots represent (x,y) coordinates where $x \sim N(0, 1)$, and the red triangles were generated using $x \sim N\left(0, \sqrt{2}\right)$. When training the model, the ν value was set to 0.05. This means that when constructing the concave hypersurfaces, 5% of the data were considered to be outliers, specifically, the ones furthest away from the origin. The RBF kernel was used, since the data is not linearly separable. One hundred points

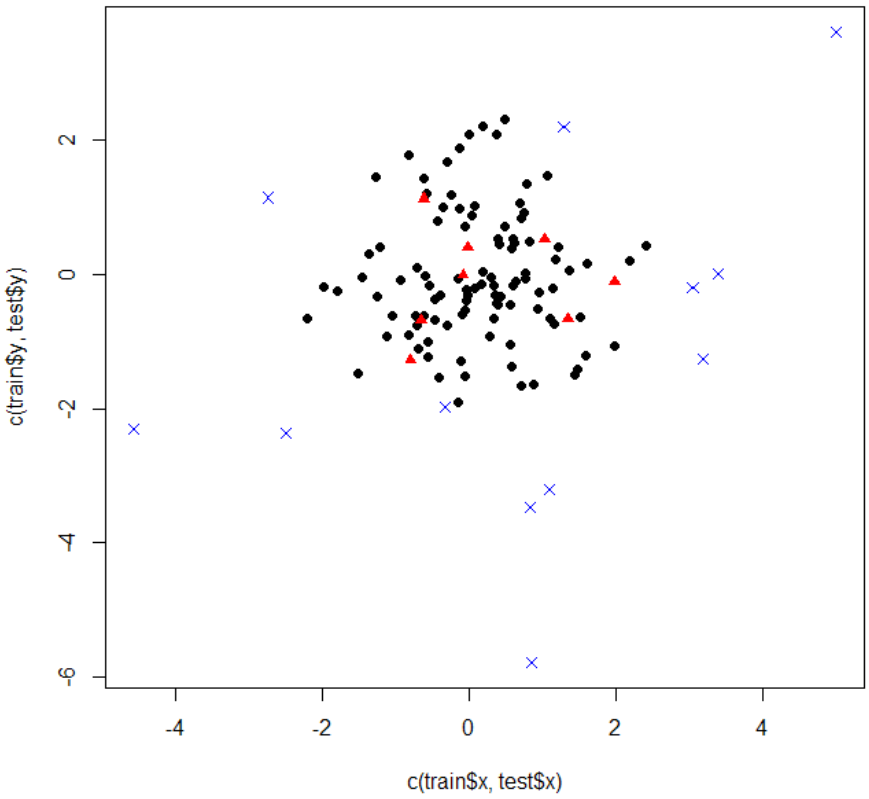


Figure 15: One-Class SVM Toy Example Continued

were used when training the SVM model. Twenty new objects were introduced with the new distribution. The model stated 60% of the new objects did not belong to the original distribution of objects.

The objects classified as not part of the distribution are shown in Figure 15 by the blue 'X' symbols. It cannot be stressed enough how greatly the choice of ν can effect the model and its classification accuracy. If desired, we could use any of the kernels listed in the last section. Given the toy example, it wouldn't make much sense to use anything besides the RBF method, but it's important to know the availability of the others.

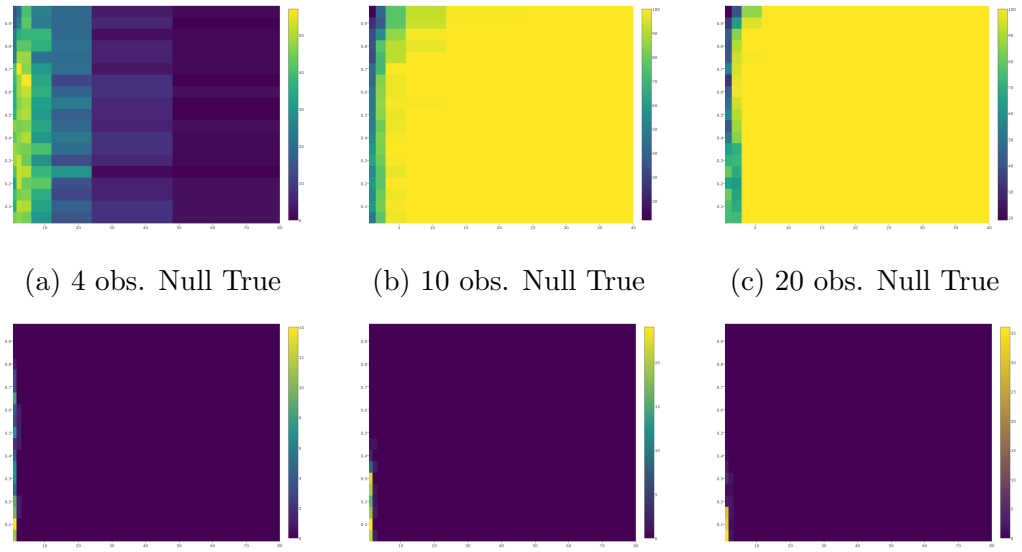
Now, we can use a one-class support vector machine to properly compare our atypicality classification method with SVMs.

3.6 One-class SVM compared to Atypicality Classification. Here, we will compare the results of a one-class SVM model to that of an atypicality model for classification. We will do this with two different experiments. The first will compare the two models when the test object is generated from the same distribution as the training data. The second will gradually move the testing object away from the mean of a known distribution of training objects.

Before executing these experiments, we must determine what to set the ν parameter as for the one-class SVM. As a reminder, the ν parameter is used to determine the percentage of objects to allow as outliers when determining the hyper-planar surface that encapsulates the data. Outliers are considered in the hyper-planar calculation process. Since ν is a percentage, then valid numbers range from $[0, 1]$. This extra experiment will vary the ν in steps of 0.05 as we also have different values for the number of training objects and the number of dimensions the objects exist in. The table 7 gives the complete results of this experiment.

Let's take some different view of these results and summarize. The table with the complete listing of results can be found in Appendix section C.4.

We can summarize this data by view the Figure Figure 16 on page 70. The Figure shows six different heat maps under different conditions. They represent the results of the SVM simulations when we adjust the ν parameter and the number of dimensions the objects have. The top three represent the case when the null hypothesis is true and the test object is sampled from the same distribution as the training data. Darker areas represent low categorization percentage, and light colors are higher number. The brighter the color, the better the SVM model performed given the conditions. The bottom three are heat maps when the training object was held at two standard deviations away from the mean. One would prefer darker colors in this instance. Looking at these heat haps, it was determined that setting the ν parameter 0.1 had favorable results in the majority of cases. Therefore, that is



(a) 4 obs. Null True (b) 10 obs. Null True (c) 20 obs. Null True

(d) 4 obs. Null False (e) 10 obs. Null False (f) 20 obs. Null False

Figure 16: Heatmaps displaying results of a varying ν parameter.

the parameter used when training the SVM models that are compared to the atypicality results.

In order to evaluate the effectiveness of atypicality classification against SVM classification, the following outline was followed.

- All objects are taken from the standard normal distribution. These simulations will illustrate each model's ability to correctly classify the test case as part of this distribution.
- Since SVMs can't consider a group of objects at one time, then only compare one test element at a time.
- ν was set to 0.1 for all simulations using the RBF kernel. The reason we chose this parameter is described below, but in summary, it had the best performance for the SVM model.
- Consider situations with varying training sample size. In this paper, we consider 4, 10, and 20 samples to train the SVM model and estimate the

parameters for the atypicality model.

- Consider differing dimensionality of the original data. For this paper, dimensionality was exponentially increased, so the set of test cases includes dimensions 1, 2, 4, 8, 16, 32, 64.

- For each combination of parameters set above, run 100 simulations where all objects (including the test case) comes from the same distribution. Then compare the percentages of the correct classifications for each model.

For the SVM, the decisions are 1 for TRUE, 0 for false. For each simulation, we sum the TRUE results and divide by 100. For the atypicality classification, we use the hypothesis that test case is a part of the distribution, so p-value results less than 0.05 would lead us to reject this hypothesis. So looking at the results, we tally all the p-values greater than 0.05 and divide by 100.

The code and complete results for these simulations can be found in the appendix. A summary of the results are shown below.

Here, we see in Table 6 that the performance of the SVM starts to break down as the dimensionality increases. This is an expected behavior, according to Rasmussen [59]. When we have a data set in N dimensions, if $N > n$, where n is the number of training objects; then there will always be a separating hypersurface. The issue is that the hypersurface may not yield good results. A kernel can be used to help with the issue, but as we saw above, the SVM using the RBF kernel started to deteriorate as the dimensionality increased. The atypicality classification system, however, maintained a higher performance, especially as the number of training objects increased. This is to be expected, as more objects allow us to get more accurate estimates for the distribution's parameters.

Now, what about detecting objects that are not part of the same distribution as the training set? For the next set of simulations, the training objects were kept at

# of training objects	# of dimensions	SVM % Correct	Atypicality % Correct
4	1	24%	63%
4	2	20%	59%
4	4	9%	61%
4	8	0%	72%
4	16	0%	68%
4	32	0%	65%
4	64	0%	58%
10	1	53%	85%
10	2	35%	89%
10	4	20%	86%
10	8	12%	88%
10	16	4%	90%
10	32	0%	91%
20	1	76%	74%
20	2	49%	83%
20	4	40%	85%
20	8	21%	88%
20	16	13%	93%
20	32	10%	92%

Table 6: Table representing the correct percentage of classifications for a SVM and our atypicality measure when the NULL hypothesis is true.

a normal distribution, and a testing object was discretely moved away from the centroid of the distribution. So, by keeping the mean vector of the training distribution near $\vec{0}$, the test object was moved by 0.5 units in all dimensions for each new set of simulations. As before, the number of training objects and dimensions fluctuated. The table found in the Appendix section C.5 shows these values and the percentage of test cases that were classified as part of the training set by atypicality and SVM models. One hundred simulations for each set of parameters (trainings objects, number of dimensions, and test object's coordinates) was executed to determine this percentage. Under certain conditions, the code was unable to handle producing multivariate normal scores when the testing value was moved away from the mean. This was the limit set by the simulations to determine how far to move the test object. One can see by looking at Table C.5 that this occurred as the number of training objects increased, along with the dimensionality.

Observation of the table shows that the SVM model quickly determines that the test object is not part of the training set, as the test object moves away from the centroid of the group; as compared to the atypicality model. As before, the SVM model performs better when the ratio of training objects to dimensions is greater than one, where it continues to improve as this ratio increases. The atypicality model maintained it's high performance. Since all the objects were pulled from the standard multivariate normal distribution, we can expect the percentage of "matching" classifications to decrease as the test value moves away from the mean vector. These expectations are shown in a more gradual fashion for the atypicality classification model, as compared to the SVM model.

3.7 Atypicality and SVM Conclusions. Looking at the two tables, we can see that the atypicality handles higher dimensions far better than the SVM model, especially when the the number of dimensions is larger than the number of objects in the training set. This is likely due to the atypicality method reducing the objects to

pair-wise scores, and estimating parameters between them, whereas the SVM model must consider all dimensions presented.

Classifying data when the null hypothesis was true proved an easier task for the atypicality model. When the null hypothesis was not true, the atypicality had a gradual decrease in positive classifications as compared to the sharper decrease by the SVM.

One thing to note is that setting the parameters for the one-class SVM can yield different outcomes. There is no set method to determining these parameters, and determining "good" ones can be quite difficult. Future work could be finding an optimal ν parameter for the one-class SVM model in order to produce "better" results.

This would lead us to recommend using the atypicality approach when dealing with smaller number of objects in higher dimensions. This is a common occurrence in certain fields, such as the forensic community; where only small amounts of evidence can be collected.

REFERENCES

[1] Bradley J Adams. The diversity of adult dental patterns in the united states and the implications for personal identification. *Journal of forensic sciences*, 48(3):497–503, 2003.

[2] John Aitchison and Ian Robert Dunsmore. *Statistical prediction analysis*. CUP Archive, 1980.

[3] Anil Alexander, Filippo Botti, D Dessimoz, and ANDRZEJ Drygajlo. The effect of mismatched recording conditions on human and automatic speaker recognition in forensic applications. *Forensic science international*, 146:S95–S99, 2004.

[4] Christophe Ambroise and Geoffrey J McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, 99(10):6562–6566, 2002.

[5] TW Anderson. *Multivariate statistical analysis*. 1984.

[6] Roberto J Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, pages 131–140. ACM, 2007.

[7] Imed Bouchrika, Michaela Goffredo, John Carter, and Mark Nixon. On using gait in forensic biometrics. *Journal of forensic sciences*, 56(4):882–889, 2011.

[8] Michael PS Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, and David Haussler. Support vector machine classification of microarray gene expression data. *University of California, Santa Cruz, Technical Report UCSC-CRL-99-09*, 1999.

[9] Michael PS Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S Furey, Manuel Ares, and David Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.

[10] William G Cochran. The distribution of quadratic forms in a normal system, with applications to the analysis of covariance. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 30, pages 178–191. Cambridge Univ Press, 1934.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[13] Philip De Chazal, John Flynn, and Richard B Reilly. Automated processing of shoeprint images based on the fourier transform for use in forensic science. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):341–350, 2005.

[14] Djavanshir Djozan, Tahmineh Baheri, Ghader Karimian, and Masomeh Shahidi. Forensic discrimination of blue ballpoint pen inks based on thin layer chromatography and image analysis. *Forensic Science International*, 179(2):199–205, 2008.

[15] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE, 2000.

[16] Vojtech Franc and Václav Hlavác. Multi-class support vector machine. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 236–239. IEEE, 2002.

[17] Glenn M Fung and Olvi L Mangasarian. Multicategory proximal support vector machine classifiers. *Machine learning*, 59(1-2):77–97, 2005.

[18] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[19] Barry D Gaudette. A supplementary discussion of probabilities and human hair comparisons. *Journal of Forensic Science*, 27(2):279–289, 1982.

[20] Joaquin Gonzalez-Rodriguez, Andrzej Drygajlo, Daniel Ramos-Castro, Marta Garcia-Gomar, and Javier Ortega-Garcia. Robust estimation, interpretation and assessment of likelihood ratios in forensic speaker recognition. *Computer Speech & Language*, 20(2):331–355, 2006.

[21] Joaquin Gonzalez-Rodriguez, Phil Rose, Daniel Ramos, Doroteo T Toledano, and Javier Ortega-Garcia. Emulating dna: Rigorous quantification of evidential weight in transparent and testable forensic speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2104–2115, 2007.

[22] PS Groom, ME Lawton, and A Cooper. Are they a pair? *Journal of the Forensic Science Society*, 27(3):189–192, 1987.

[23] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[24] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[25] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[26] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust support vector machines for anomaly detection in computer security. In *ICMLA*, pages 168–174, 2003.

[27] Marilena V Iorio, Manuela Ferracin, Chang-Gong Liu, Angelo Veronese, Riccardo Spizzo, Silvia Sabbioni, Eros Magri, Massimo Pedriali, Muller Fabbri, Manuela Campiglio, et al. Microrna gene expression deregulation in human breast cancer. *Cancer research*, 65(16):7065–7070, 2005.

[28] Dino Isa, Lam H Lee, VP Kallimani, and Rajprasad Rajkumar. Text document preprocessing with the bayes formula for classification using the support vector machine. *IEEE Transactions on Knowledge and Data engineering*, 20(9):1264–1272, 2008.

[29] Alan Julian Izenman. Modern multivariate statistical techniques. *Regression, classification and manifold learning*, 2008.

[30] Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer, 2013.

[31] Anil K Jain, Salil Prabhakar, and Sharath Pankanti. On the similarity of identical twin fingerprints. *Pattern Recognition*, 35(11):2653–2663, 2002.

[32] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[33] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[34] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 128–136. ACM, 2001.

[35] DA Jones. Blood samples: probability of discrimination. *Journal of the Forensic Science Society*, 12(2):355–359, 1972.

[36] James Tin-Yau Kwok. Automated text categorization using support vector machine. In *In Proceedings of the International Conference on Neural Information Processing (ICONIP*. Citeseer, 1998.

[37] Brett Lantz. *Machine Learning with R*. Packt Publishing, 2nd edition, 2015.

[38] Yoonkyung Lee and Cheol-Koo Lee. Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics*, 19(9):1132–1139, 2003.

[39] Edda Leopold and Jorg Kindermann. Text categorization with support vector machines. how to represent texts in input space. *Machine Learning*, 46(1-3):423–444, 2002.

[40] M Lichman. Uci machine learning repository. `http://archive.ics.uci.edu/ml`.

[41] Amy B Lock and Max D Morris. Significance of angle in the statistical comparison of forensic tool marks. *Technometrics*, 55(4):548–561, 2013.

[42] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[43] G Massonnet and W Stoecklein. Identification of organic pigments in coatings: applications to red automotive topcoats: Part iii: Raman spectroscopy (nir ft-raman). *Science & justice*, 39(3):181–187, 1999.

[44] Geoffrey McLachlan. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

[45] Didier Meuwly and Andrzej Drygajlo. Forensic speaker recognition based on a bayesian framework and gaussian mixture modelling (gmm). In *2001: A Speaker Odyssey-The Speaker Recognition Workshop*, 2001.

[46] Stefan Michiels, Serge Koscielny, and Catherine Hill. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *The Lancet*, 365(9458):488–492, 2005.

[47] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[48] John Miller, Donald Gantz, and Christopher Saunders. On parametric models for pairwise comparisons. *None*, 2014.

[49] Sayan Mukherjee, P Tamayo, D Slonim, A Verri, T Golub, J Mesirov, and T Poggio. Support vector machine classification of microarray data. 1999.

[50] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[51] JB Parker. A statistical treatment of identification problems. *Journal of the Forensic Science Society*, 6(1):33–39, 1966.

[52] JB Parker. The mathematical evaluation of numerical evidence. *Journal of the Forensic Science Society*, 7(3):134–144, 1967.

[53] Sihua Peng, Qianghua Xu, Xuefeng Bruce Ling, Xiaoning Peng, Wei Du, and Liangbiao Chen. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS letters*, 555(2):358–362, 2003.

[54] P Jonathon Phillips, Patrick J Flynn, Todd Scruggs, Kevin W Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on*, volume 1, pages 947–954. IEEE, 2005.

[55] Scott L Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M Sturla, Michael Angelo, Margaret E McLaughlin, John YH Kim, Liliana C Goumnerova, Peter M Black, Ching Lau, et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[56] Weiliang Qiu and Harry Joe. *clusterGeneration: random cluster generation (with specified degree of separation)*, 2013. R package version 1.3.1.

[57] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[58] Sridhar Ramaswamy, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, Michael Reich, Eva Latulippe, Jill P Mesirov, et al. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences*, 98(26):15149–15154, 2001.

[59] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[60] Jason DM Rennie and Ryan Rifkin. Improving multiclass text classification with the support vector machine. 2001.

[61] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern recognition letters*, 24(13):2115–2125, 2003.

[62] Bernhard Scholkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[63] Bernhard Scholkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588, 1999.

[64] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[65] CS Shin, KI Kim, MH Park, and Hang Joon Kim. Support vector machine-based text detection in digital video. In *Neural networks for signal processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 634–641. IEEE, 2000.

[66] KW Smalldon and AC Moffat. The calculation of discriminating power for a series of correlated attributes. *Journal of the Forensic Science Society*, 13(4):291–295, 1973.

[67] Dirk Smeets, Peter Claes, Dirk Vandermeulen, and John Gerald Clement. Objective 3d face recognition: Evolution, approaches and challenges. *Forensic science international*, 201(1):125–132, 2010.

[68] Alexander Statnikov, Constantin F Aliferis, Ioannis Tsamardinos, Douglas Hardin, and Shawn Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

[69] Alexander Statnikov, Lily Wang, and Constantin F Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, 9(1):1, 2008.

[70] Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. Web classification using support vector machine. In *Proceedings of the 4th international workshop on Web information and data management*, pages 96–99. ACM, 2002.

[71] Andrew H Sung and Srinivas Mukkamala. Identifying important features for intrusion detection using support vector machines and neural networks. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pages 209–216. IEEE, 2003.

[72] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11):1191–1199, 1999.

[73] CF Tippett, VJ Emerson, MJ Fereday, F Lawton, A Richardson, LT Jones, and Miss SM Lampert. The evidential value of the comparison of paint flakes from sources other than vehicles. *Journal of the Forensic Science Society*, 8(2-3):61–65, 1968.

[74] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[75] Peng Wang, Qiang Ji, and James L Wayman. Modeling and predicting face recognition system performance based on analysis of similarity scores. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4), 2007.

[76] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

[77] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, pages 219–224, 1999.

[78] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, 21(8):879–886, 2008.

APPENDIX

Appendices

A Hardware & Software

A.1 Hardware

The code used was run on a MacBook Pro, Mid 2012.

Processor: 2.3 GHz Intel Core i7

Memory: 16 GB 1600 MHz DDR3

Software: Mac OS X Lion 10.7.5 (11G63b)

A.2 Software

R 3.0.2 GUI 1.62 Snow Leopard build (6558)

R for Mac OS X GUI written by:

Simon Urbanek

Hans-JÃűrg Bibiko

Stefano M. Iacus

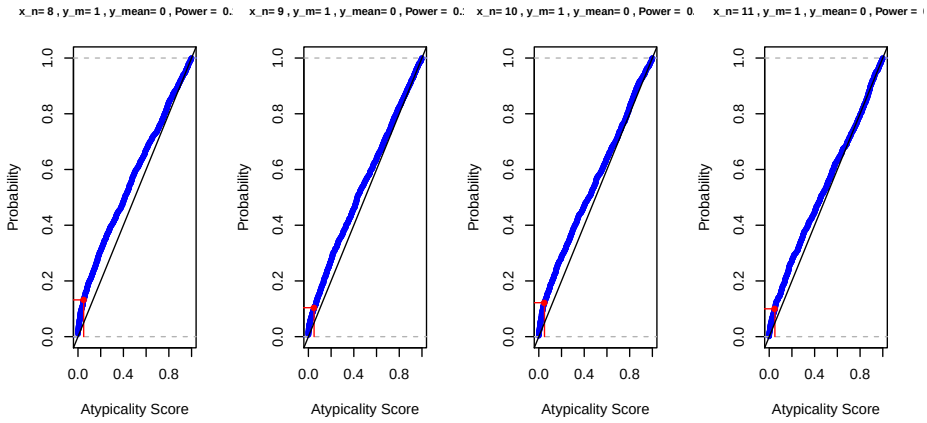
A.3 Computational Limitations

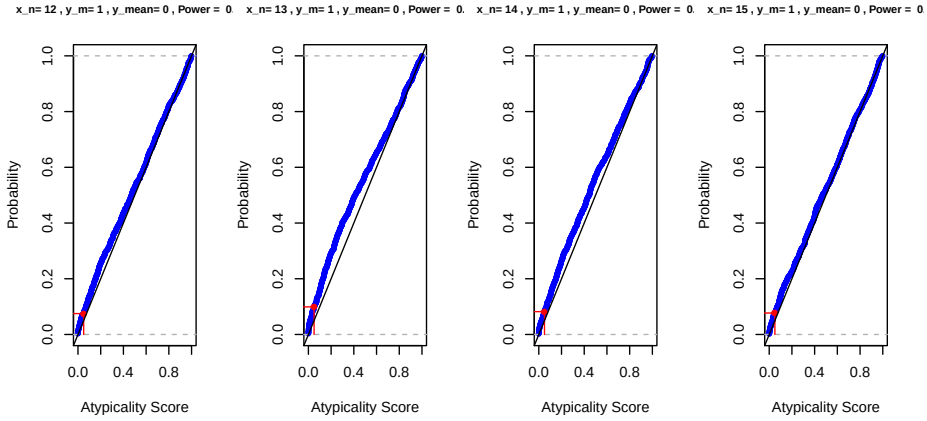
It is clear that the estimation of our parameters relies heavily on the size of our initial population; the larger the population, the more accurate our parameters are likely to be. Current simulations require at least four objects in the population to estimate the parameters.

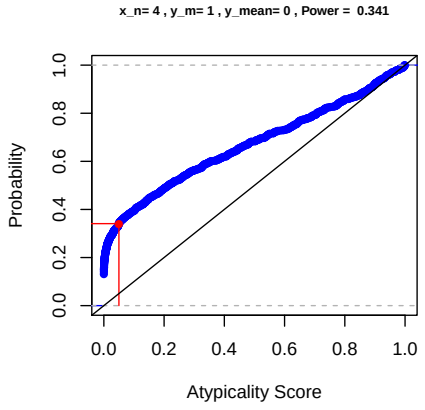
B Results Tables and Graphs

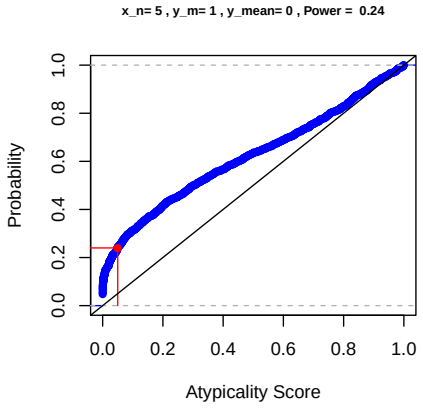
C Plots

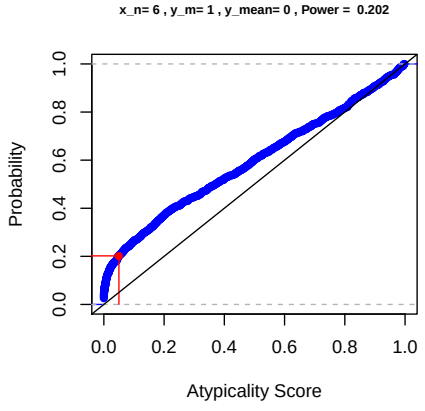
C.1 True Null Hypothesis Plots

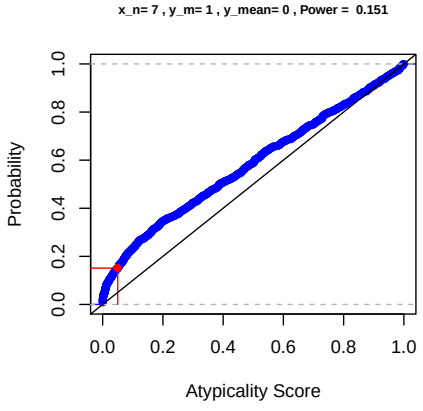

x_n= 8 , y_m= 1 , y_mean= 0 , Power = 0.
x_n= 9 , y_m= 1 , y_mean= 0 , Power = 0.
x_n= 10 , y_m= 1 , y_mean= 0 , Power = 0.
x_n= 11 , y_m= 1 , y_mean= 0 , Power =
Probability
Atypicality Score

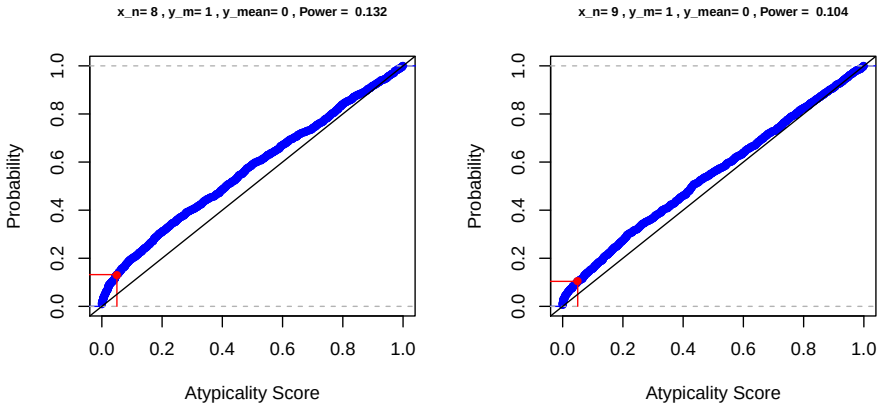

x_n= 12 , y_m= 1 , y_mean= 0 , Power = 0
x_n= 13 , y_m= 1 , y_mean= 0 , Power = 0.
x_n= 14 , y_m= 1 , y_mean= 0 , Power = 0.
x_n= 15 , y_m= 1 , y_mean= 0 , Power = 0
Probability
Atypicality Score

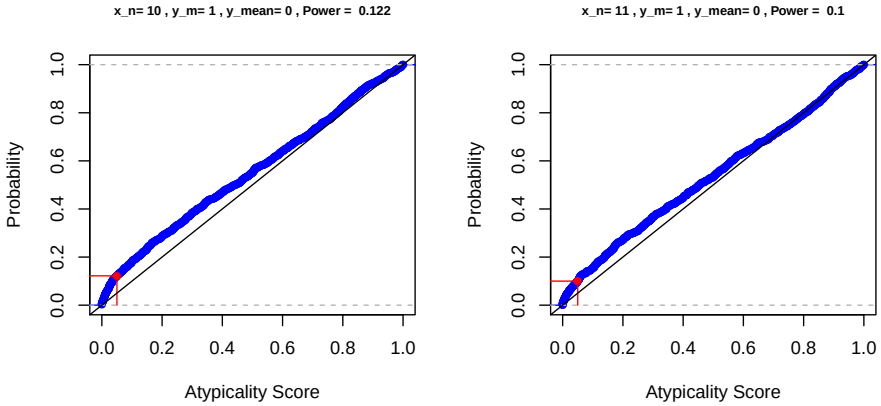

x_n= 4 , y_m= 1 , y_mean= 0 , Power = 0.341
Probability
Atypicality Score

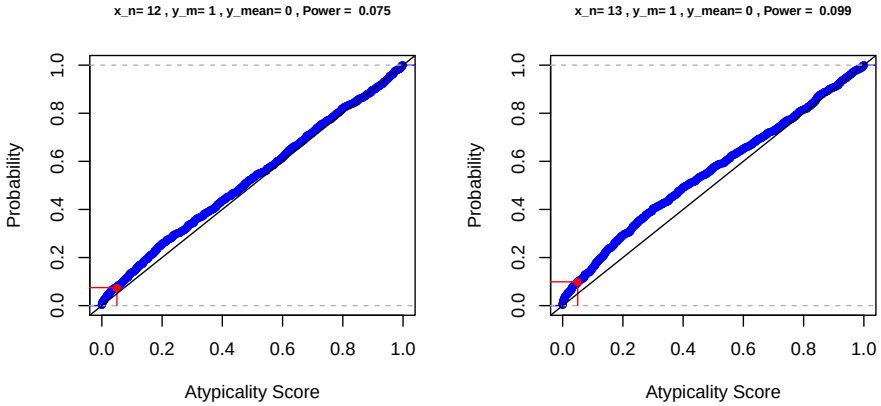

x_n= 5 , y_m= 1 , y_mean= 0 , Power = 0.24
Probability
Atypicality Score

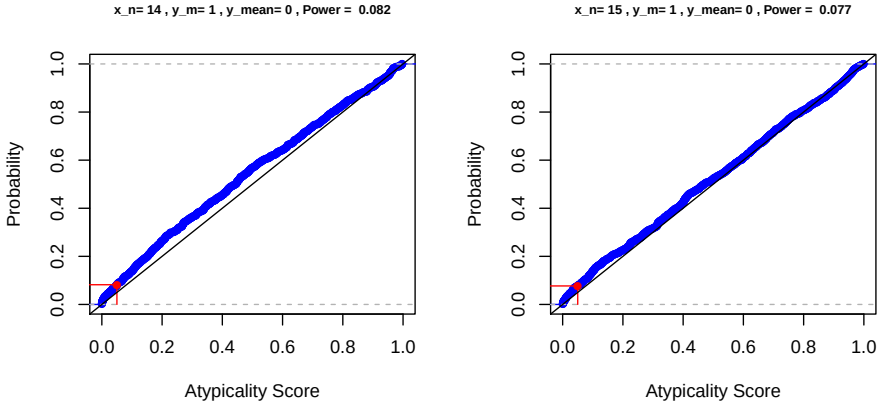

x_n= 6 , y_m= 1 , y_mean= 0 , Power = 0.202
Probability
Atypicality Score


x_n= 7 , y_m= 1 , y_mean= 0 , Power = 0.151
Probability
Atypicality Score

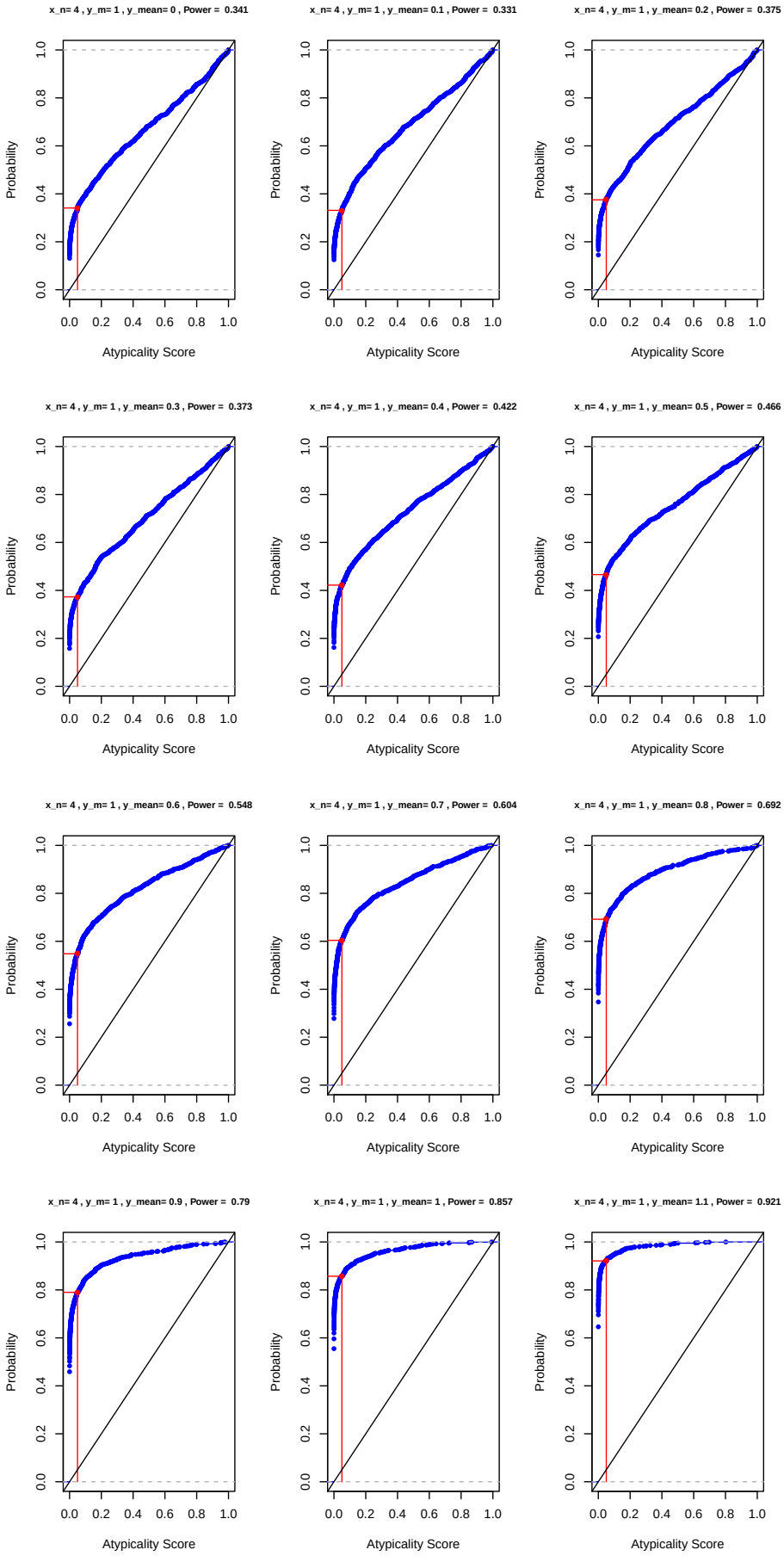

x_n= 8 , y_m= 1 , y_mean= 0 , Power = 0.132
x_n= 9 , y_m= 1 , y_mean= 0 , Power = 0.104
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

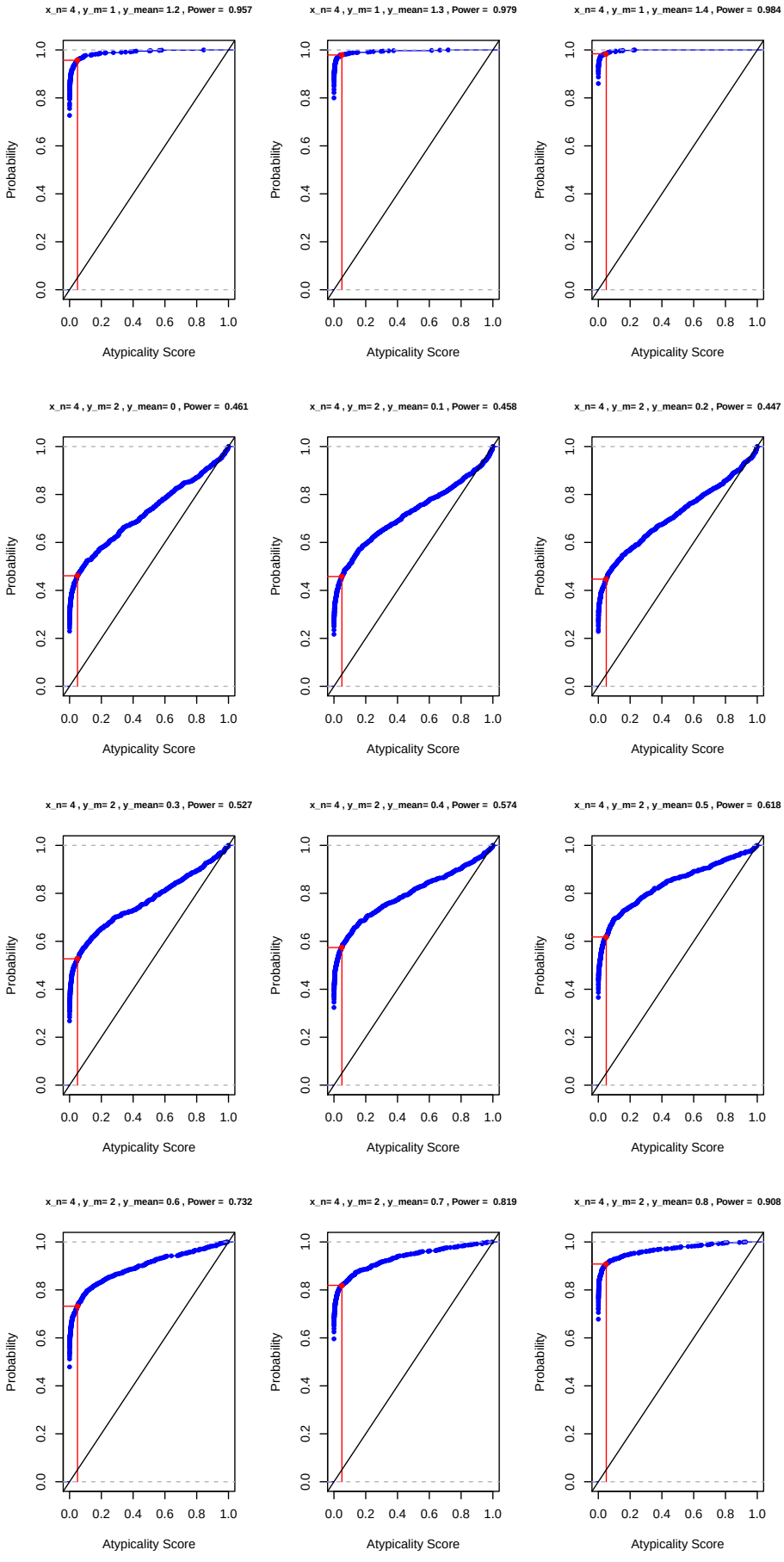

x_n= 10 , y_m= 1 , y_mean= 0 , Power = 0.122
x_n= 11 , y_m= 1 , y_mean= 0 , Power = 0.1
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

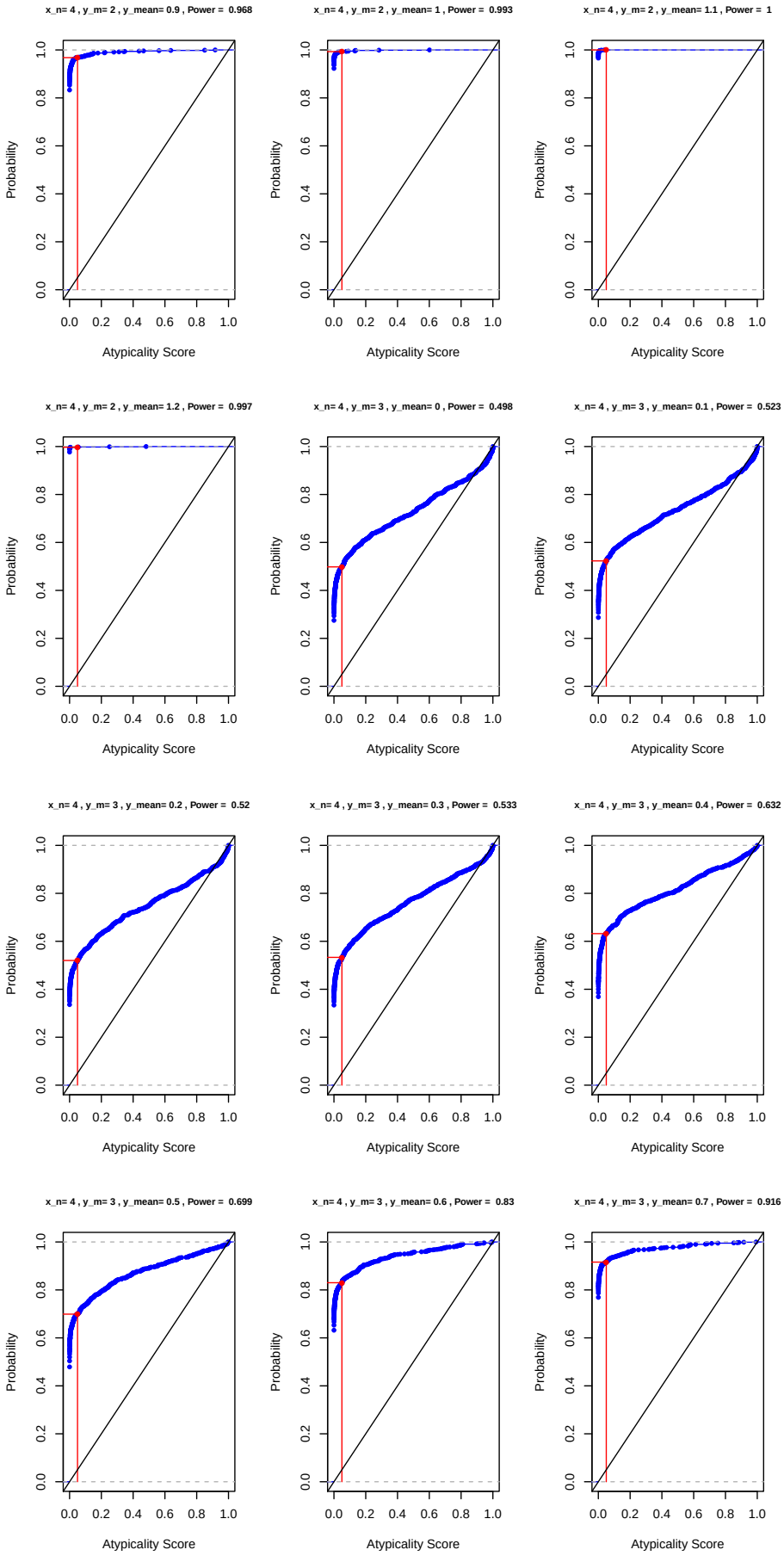

x_n= 12 , y_m= 1 , y_mean= 0 , Power = 0.075
x_n= 13 , y_m= 1 , y_mean= 0 , Power = 0.099
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

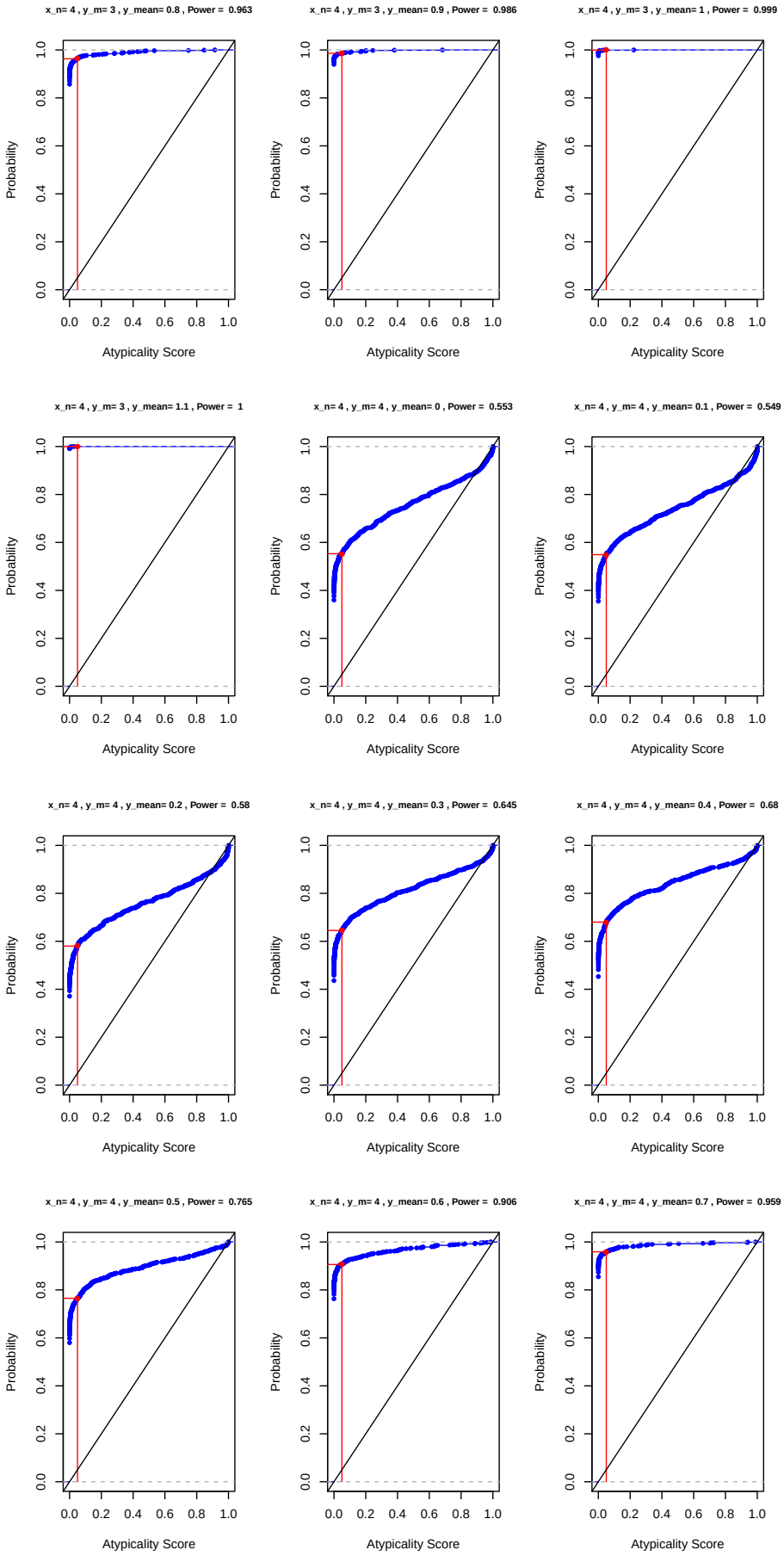

x_n= 14 , y_m= 1 , y_mean= 0 , Power = 0.082
x_n= 15 , y_m= 1 , y_mean= 0 , Power = 0.077
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

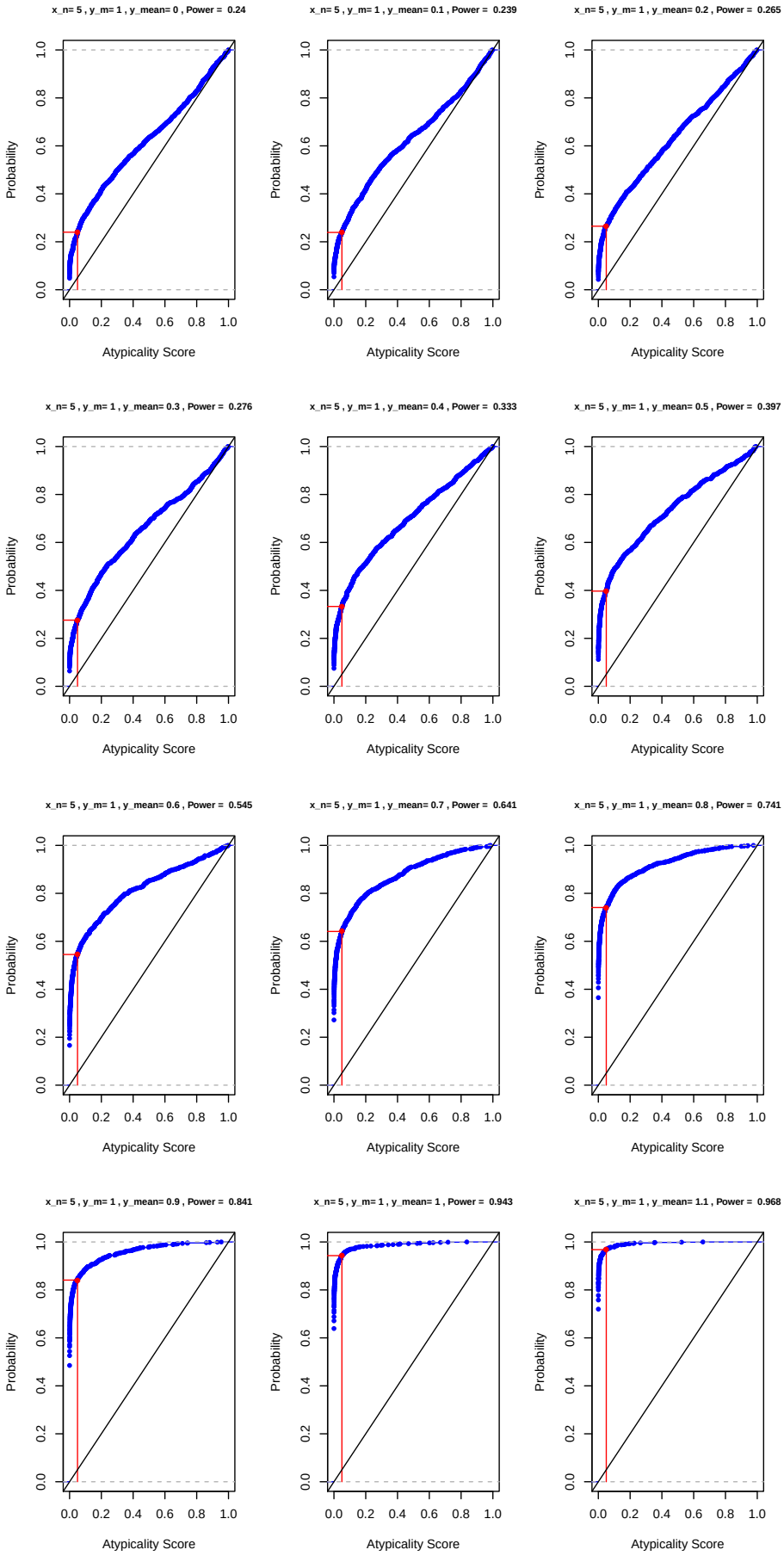
C.2 Power Plots

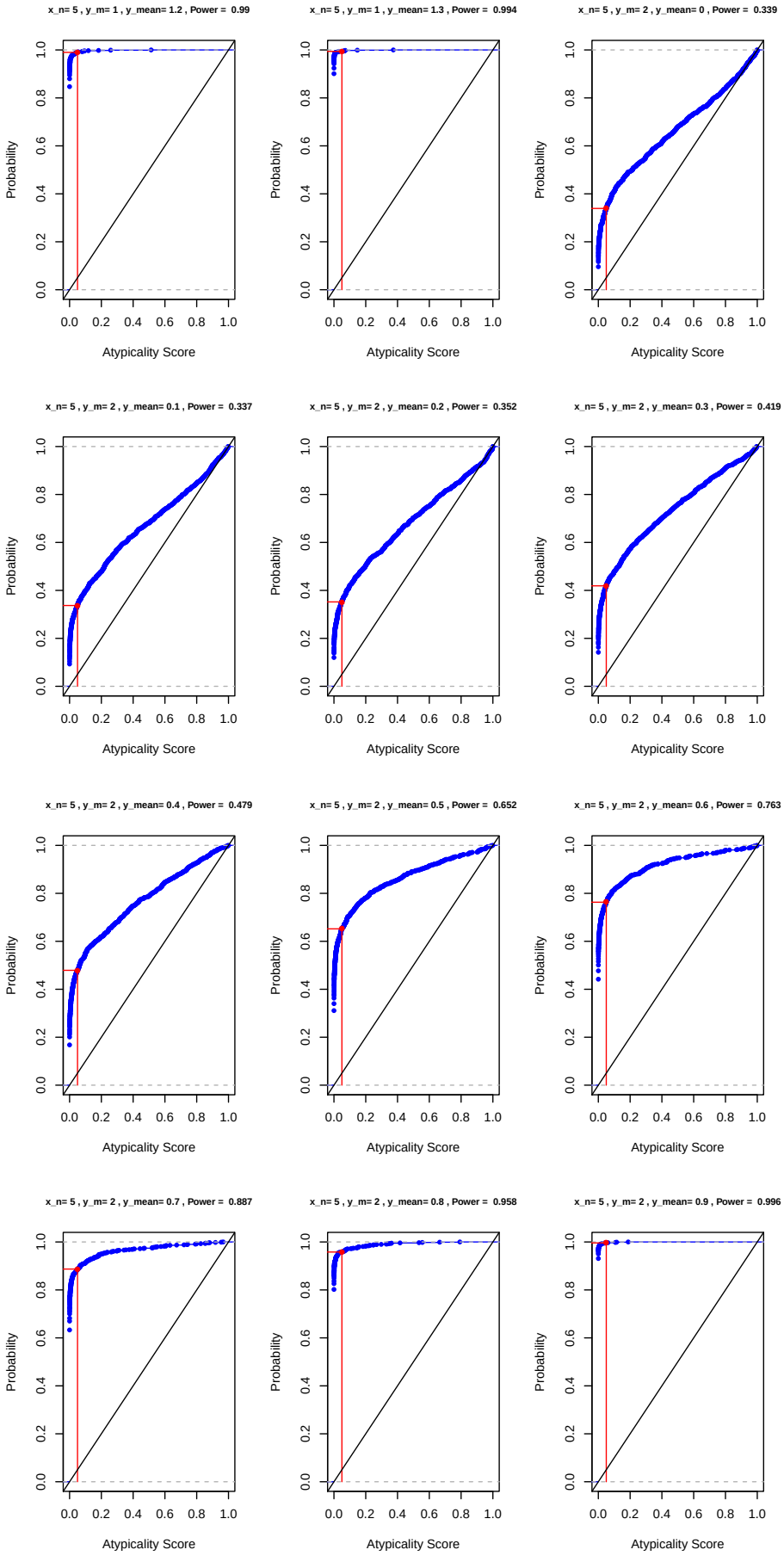

x_n= 4 , y_m= 1 , y_mean= 0 , Power = 0.341
x_n= 4 , y_m= 1 , y_mean= 0.1 , Power = 0.331
x_n= 4 , y_m= 1 , y_mean= 0.2 , Power = 0.375
x_n= 4 , y_m= 1 , y_mean= 0.3 , Power = 0.373
x_n= 4 , y_m= 1 , y_mean= 0.4 , Power = 0.422
x_n= 4 , y_m= 1 , y_mean= 0.5 , Power = 0.466
x_n= 4 , y_m= 1 , y_mean= 0.6 , Power = 0.548
x_n= 4 , y_m= 1 , y_mean= 0.7 , Power = 0.604
x_n= 4 , y_m= 1 , y_mean= 0.8 , Power = 0.692
x_n= 4 , y_m= 1 , y_mean= 0.9 , Power = 0.79
x_n= 4 , y_m= 1 , y_mean= 1 , Power = 0.857
x_n= 4 , y_m= 1 , y_mean= 1.1 , Power = 0.921
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

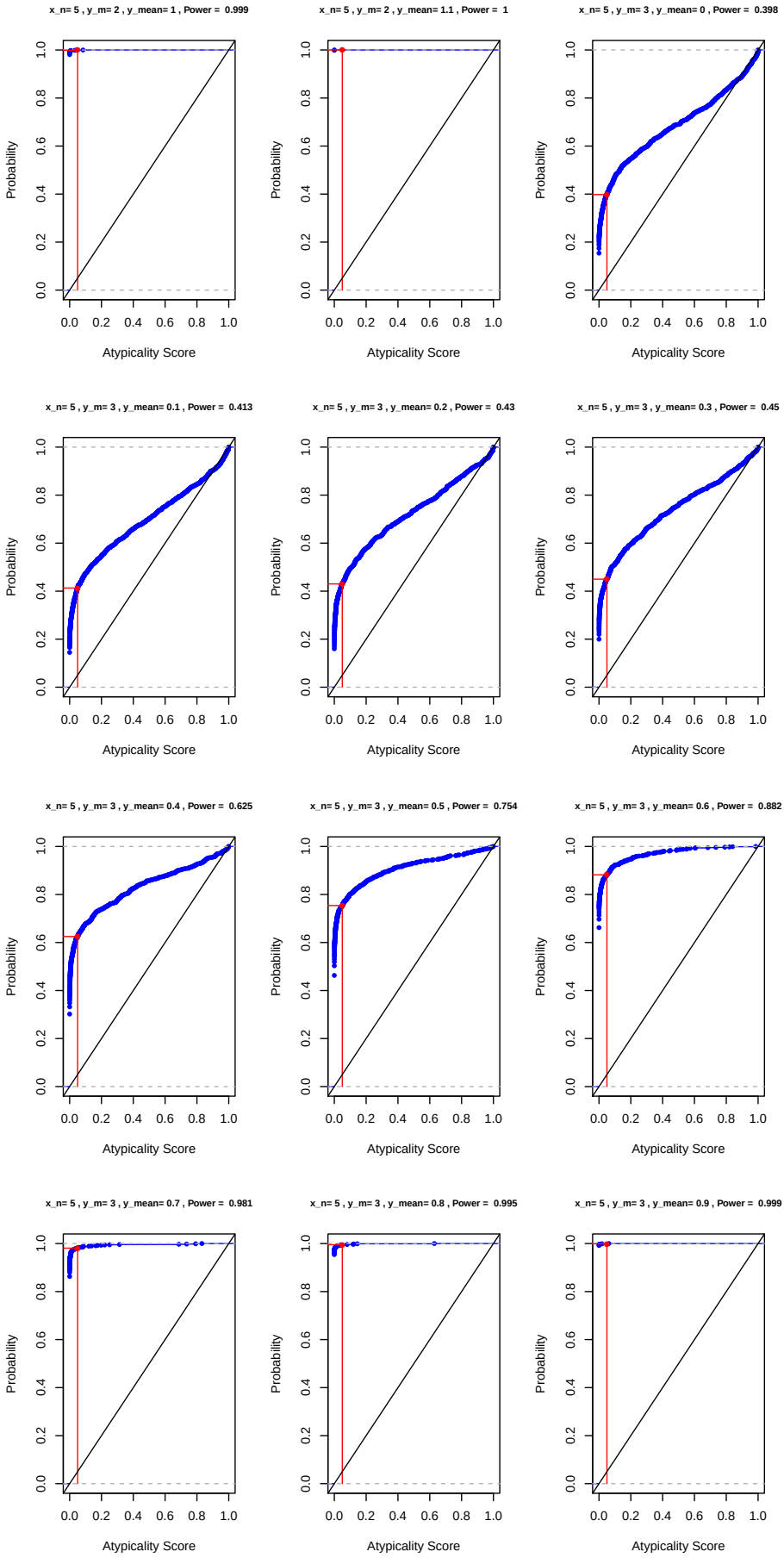

x_n= 4 , y_m= 1 , y_mean= 1.2 , Power = 0.957
x_n= 4 , y_m= 1 , y_mean= 1.3 , Power = 0.979
x_n= 4 , y_m= 1 , y_mean= 1.4 , Power = 0.984
x_n= 4 , y_m= 2 , y_mean= 0 , Power = 0.461
x_n= 4 , y_m= 2 , y_mean= 0.1 , Power = 0.458
x_n= 4 , y_m= 2 , y_mean= 0.2 , Power = 0.447
x_n= 4 , y_m= 2 , y_mean= 0.3 , Power = 0.527
x_n= 4 , y_m= 2 , y_mean= 0.4 , Power = 0.574
x_n= 4 , y_m= 2 , y_mean= 0.5 , Power = 0.618
x_n= 4 , y_m= 2 , y_mean= 0.6 , Power = 0.732
x_n= 4 , y_m= 2 , y_mean= 0.7 , Power = 0.819
x_n= 4 , y_m= 2 , y_mean= 0.8 , Power = 0.908
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

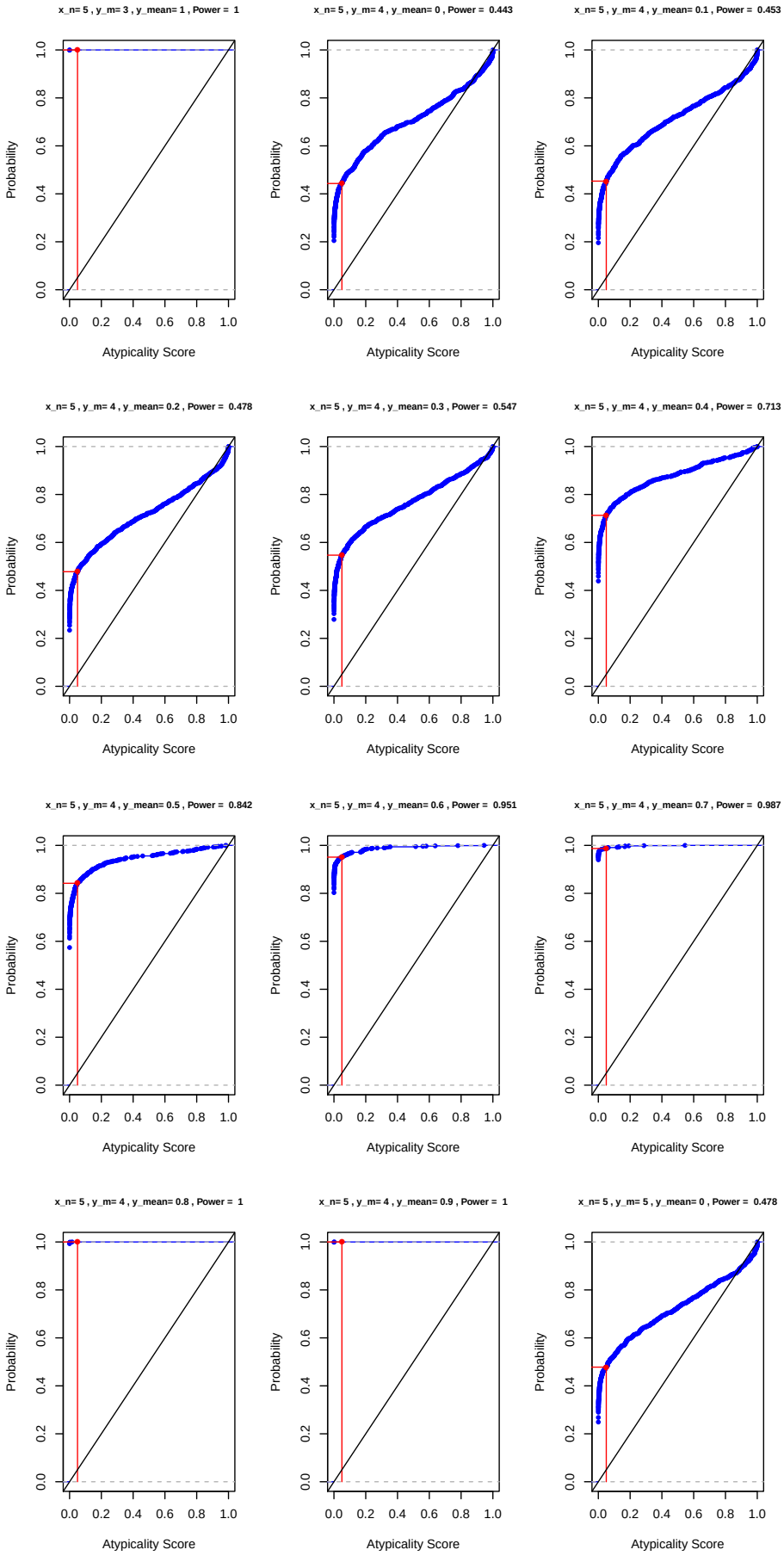

x_n= 4 , y_m= 2 , y_mean= 0.9 , Power = 0.968
x_n= 4 , y_m= 2 , y_mean= 1 , Power = 0.993
x_n= 4 , y_m= 2 , y_mean= 1.1 , Power = 1
x_n= 4 , y_m= 2 , y_mean= 1.2 , Power = 0.997
x_n= 4 , y_m= 3 , y_mean= 0 , Power = 0.498
x_n= 4 , y_m= 3 , y_mean= 0.1 , Power = 0.523
x_n= 4 , y_m= 3 , y_mean= 0.2 , Power = 0.52
x_n= 4 , y_m= 3 , y_mean= 0.3 , Power = 0.533
x_n= 4 , y_m= 3 , y_mean= 0.4 , Power = 0.632
x_n= 4 , y_m= 3 , y_mean= 0.5 , Power = 0.699
x_n= 4 , y_m= 3 , y_mean= 0.6 , Power = 0.83
x_n= 4 , y_m= 3 , y_mean= 0.7 , Power = 0.916
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

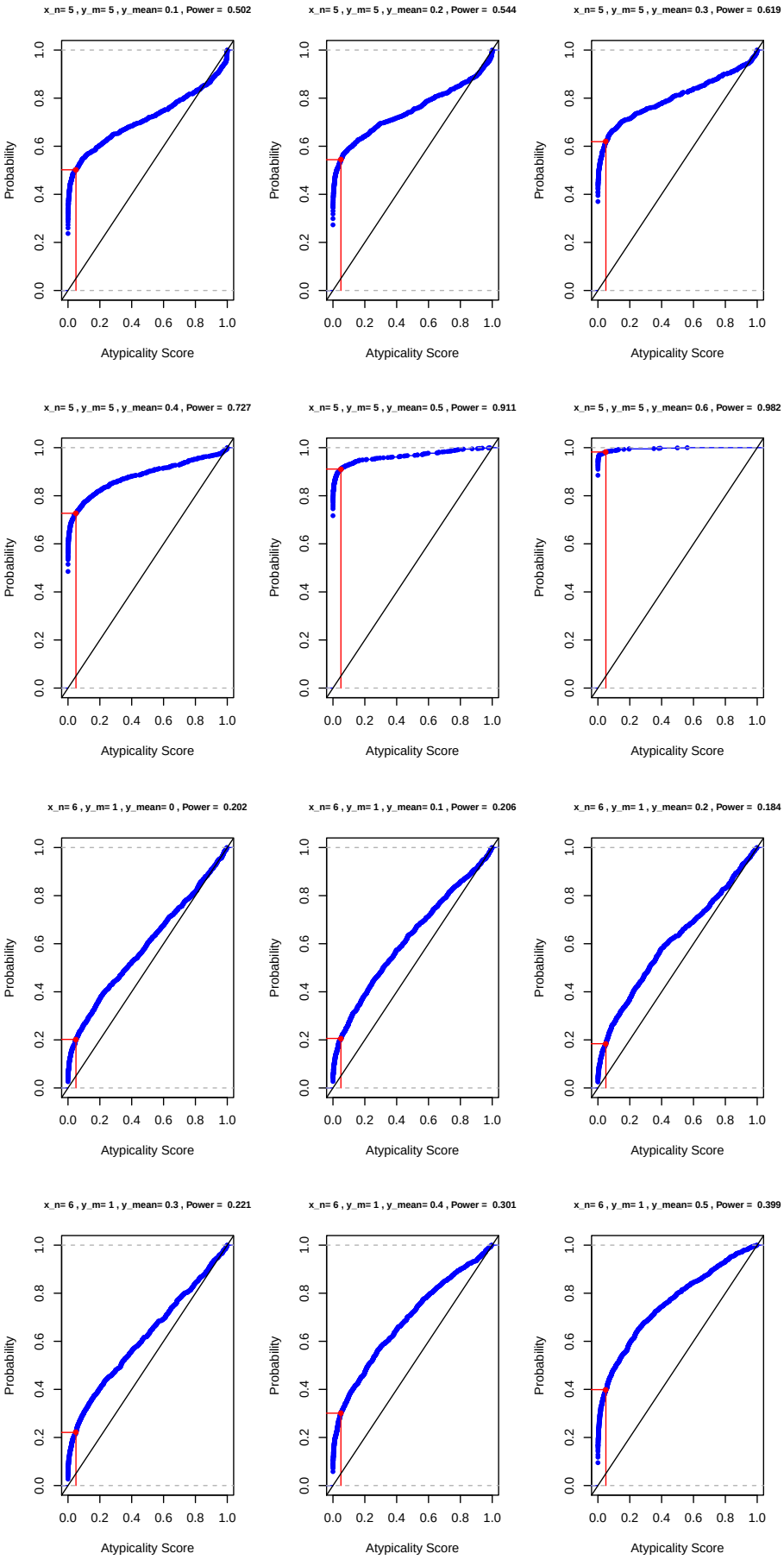

x_n= 4 , y_m= 3 , y_mean= 0.8 , Power = 0.963
x_n= 4 , y_m= 3 , y_mean= 0.9 , Power = 0.986
x_n= 4 , y_m= 3 , y_mean= 1 , Power = 0.999
x_n= 4 , y_m= 3 , y_mean= 1.1 , Power = 1
x_n= 4 , y_m= 4 , y_mean= 0 , Power = 0.553
x_n= 4 , y_m= 4 , y_mean= 0.1 , Power = 0.549
x_n= 4 , y_m= 4 , y_mean= 0.2 , Power = 0.58
x_n= 4 , y_m= 4 , y_mean= 0.3 , Power = 0.645
x_n= 4 , y_m= 4 , y_mean= 0.4 , Power = 0.68
x_n= 4 , y_m= 4 , y_mean= 0.5 , Power = 0.765
x_n= 4 , y_m= 4 , y_mean= 0.6 , Power = 0.906
x_n= 4 , y_m= 4 , y_mean= 0.7 , Power = 0.959
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

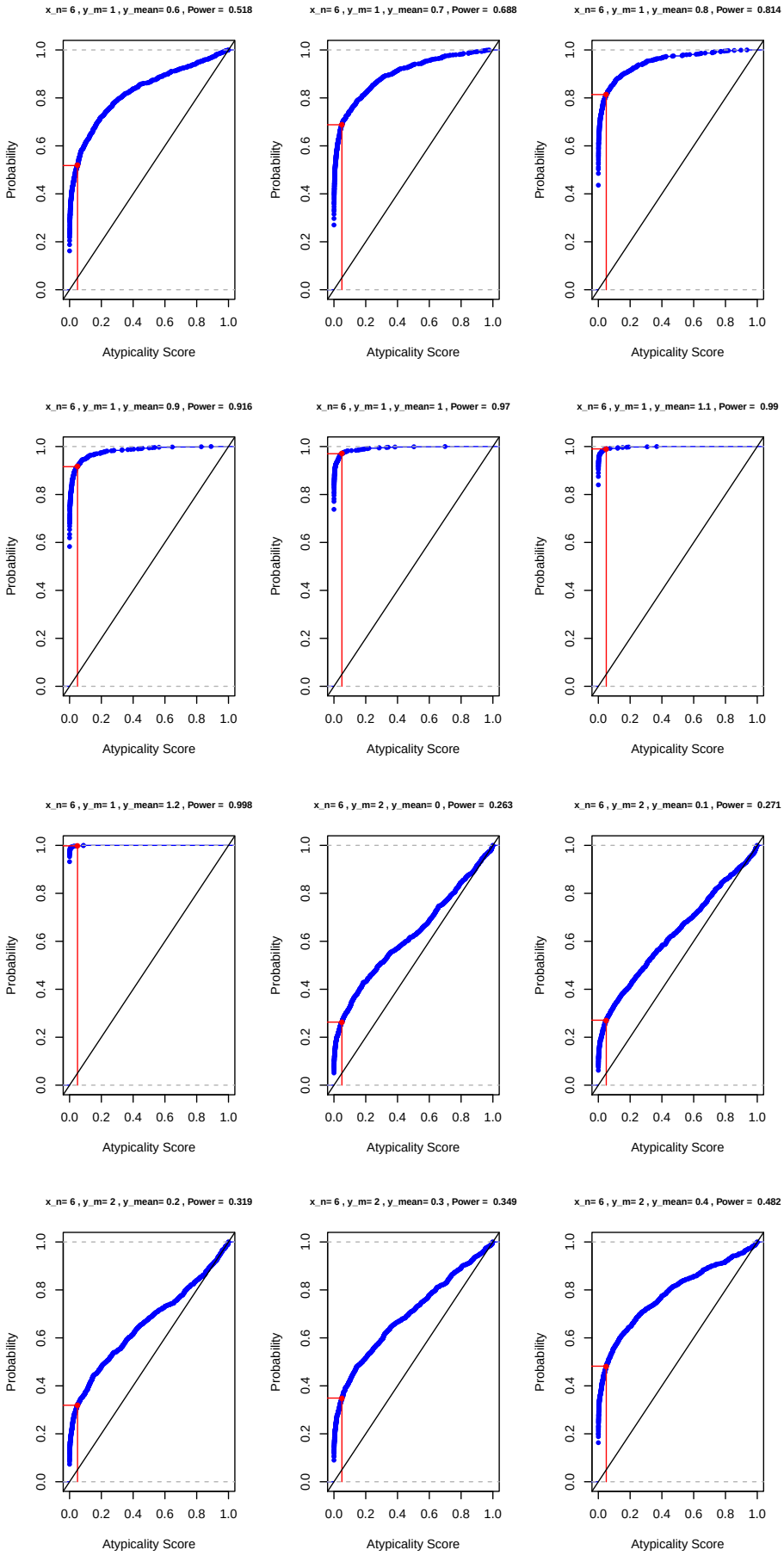

x_n= 5 , y_m= 1 , y_mean= 0 , Power = 0.24
x_n= 5 , y_m= 1 , y_mean= 0.1 , Power = 0.239
x_n= 5 , y_m= 1 , y_mean= 0.2 , Power = 0.265
x_n= 5 , y_m= 1 , y_mean= 0.3 , Power = 0.276
x_n= 5 , y_m= 1 , y_mean= 0.4 , Power = 0.333
x_n= 5 , y_m= 1 , y_mean= 0.5 , Power = 0.397
x_n= 5 , y_m= 1 , y_mean= 0.6 , Power = 0.545
x_n= 5 , y_m= 1 , y_mean= 0.7 , Power = 0.641
x_n= 5 , y_m= 1 , y_mean= 0.8 , Power = 0.741
x_n= 5 , y_m= 1 , y_mean= 0.9 , Power = 0.841
x_n= 5 , y_m= 1 , y_mean= 1 , Power = 0.943
x_n= 5 , y_m= 1 , y_mean= 1.1 , Power = 0.968
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

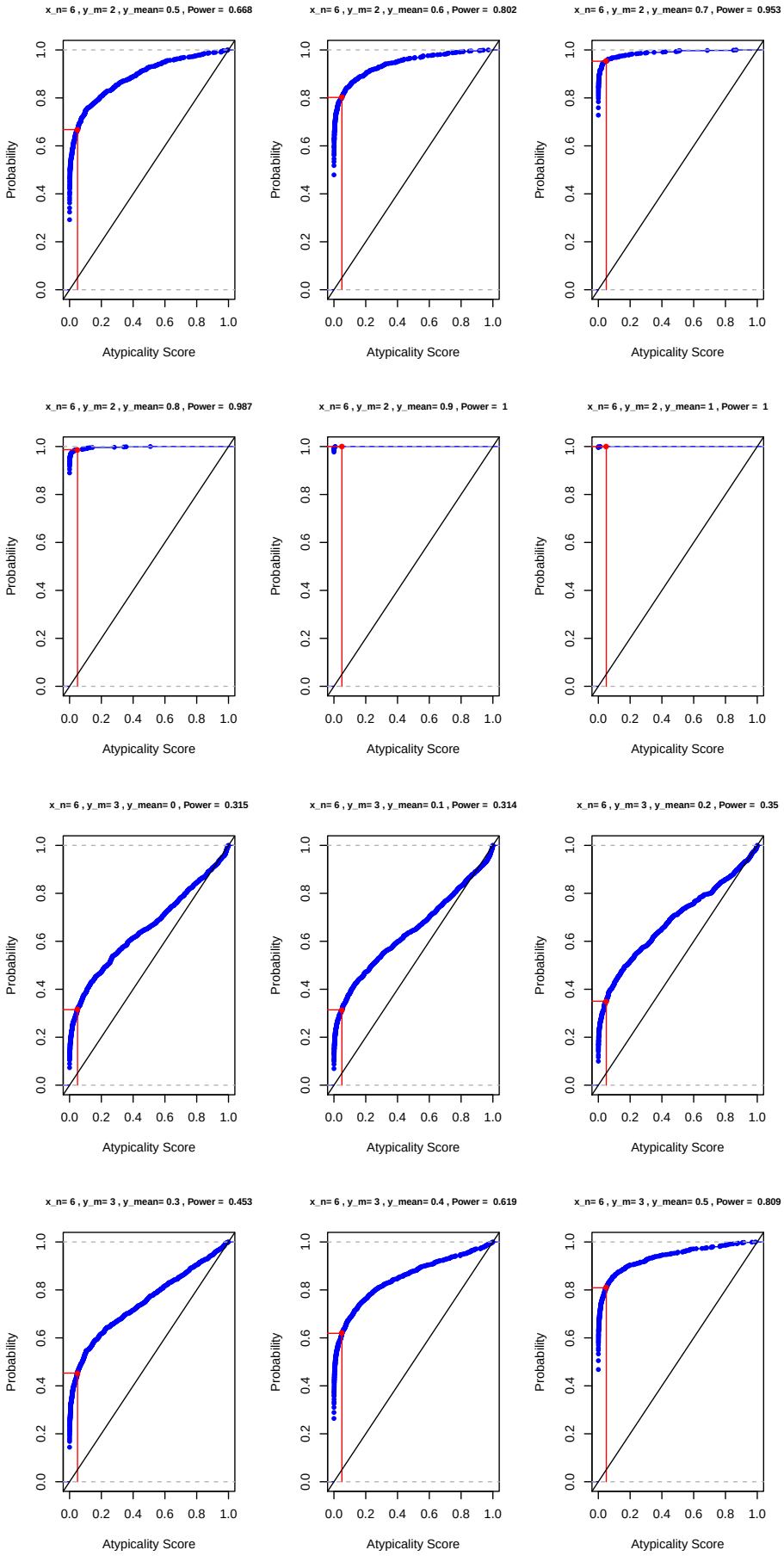

x_n= 5 , y_m= 1 , y_mean= 1.2 , Power = 0.99
x_n= 5 , y_m= 1 , y_mean= 1.3 , Power = 0.994
x_n= 5 , y_m= 2 , y_mean= 0 , Power = 0.339
x_n= 5 , y_m= 2 , y_mean= 0.1 , Power = 0.337
x_n= 5 , y_m= 2 , y_mean= 0.2 , Power = 0.352
x_n= 5 , y_m= 2 , y_mean= 0.3 , Power = 0.419
x_n= 5 , y_m= 2 , y_mean= 0.4 , Power = 0.479
x_n= 5 , y_m= 2 , y_mean= 0.5 , Power = 0.652
x_n= 5 , y_m= 2 , y_mean= 0.6 , Power = 0.763
x_n= 5 , y_m= 2 , y_mean= 0.7 , Power = 0.887
x_n= 5 , y_m= 2 , y_mean= 0.8 , Power = 0.958
x_n= 5 , y_m= 2 , y_mean= 0.9 , Power = 0.996
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

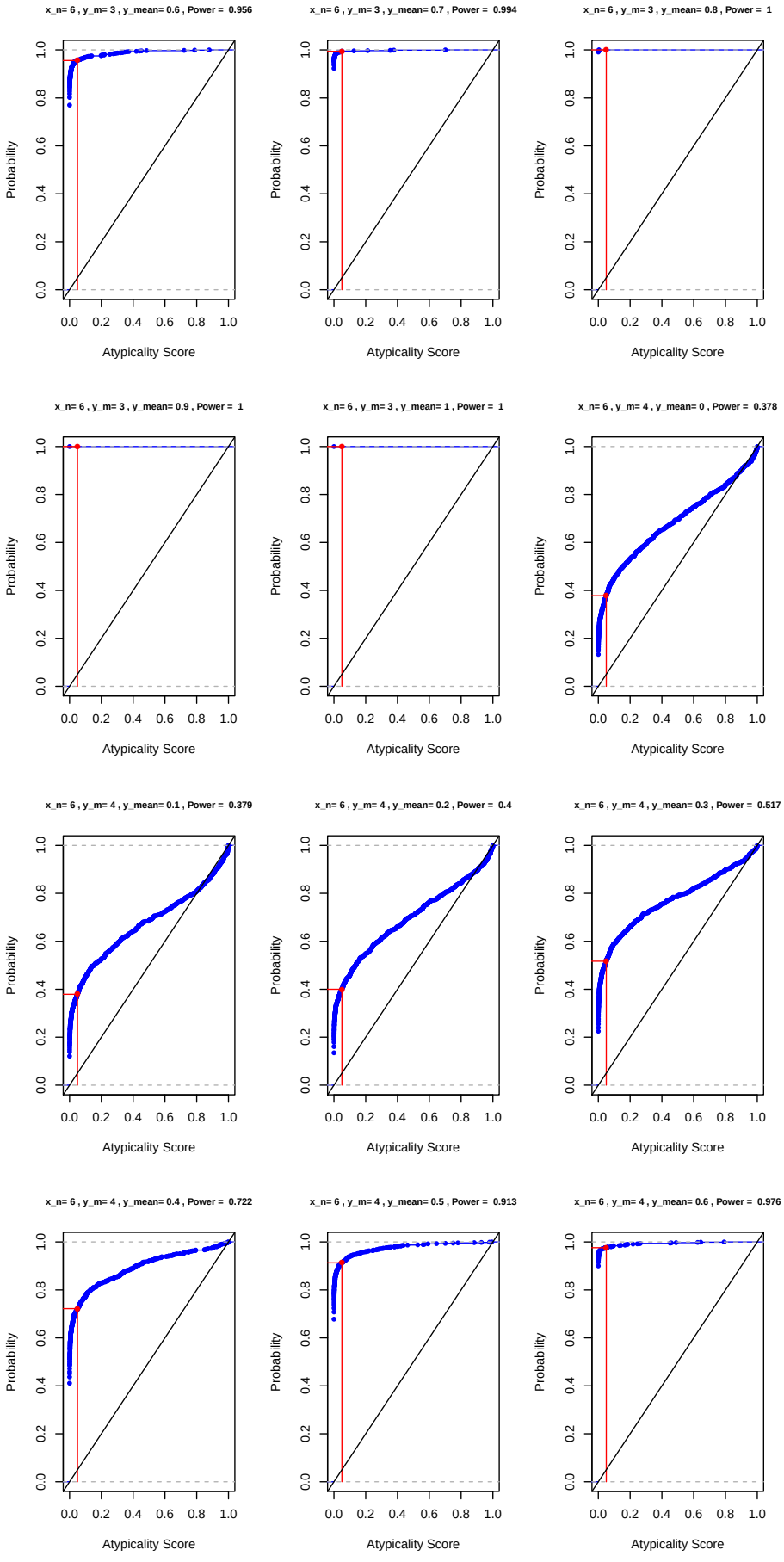

x_n= 5 , y_m= 2 , y_mean= 1 , Power = 0.999
x_n= 5 , y_m= 2 , y_mean= 1.1 , Power = 1
x_n= 5 , y_m= 3 , y_mean= 0 , Power = 0.398
x_n= 5 , y_m= 3 , y_mean= 0.1 , Power = 0.413
x_n= 5 , y_m= 3 , y_mean= 0.2 , Power = 0.43
x_n= 5 , y_m= 3 , y_mean= 0.3 , Power = 0.45
x_n= 5 , y_m= 3 , y_mean= 0.4 , Power = 0.625
x_n= 5 , y_m= 3 , y_mean= 0.5 , Power = 0.754
x_n= 5 , y_m= 3 , y_mean= 0.6 , Power = 0.882
x_n= 5 , y_m= 3 , y_mean= 0.7 , Power = 0.981
x_n= 5 , y_m= 3 , y_mean= 0.8 , Power = 0.995
x_n= 5 , y_m= 3 , y_mean= 0.9 , Power = 0.999
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

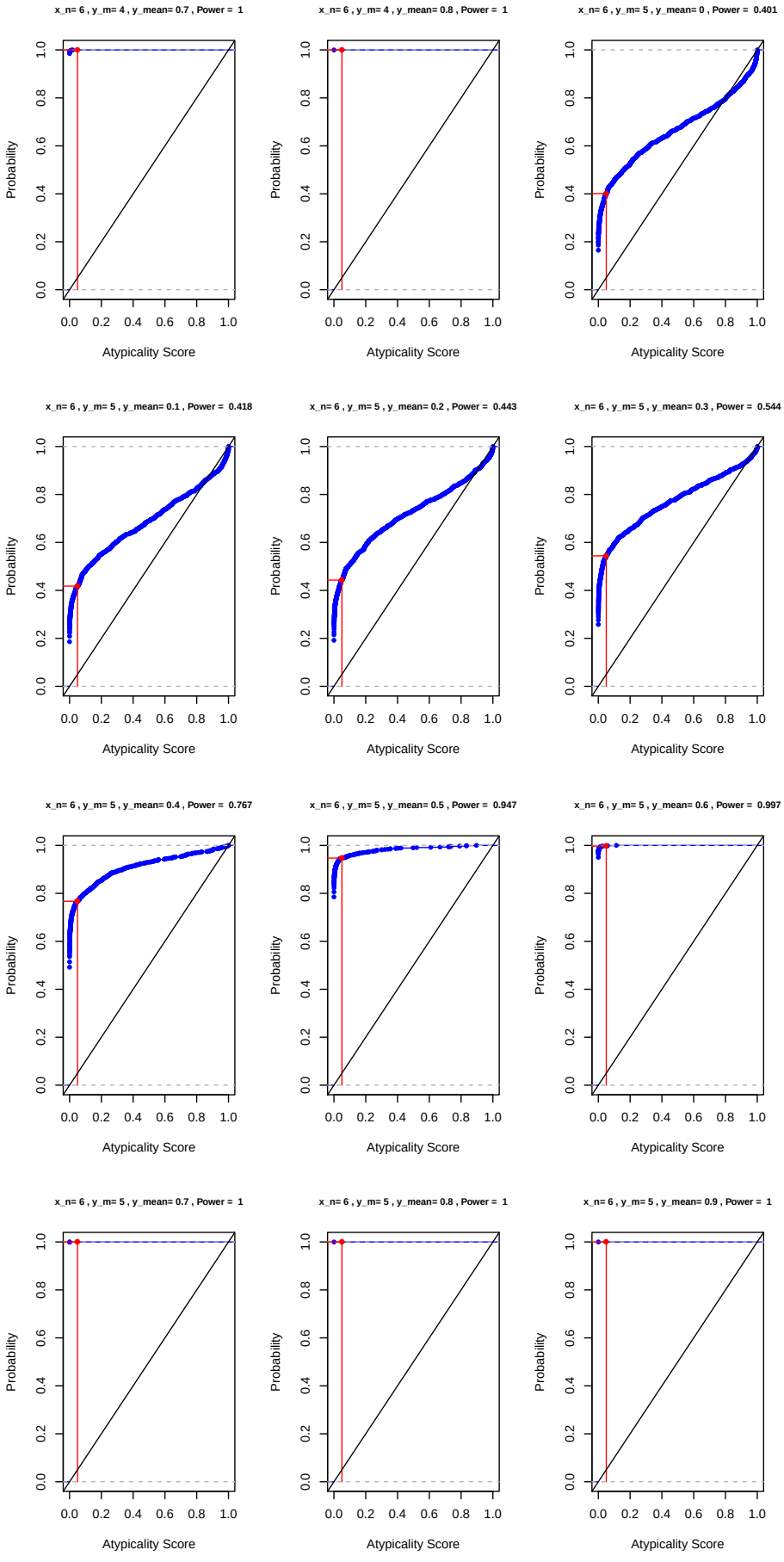

x_n= 5 , y_m= 3 , y_mean= 1 , Power = 1
x_n= 5 , y_m= 4 , y_mean= 0 , Power = 0.443
x_n= 5 , y_m= 4 , y_mean= 0.1 , Power = 0.453
x_n= 5 , y_m= 4 , y_mean= 0.2 , Power = 0.478
x_n= 5 , y_m= 4 , y_mean= 0.3 , Power = 0.547
x_n= 5 , y_m= 4 , y_mean= 0.4 , Power = 0.713
x_n= 5 , y_m= 4 , y_mean= 0.5 , Power = 0.842
x_n= 5 , y_m= 4 , y_mean= 0.6 , Power = 0.951
x_n= 5 , y_m= 4 , y_mean= 0.7 , Power = 0.987
x_n= 5 , y_m= 4 , y_mean= 0.8 , Power = 1
x_n= 5 , y_m= 4 , y_mean= 0.9 , Power = 1
x_n= 5 , y_m= 5 , y_mean= 0 , Power = 0.478
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

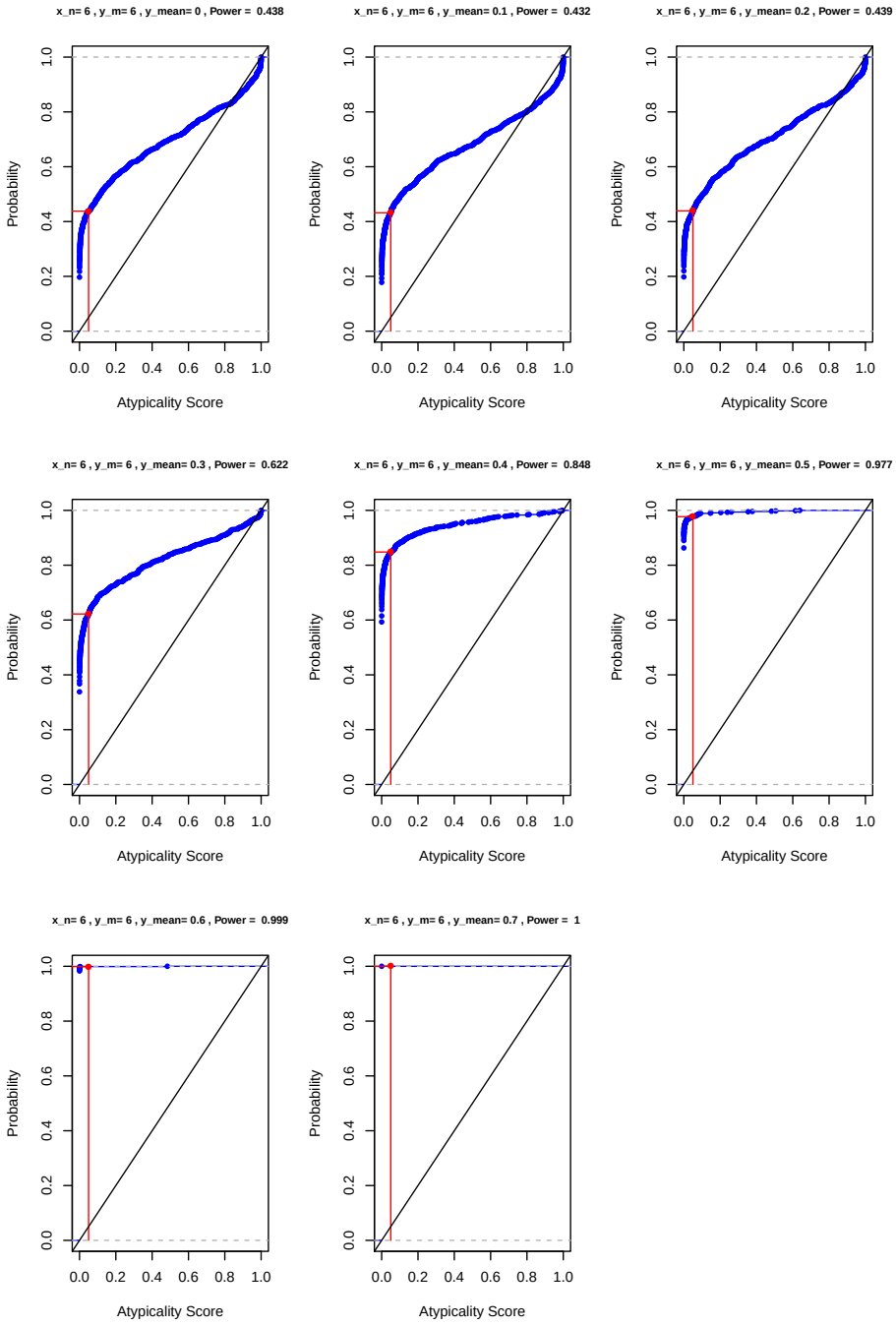

x_n= 5 , y_m= 5 , y_mean= 0.1 , Power = 0.502
x_n= 5 , y_m= 5 , y_mean= 0.2 , Power = 0.544
x_n= 5 , y_m= 5 , y_mean= 0.3 , Power = 0.619
x_n= 5 , y_m= 5 , y_mean= 0.4 , Power = 0.727
x_n= 5 , y_m= 5 , y_mean= 0.5 , Power = 0.911
x_n= 5 , y_m= 5 , y_mean= 0.6 , Power = 0.982
x_n= 6 , y_m= 1 , y_mean= 0 , Power = 0.202
x_n= 6 , y_m= 1 , y_mean= 0.1 , Power = 0.206
x_n= 6 , y_m= 1 , y_mean= 0.2 , Power = 0.184
x_n= 6 , y_m= 1 , y_mean= 0.3 , Power = 0.221
x_n= 6 , y_m= 1 , y_mean= 0.4 , Power = 0.301
x_n= 6 , y_m= 1 , y_mean= 0.5 , Power = 0.399
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

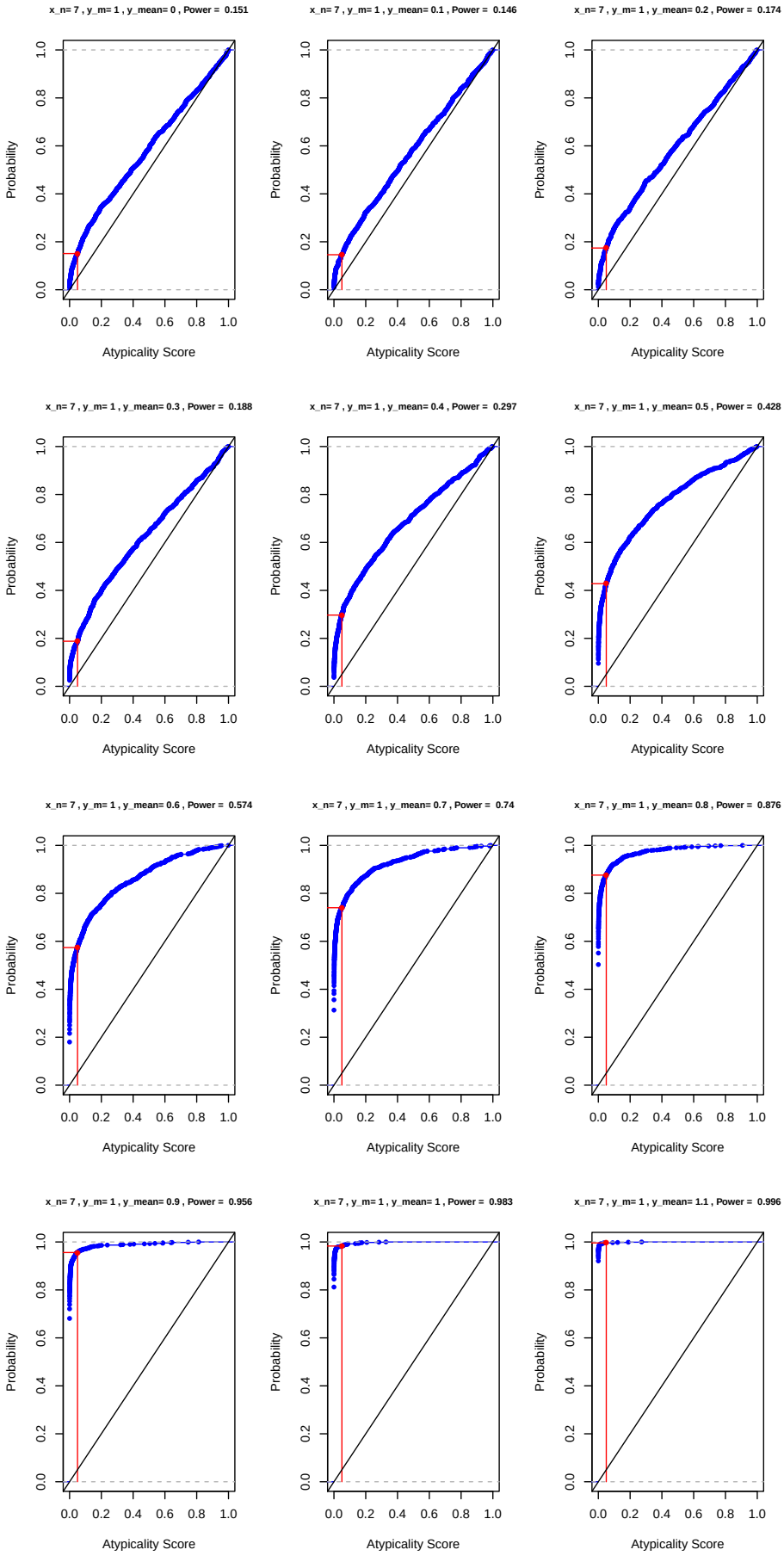

x_n= 6 , y_m= 1 , y_mean= 0.6 , Power = 0.518
x_n= 6 , y_m= 1 , y_mean= 0.7 , Power = 0.688
x_n= 6 , y_m= 1 , y_mean= 0.8 , Power = 0.814
x_n= 6 , y_m= 1 , y_mean= 0.9 , Power = 0.916
x_n= 6 , y_m= 1 , y_mean= 1 , Power = 0.97
x_n= 6 , y_m= 1 , y_mean= 1.1 , Power = 0.99
x_n= 6 , y_m= 1 , y_mean= 1.2 , Power = 0.998
x_n= 6 , y_m= 2 , y_mean= 0 , Power = 0.263
x_n= 6 , y_m= 2 , y_mean= 0.1 , Power = 0.271
x_n= 6 , y_m= 2 , y_mean= 0.2 , Power = 0.319
x_n= 6 , y_m= 2 , y_mean= 0.3 , Power = 0.349
x_n= 6 , y_m= 2 , y_mean= 0.4 , Power = 0.482
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

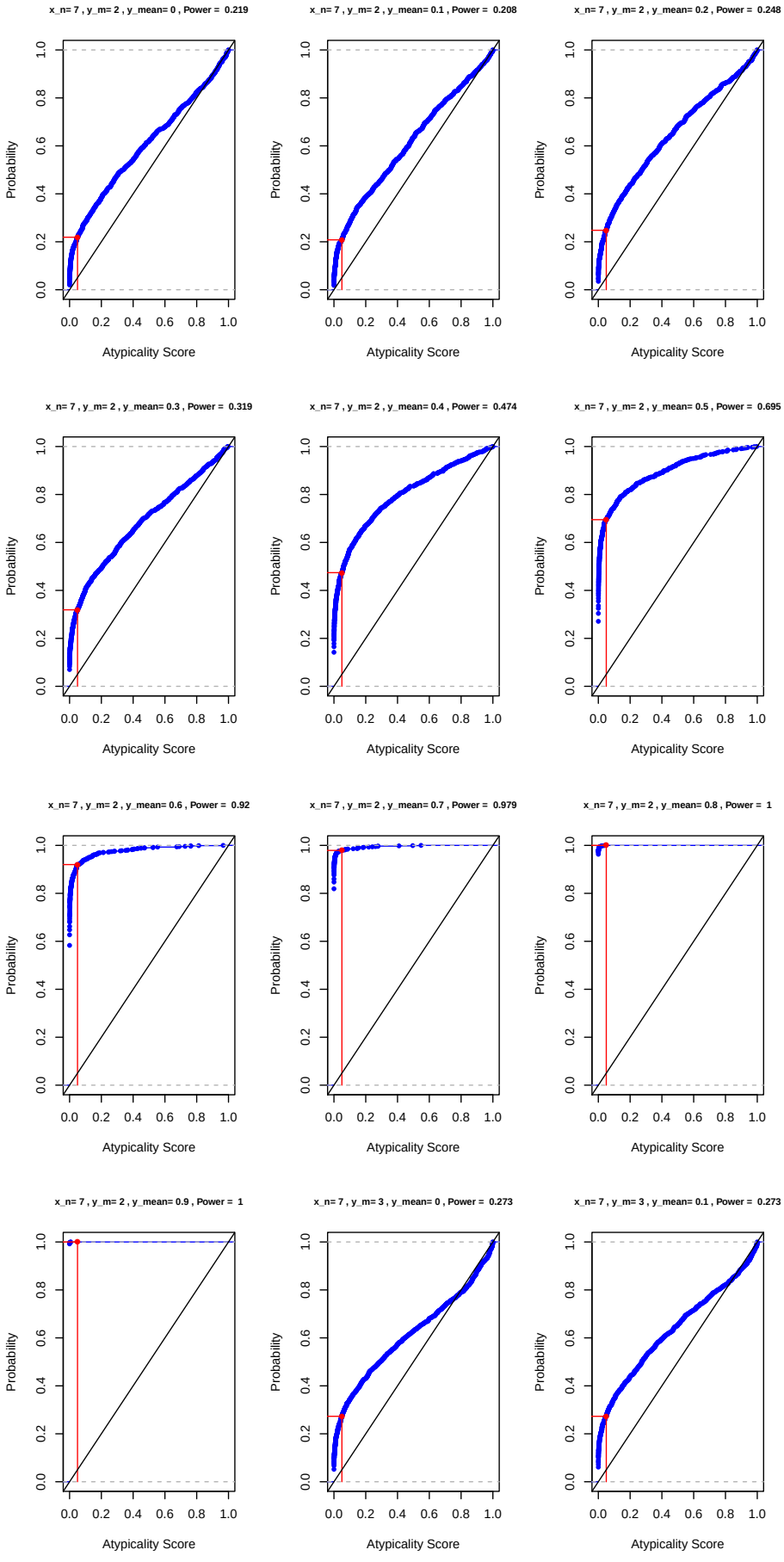

x_n= 6 , y_m= 2 , y_mean= 0.5 , Power = 0.668
x_n= 6 , y_m= 2 , y_mean= 0.6 , Power = 0.802
x_n= 6 , y_m= 2 , y_mean= 0.7 , Power = 0.953
x_n= 6 , y_m= 2 , y_mean= 0.8 , Power = 0.987
x_n= 6 , y_m= 2 , y_mean= 0.9 , Power = 1
x_n= 6 , y_m= 2 , y_mean= 1 , Power = 1
x_n= 6 , y_m= 3 , y_mean= 0 , Power = 0.315
x_n= 6 , y_m= 3 , y_mean= 0.1 , Power = 0.314
x_n= 6 , y_m= 3 , y_mean= 0.2 , Power = 0.35
x_n= 6 , y_m= 3 , y_mean= 0.3 , Power = 0.453
x_n= 6 , y_m= 3 , y_mean= 0.4 , Power = 0.619
x_n= 6 , y_m= 3 , y_mean= 0.5 , Power = 0.809
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

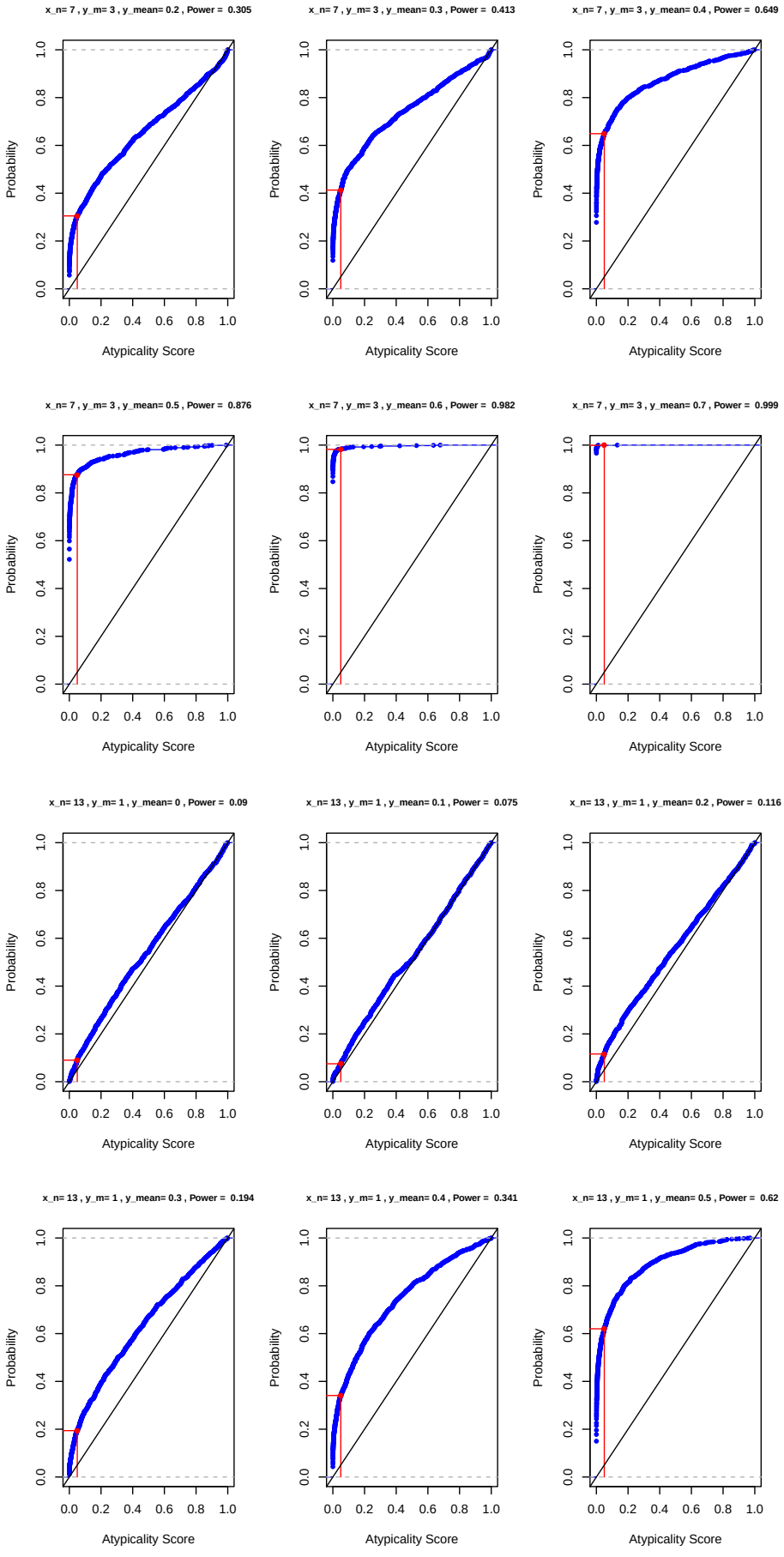

x_n= 6 , y_m= 3 , y_mean= 0.6 , Power = 0.956
x_n= 6 , y_m= 3 , y_mean= 0.7 , Power = 0.994
x_n= 6 , y_m= 3 , y_mean= 0.8 , Power = 1
x_n= 6 , y_m= 3 , y_mean= 0.9 , Power = 1
x_n= 6 , y_m= 3 , y_mean= 1 , Power = 1
x_n= 6 , y_m= 4 , y_mean= 0 , Power = 0.378
x_n= 6 , y_m= 4 , y_mean= 0.1 , Power = 0.379
x_n= 6 , y_m= 4 , y_mean= 0.2 , Power = 0.4
x_n= 6 , y_m= 4 , y_mean= 0.3 , Power = 0.517
x_n= 6 , y_m= 4 , y_mean= 0.4 , Power = 0.722
x_n= 6 , y_m= 4 , y_mean= 0.5 , Power = 0.913
x_n= 6 , y_m= 4 , y_mean= 0.6 , Power = 0.976
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

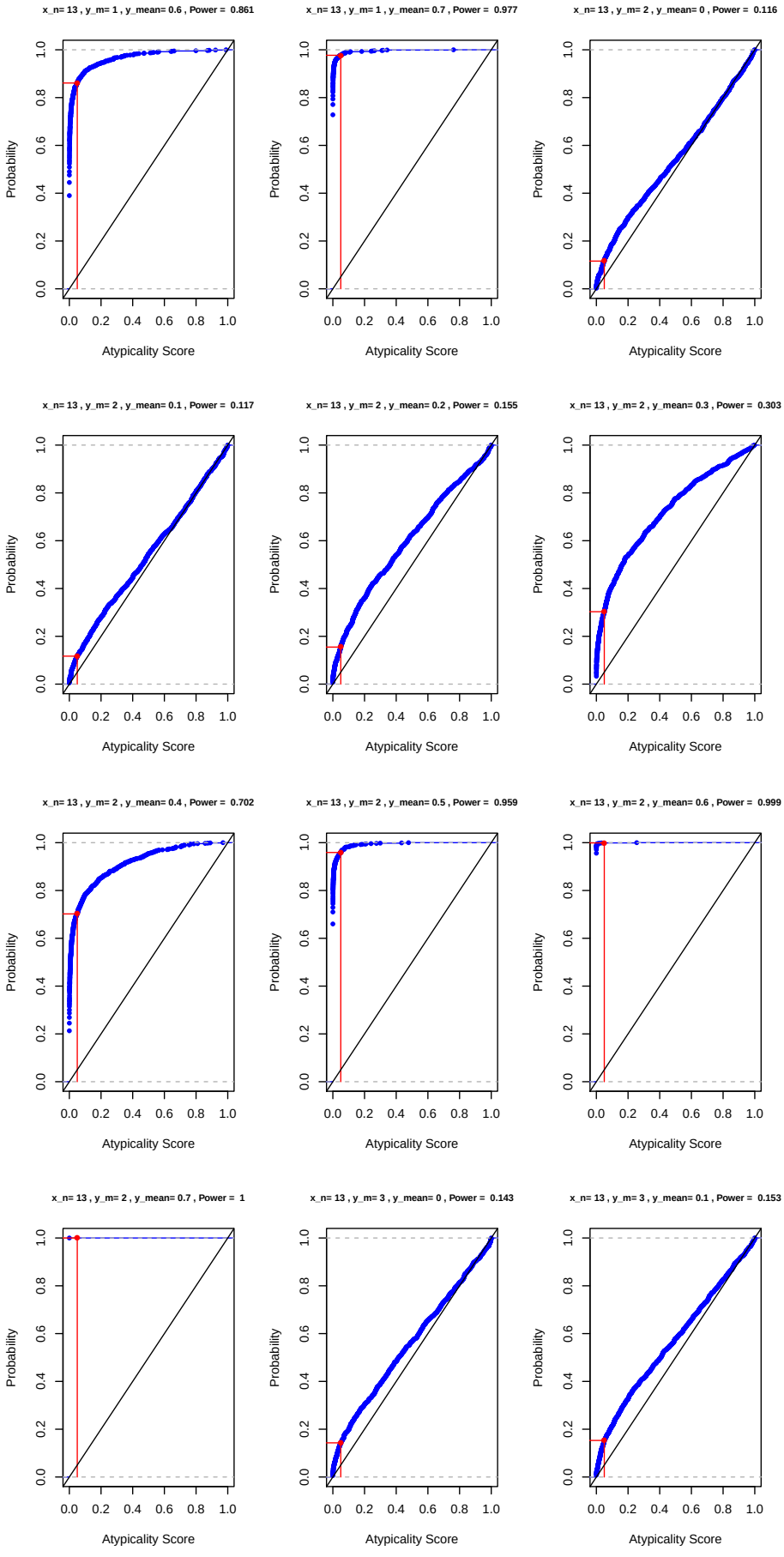

x_n= 6 , y_m= 4 , y_mean= 0.7 , Power = 1
x_n= 6 , y_m= 4 , y_mean= 0.8 , Power = 1
x_n= 6 , y_m= 5 , y_mean= 0 , Power = 0.401
x_n= 6 , y_m= 5 , y_mean= 0.1 , Power = 0.418
x_n= 6 , y_m= 5 , y_mean= 0.2 , Power = 0.443
x_n= 6 , y_m= 5 , y_mean= 0.3 , Power = 0.544
x_n= 6 , y_m= 5 , y_mean= 0.4 , Power = 0.767
x_n= 6 , y_m= 5 , y_mean= 0.5 , Power = 0.947
x_n= 6 , y_m= 5 , y_mean= 0.6 , Power = 0.997
x_n= 6 , y_m= 5 , y_mean= 0.7 , Power = 1
x_n= 6 , y_m= 5 , y_mean= 0.8 , Power = 1
x_n= 6 , y_m= 5 , y_mean= 0.9 , Power = 1
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

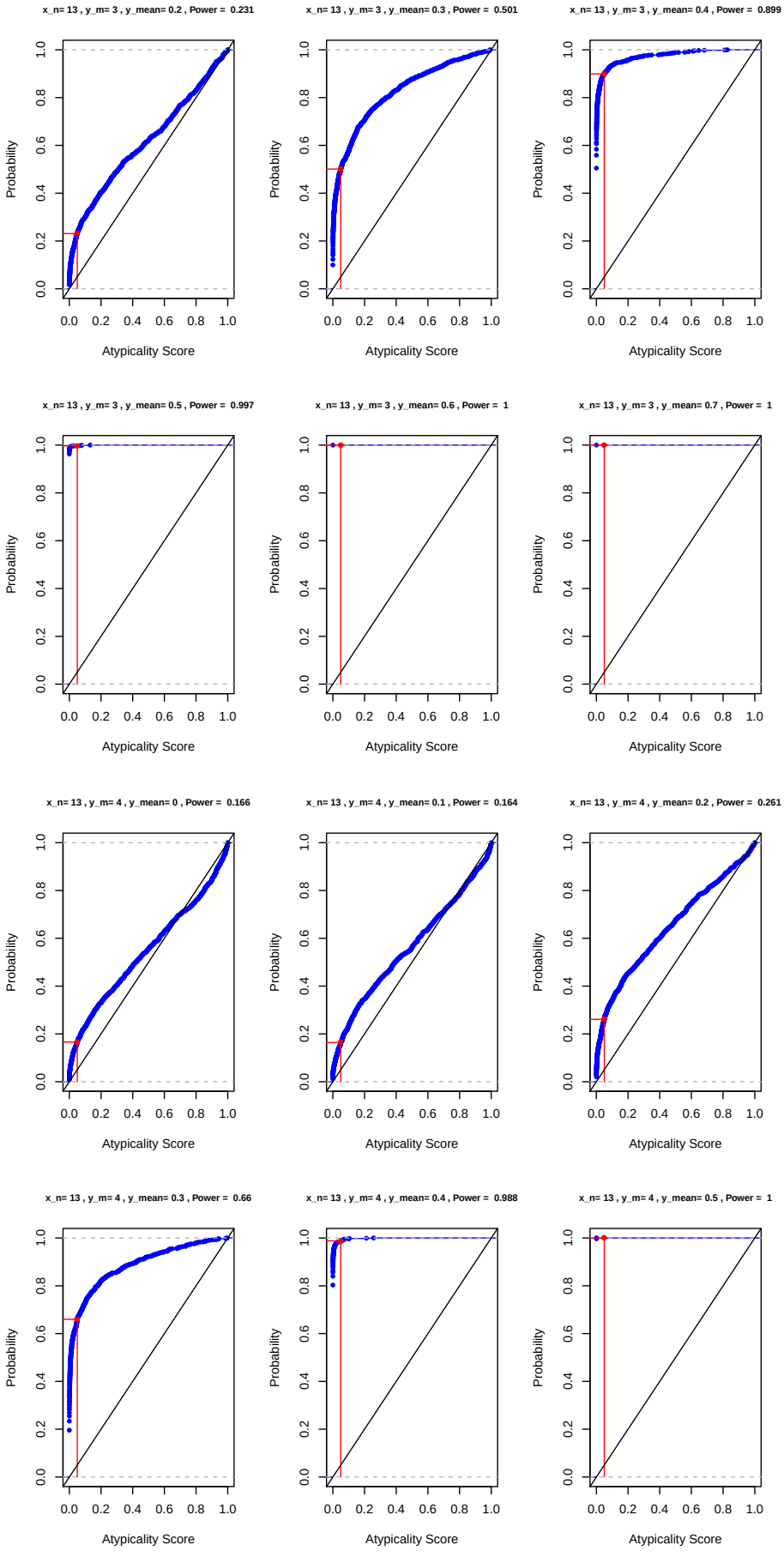

x_n= 6 , y_m= 6 , y_mean= 0 , Power = 0.438
x_n= 6 , y_m= 6 , y_mean= 0.1 , Power = 0.432
x_n= 6 , y_m= 6 , y_mean= 0.2 , Power = 0.439
x_n= 6 , y_m= 6 , y_mean= 0.3 , Power = 0.622
x_n= 6 , y_m= 6 , y_mean= 0.4 , Power = 0.848
x_n= 6 , y_m= 6 , y_mean= 0.5 , Power = 0.977
x_n= 6 , y_m= 6 , y_mean= 0.6 , Power = 0.999
x_n= 6 , y_m= 6 , y_mean= 0.7 , Power = 1
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

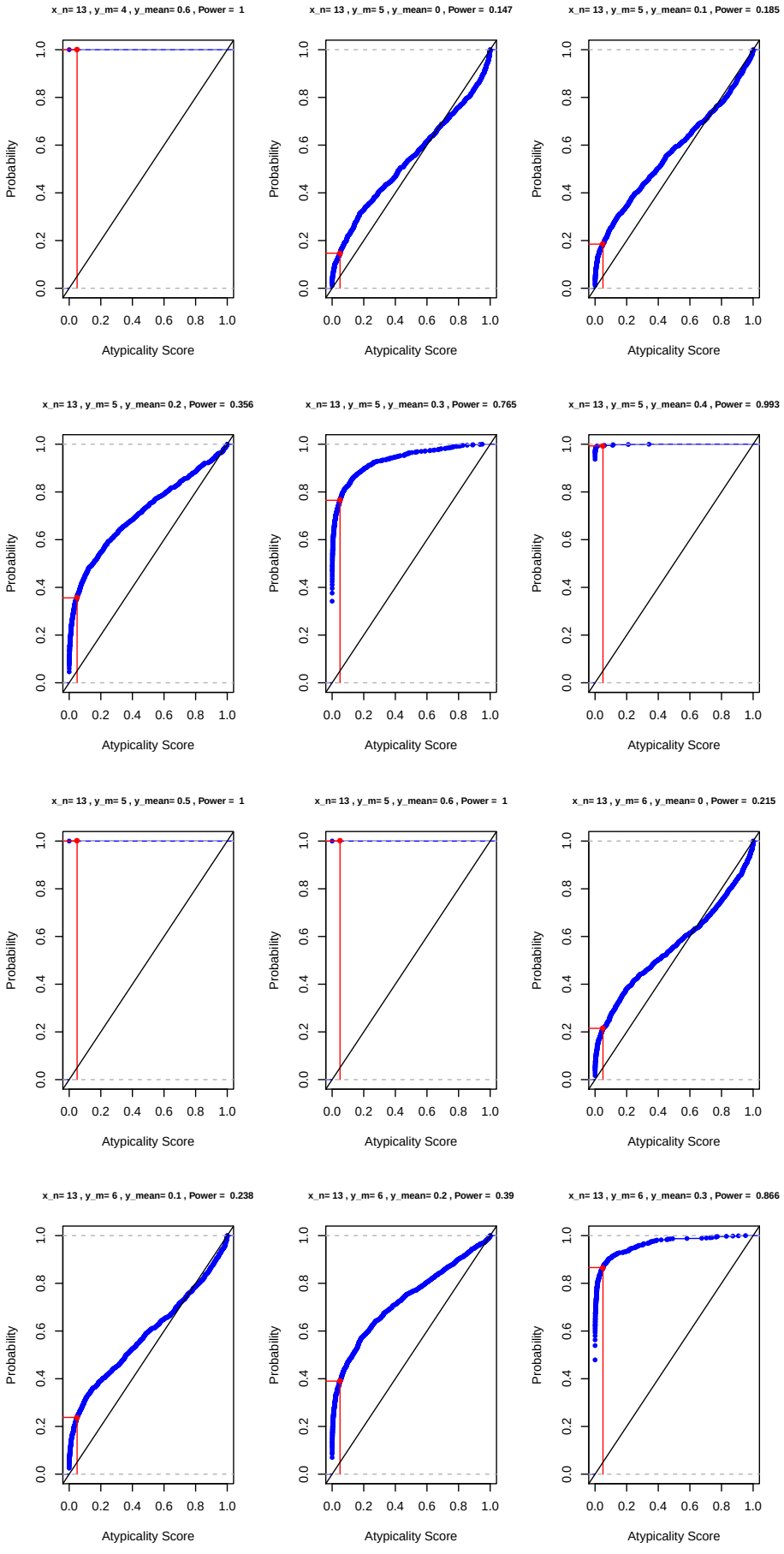

x_n= 7 , y_m= 1 , y_mean= 0 , Power = 0.151
x_n= 7 , y_m= 1 , y_mean= 0.1 , Power = 0.146
x_n= 7 , y_m= 1 , y_mean= 0.2 , Power = 0.174
x_n= 7 , y_m= 1 , y_mean= 0.3 , Power = 0.188
x_n= 7 , y_m= 1 , y_mean= 0.4 , Power = 0.297
x_n= 7 , y_m= 1 , y_mean= 0.5 , Power = 0.428
x_n= 7 , y_m= 1 , y_mean= 0.6 , Power = 0.574
x_n= 7 , y_m= 1 , y_mean= 0.7 , Power = 0.74
x_n= 7 , y_m= 1 , y_mean= 0.8 , Power = 0.876
x_n= 7 , y_m= 1 , y_mean= 0.9 , Power = 0.956
x_n= 7 , y_m= 1 , y_mean= 1 , Power = 0.983
x_n= 7 , y_m= 1 , y_mean= 1.1 , Power = 0.996
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

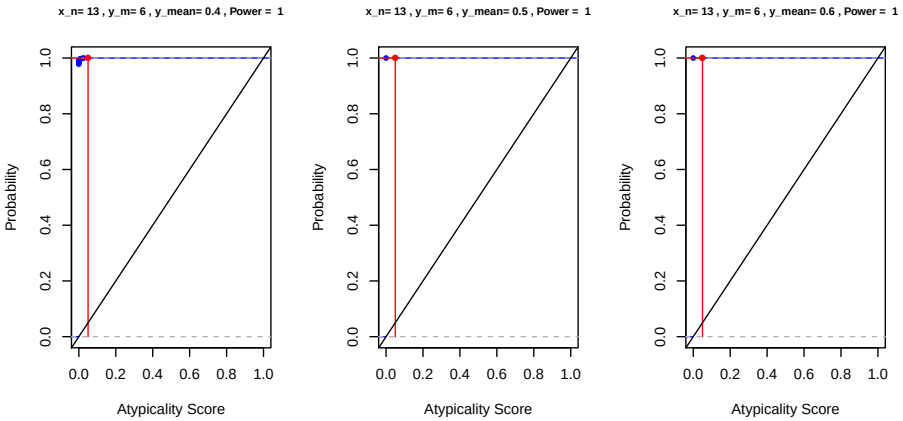

x_n= 7 , y_m= 2 , y_mean= 0 , Power = 0.219
x_n= 7 , y_m= 2 , y_mean= 0.1 , Power = 0.208
x_n= 7 , y_m= 2 , y_mean= 0.2 , Power = 0.248
x_n= 7 , y_m= 2 , y_mean= 0.3 , Power = 0.319
x_n= 7 , y_m= 2 , y_mean= 0.4 , Power = 0.474
x_n= 7 , y_m= 2 , y_mean= 0.5 , Power = 0.695
x_n= 7 , y_m= 2 , y_mean= 0.6 , Power = 0.92
x_n= 7 , y_m= 2 , y_mean= 0.7 , Power = 0.979
x_n= 7 , y_m= 2 , y_mean= 0.8 , Power = 1
x_n= 7 , y_m= 2 , y_mean= 0.9 , Power = 1
x_n= 7 , y_m= 3 , y_mean= 0 , Power = 0.273
x_n= 7 , y_m= 3 , y_mean= 0.1 , Power = 0.273
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

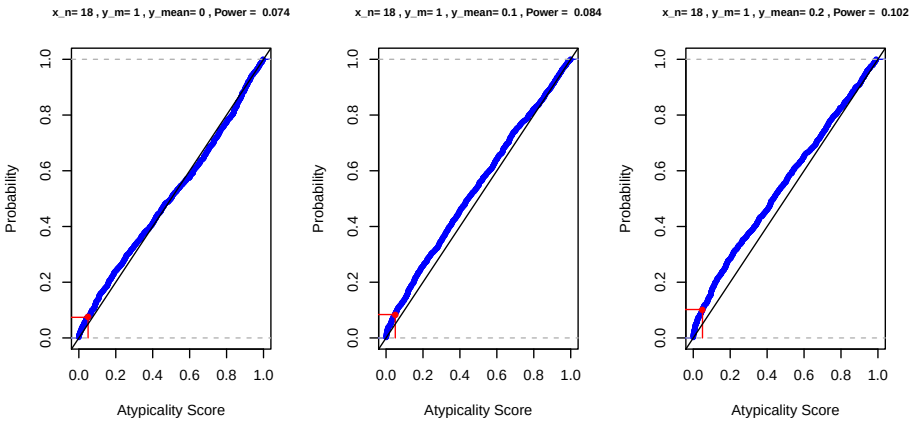

x_n= 7 , y_m= 3 , y_mean= 0.2 , Power = 0.305
x_n= 7 , y_m= 3 , y_mean= 0.3 , Power = 0.413
x_n= 7 , y_m= 3 , y_mean= 0.4 , Power = 0.649
x_n= 7 , y_m= 3 , y_mean= 0.5 , Power = 0.876
x_n= 7 , y_m= 3 , y_mean= 0.6 , Power = 0.982
x_n= 7 , y_m= 3 , y_mean= 0.7 , Power = 0.999
x_n= 13 , y_m= 1 , y_mean= 0 , Power = 0.09
x_n= 13 , y_m= 1 , y_mean= 0.1 , Power = 0.075
x_n= 13 , y_m= 1 , y_mean= 0.2 , Power = 0.116
x_n= 13 , y_m= 1 , y_mean= 0.3 , Power = 0.194
x_n= 13 , y_m= 1 , y_mean= 0.4 , Power = 0.341
x_n= 13 , y_m= 1 , y_mean= 0.5 , Power = 0.62
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

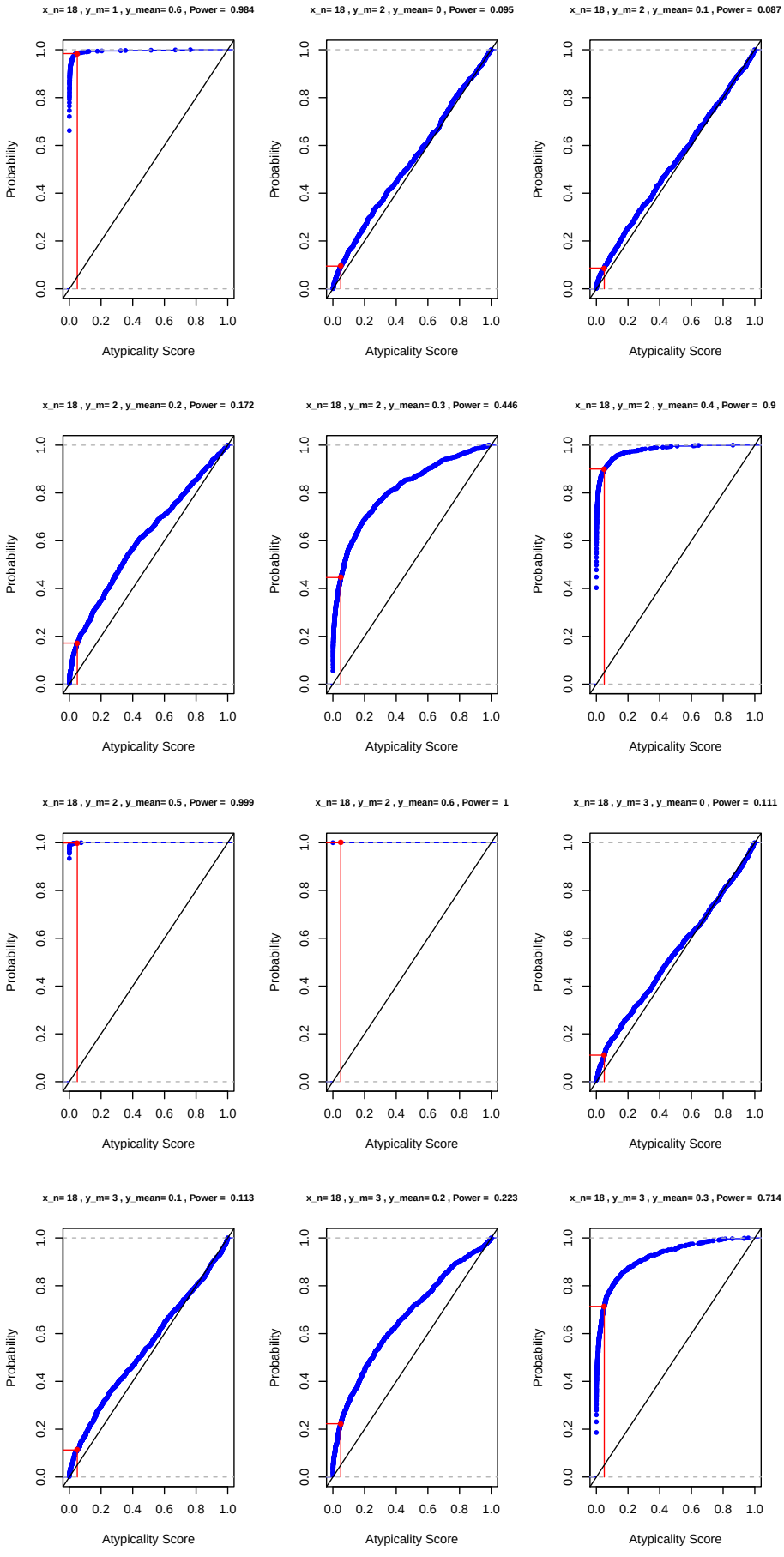

x_n= 13 , y_m= 1 , y_mean= 0.6 , Power = 0.861
x_n= 13 , y_m= 1 , y_mean= 0.7 , Power = 0.977
x_n= 13 , y_m= 2 , y_mean= 0 , Power = 0.116
x_n= 13 , y_m= 2 , y_mean= 0.1 , Power = 0.117
x_n= 13 , y_m= 2 , y_mean= 0.2 , Power = 0.155
x_n= 13 , y_m= 2 , y_mean= 0.3 , Power = 0.303
x_n= 13 , y_m= 2 , y_mean= 0.4 , Power = 0.702
x_n= 13 , y_m= 2 , y_mean= 0.5 , Power = 0.959
x_n= 13 , y_m= 2 , y_mean= 0.6 , Power = 0.999
x_n= 13 , y_m= 2 , y_mean= 0.7 , Power = 1
x_n= 13 , y_m= 3 , y_mean= 0 , Power = 0.143
x_n= 13 , y_m= 3 , y_mean= 0.1 , Power = 0.153
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

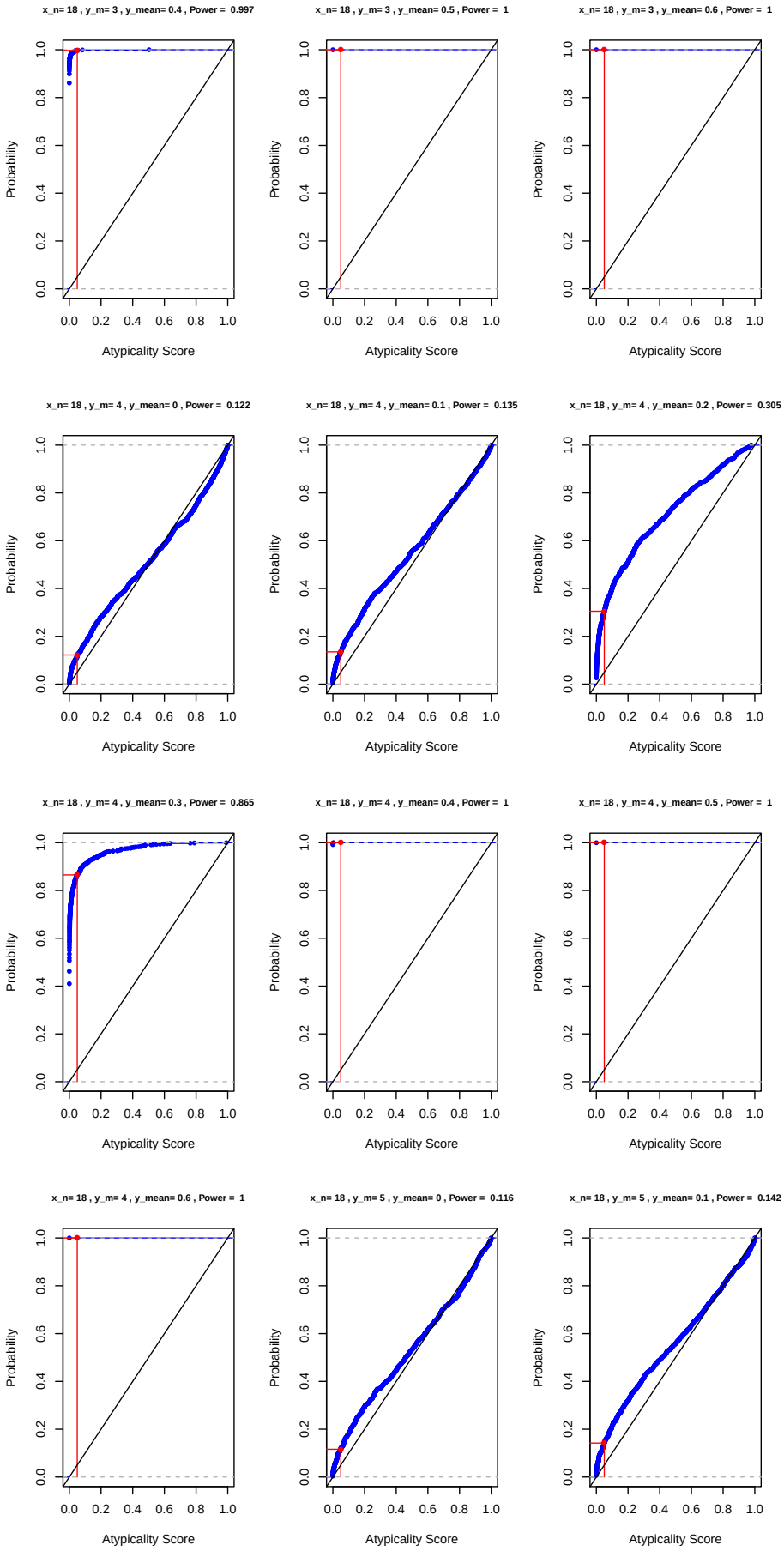

x_n= 13 , y_m= 3 , y_mean= 0.2 , Power = 0.231
x_n= 13 , y_m= 3 , y_mean= 0.3 , Power = 0.501
x_n= 13 , y_m= 3 , y_mean= 0.4 , Power = 0.899
x_n= 13 , y_m= 3 , y_mean= 0.5 , Power = 0.997
x_n= 13 , y_m= 3 , y_mean= 0.6 , Power = 1
x_n= 13 , y_m= 3 , y_mean= 0.7 , Power = 1
x_n= 13 , y_m= 4 , y_mean= 0 , Power = 0.166
x_n= 13 , y_m= 4 , y_mean= 0.1 , Power = 0.164
x_n= 13 , y_m= 4 , y_mean= 0.2 , Power = 0.261
x_n= 13 , y_m= 4 , y_mean= 0.3 , Power = 0.66
x_n= 13 , y_m= 4 , y_mean= 0.4 , Power = 0.988
x_n= 13 , y_m= 4 , y_mean= 0.5 , Power = 1
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

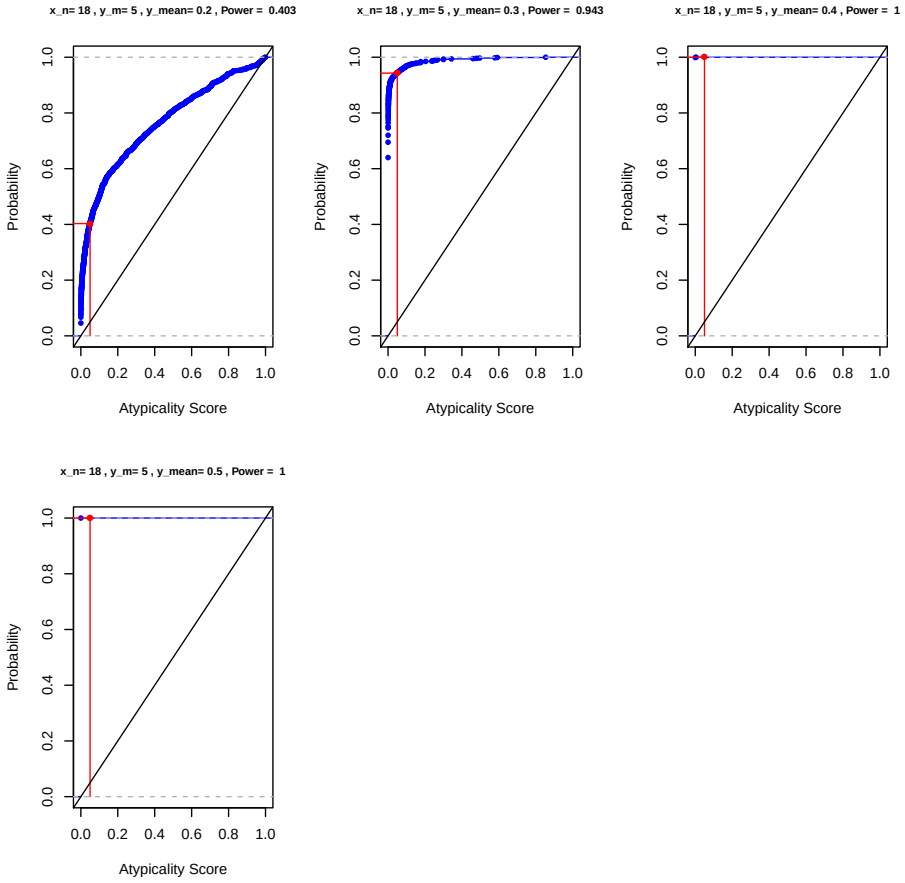

x_n= 13 , y_m= 4 , y_mean= 0.6 , Power = 1
x_n= 13 , y_m= 5 , y_mean= 0 , Power = 0.147
x_n= 13 , y_m= 5 , y_mean= 0.1 , Power = 0.185
x_n= 13 , y_m= 5 , y_mean= 0.2 , Power = 0.356
x_n= 13 , y_m= 5 , y_mean= 0.3 , Power = 0.765
x_n= 13 , y_m= 5 , y_mean= 0.4 , Power = 0.993
x_n= 13 , y_m= 5 , y_mean= 0.5 , Power = 1
x_n= 13 , y_m= 5 , y_mean= 0.6 , Power = 1
x_n= 13 , y_m= 6 , y_mean= 0 , Power = 0.215
x_n= 13 , y_m= 6 , y_mean= 0.1 , Power = 0.238
x_n= 13 , y_m= 6 , y_mean= 0.2 , Power = 0.39
x_n= 13 , y_m= 6 , y_mean= 0.3 , Power = 0.866
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0


x_n= 13 , y_m= 6 , y_mean= 0.4 , Power = 1
x_n= 13 , y_m= 6 , y_mean= 0.5 , Power = 1
x_n= 13 , y_m= 6 , y_mean= 0.6 , Power = 1
Probability
Atypicality Score
0.0 0.2 0.4 0.6 0.8 1.0


x_n= 18 , y_m= 1 , y_mean= 0 , Power = 0.074
x_n= 18 , y_m= 1 , y_mean= 0.1 , Power = 0.084
x_n= 18 , y_m= 1 , y_mean= 0.2 , Power = 0.102
x_n= 18 , y_m= 1 , y_mean= 0.3 , Power = 0.196
x_n= 18 , y_m= 1 , y_mean= 0.4 , Power = 0.5
x_n= 18 , y_m= 1 , y_mean= 0.5 , Power = 0.802
Probability
Atypicality Score
0.0 0.2 0.4 0.6 0.8 1.0


x_n= 18 , y_m= 1 , y_mean= 0.6 , Power = 0.984
x_n= 18 , y_m= 2 , y_mean= 0 , Power = 0.095
x_n= 18 , y_m= 2 , y_mean= 0.1 , Power = 0.087
x_n= 18 , y_m= 2 , y_mean= 0.2 , Power = 0.172
x_n= 18 , y_m= 2 , y_mean= 0.3 , Power = 0.446
x_n= 18 , y_m= 2 , y_mean= 0.4 , Power = 0.9
x_n= 18 , y_m= 2 , y_mean= 0.5 , Power = 0.999
x_n= 18 , y_m= 2 , y_mean= 0.6 , Power = 1
x_n= 18 , y_m= 3 , y_mean= 0 , Power = 0.111
x_n= 18 , y_m= 3 , y_mean= 0.1 , Power = 0.113
x_n= 18 , y_m= 3 , y_mean= 0.2 , Power = 0.223
x_n= 18 , y_m= 3 , y_mean= 0.3 , Power = 0.714
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0


x_n= 18 , y_m= 3 , y_mean= 0.4 , Power = 0.997
x_n= 18 , y_m= 3 , y_mean= 0.5 , Power = 1
x_n= 18 , y_m= 3 , y_mean= 0.6 , Power = 1
x_n= 18 , y_m= 4 , y_mean= 0 , Power = 0.122
x_n= 18 , y_m= 4 , y_mean= 0.1 , Power = 0.135
x_n= 18 , y_m= 4 , y_mean= 0.2 , Power = 0.305
x_n= 18 , y_m= 4 , y_mean= 0.3 , Power = 0.865
x_n= 18 , y_m= 4 , y_mean= 0.4 , Power = 1
x_n= 18 , y_m= 4 , y_mean= 0.5 , Power = 1
x_n= 18 , y_m= 4 , y_mean= 0.6 , Power = 1
x_n= 18 , y_m= 5 , y_mean= 0 , Power = 0.116
x_n= 18 , y_m= 5 , y_mean= 0.1 , Power = 0.142
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0


x_n= 18 , y_m= 5 , y_mean= 0.2 , Power = 0.403
x_n= 18 , y_m= 5 , y_mean= 0.3 , Power = 0.943
x_n= 18 , y_m= 5 , y_mean= 0.4 , Power = 1
x_n= 18 , y_m= 5 , y_mean= 0.5 , Power = 1
Probability
Atypicality Score
0.0
0.2
0.4
0.6
0.8
1.0

C.3 Power Function Plots

C.4 SVM finding best ν parameter

# of training obs	# of dimensions	ν parameter	% SVM positive classification
4	1	0.05	45
4	2	0.05	47
4	4	0.05	45
4	8	0.05	30
4	16	0.05	15
4	32	0.05	8
4	64	0.05	2
4	1	0.10	35
4	2	0.10	51
4	4	0.10	50
4	8	0.10	33
4	16	0.10	18
4	32	0.10	8
4	64	0.10	2
4	1	0.15	39
4	2	0.15	45
4	4	0.15	46
4	8	0.15	35
4	16	0.15	12
4	32	0.15	5
4	64	0.15	2
4	1	0.20	33
4	2	0.20	52

4	4	0.20	44
4	8	0.20	42
4	16	0.20	14
4	32	0.20	5
4	64	0.20	2
4	1	0.25	41
4	2	0.25	52
4	4	0.25	51
4	8	0.25	38
4	16	0.25	30
4	32	0.25	1
4	64	0.25	0
4	1	0.30	48
4	2	0.30	52
4	4	0.30	44
4	8	0.30	29
4	16	0.30	13
4	32	0.30	6
4	64	0.30	1
4	1	0.35	48
4	2	0.35	46
4	4	0.35	45
4	8	0.35	40
4	16	0.35	20
4	32	0.35	8
4	64	0.35	1
4	1	0.40	42

4	2	0.40	47
4	4	0.40	50
4	8	0.40	30
4	16	0.40	23
4	32	0.40	8
4	64	0.40	1
4	1	0.45	47
4	2	0.45	45
4	4	0.45	46
4	8	0.45	34
4	16	0.45	19
4	32	0.45	5
4	64	0.45	2
4	1	0.50	37
4	2	0.50	48
4	4	0.50	50
4	8	0.50	31
4	16	0.50	17
4	32	0.50	6
4	64	0.50	0
4	1	0.55	35
4	2	0.55	46
4	4	0.55	49
4	8	0.55	32
4	16	0.55	24
4	32	0.55	6
4	64	0.55	0

4	1	0.60	36
4	2	0.60	48
4	4	0.60	52
4	8	0.60	36
4	16	0.60	18
4	32	0.60	8
4	64	0.60	2
4	1	0.65	38
4	2	0.65	48
4	4	0.65	57
4	8	0.65	34
4	16	0.65	12
4	32	0.65	8
4	64	0.65	0
4	1	0.70	40
4	2	0.70	56
4	4	0.70	47
4	8	0.70	31
4	16	0.70	21
4	32	0.70	3
4	64	0.70	1
4	1	0.75	24
4	2	0.75	48
4	4	0.75	48
4	8	0.75	21
4	16	0.75	20
4	32	0.75	5

4	64	0.75	1
4	1	0.80	22
4	2	0.80	42
4	4	0.80	42
4	8	0.80	36
4	16	0.80	21
4	32	0.80	5
4	64	0.80	1
4	1	0.85	23
4	2	0.85	35
4	4	0.85	38
4	8	0.85	38
4	16	0.85	19
4	32	0.85	2
4	64	0.85	1
4	1	0.90	15
4	2	0.90	25
4	4	0.90	42
4	8	0.90	24
4	16	0.90	19
4	32	0.90	6
4	64	0.90	0
4	1	0.95	14
4	2	0.95	21
4	4	0.95	40
4	8	0.95	25
4	16	0.95	19

4	32	0.95	4
4	64	0.95	1
10	1	0.05	47
10	2	0.05	77
10	4	0.05	98
10	8	0.05	100
10	16	0.05	100
10	32	0.05	100
10	1	0.10	59
10	2	0.10	81
10	4	0.10	99
10	8	0.10	100
10	16	0.10	100
10	32	0.10	100
10	1	0.15	54
10	2	0.15	80
10	4	0.15	98
10	8	0.15	100
10	16	0.15	100
10	32	0.15	100
10	1	0.20	59
10	2	0.20	76
10	4	0.20	100
10	8	0.20	100
10	16	0.20	100
10	32	0.20	100
10	1	0.25	53

10	2	0.25	77
10	4	0.25	97
10	8	0.25	100
10	16	0.25	100
10	32	0.25	100
10	1	0.30	62
10	2	0.30	80
10	4	0.30	99
10	8	0.30	100
10	16	0.30	100
10	32	0.30	100
10	1	0.35	58
10	2	0.35	82
10	4	0.35	100
10	8	0.35	100
10	16	0.35	100
10	32	0.35	100
10	1	0.40	53
10	2	0.40	83
10	4	0.40	99
10	8	0.40	100
10	16	0.40	100
10	32	0.40	100
10	1	0.45	49
10	2	0.45	80
10	4	0.45	98
10	8	0.45	100

10	16	0.45	100
10	32	0.45	100
10	1	0.50	49
10	2	0.50	84
10	4	0.50	97
10	8	0.50	100
10	16	0.50	100
10	32	0.50	100
10	1	0.55	48
10	2	0.55	85
10	4	0.55	99
10	8	0.55	99
10	16	0.55	100
10	32	0.55	100
10	1	0.60	47
10	2	0.60	84
10	4	0.60	98
10	8	0.60	100
10	16	0.60	100
10	32	0.60	100
10	1	0.65	42
10	2	0.65	83
10	4	0.65	98
10	8	0.65	100
10	16	0.65	100
10	32	0.65	100
10	1	0.70	44

10	2	0.70	72
10	4	0.70	100
10	8	0.70	100
10	16	0.70	100
10	32	0.70	100
10	1	0.75	32
10	2	0.75	72
10	4	0.75	91
10	8	0.75	100
10	16	0.75	100
10	32	0.75	100
10	1	0.80	34
10	2	0.80	76
10	4	0.80	91
10	8	0.80	97
10	16	0.80	100
10	32	0.80	100
10	1	0.85	33
10	2	0.85	55
10	4	0.85	85
10	8	0.85	99
10	16	0.85	100
10	32	0.85	100
10	1	0.90	26
10	2	0.90	36
10	4	0.90	77
10	8	0.90	93

10	16	0.90	100
10	32	0.90	100
10	1	0.95	12
10	2	0.95	41
10	4	0.95	76
10	8	0.95	92
10	16	0.95	99
10	32	0.95	100
20	1	0.05	75
20	2	0.05	74
20	4	0.05	100
20	8	0.05	100
20	16	0.05	100
20	32	0.05	100
20	1	0.10	76
20	2	0.10	68
20	4	0.10	100
20	8	0.10	100
20	16	0.10	100
20	32	0.10	100
20	1	0.15	81
20	2	0.15	66
20	4	0.15	100
20	8	0.15	100
20	16	0.15	100
20	32	0.15	100
20	1	0.20	65

20	2	0.20	63
20	4	0.20	100
20	8	0.20	100
20	16	0.20	100
20	32	0.20	100
20	1	0.25	80
20	2	0.25	69
20	4	0.25	100
20	8	0.25	100
20	16	0.25	100
20	32	0.25	100
20	1	0.30	71
20	2	0.30	73
20	4	0.30	100
20	8	0.30	100
20	16	0.30	100
20	32	0.30	100
20	1	0.35	65
20	2	0.35	70
20	4	0.35	100
20	8	0.35	100
20	16	0.35	100
20	32	0.35	100
20	1	0.40	51
20	2	0.40	85
20	4	0.40	100
20	8	0.40	100

20	16	0.40	100
20	32	0.40	100
20	1	0.45	41
20	2	0.45	88
20	4	0.45	100
20	8	0.45	100
20	16	0.45	100
20	32	0.45	100
20	1	0.50	56
20	2	0.50	91
20	4	0.50	100
20	8	0.50	100
20	16	0.50	100
20	32	0.50	100
20	1	0.55	49
20	2	0.55	94
20	4	0.55	100
20	8	0.55	100
20	16	0.55	100
20	32	0.55	100
20	1	0.60	43
20	2	0.60	98
20	4	0.60	100
20	8	0.60	100
20	16	0.60	100
20	32	0.60	100
20	1	0.65	32

20	2	0.65	97
20	4	0.65	100
20	8	0.65	100
20	16	0.65	100
20	32	0.65	100
20	1	0.70	54
20	2	0.70	95
20	4	0.70	100
20	8	0.70	100
20	16	0.70	100
20	32	0.70	100
20	1	0.75	41
20	2	0.75	94
20	4	0.75	99
20	8	0.75	100
20	16	0.75	100
20	32	0.75	100
20	1	0.80	40
20	2	0.80	91
20	4	0.80	100
20	8	0.80	100
20	16	0.80	100
20	32	0.80	100
20	1	0.85	42
20	2	0.85	85
20	4	0.85	100
20	8	0.85	100

20	16	0.85	100
20	32	0.85	100
20	1	0.90	33
20	2	0.90	64
20	4	0.90	96
20	8	0.90	100
20	16	0.90	100
20	32	0.90	100
20	1	0.95	19
20	2	0.95	40
20	4	0.95	86
20	8	0.95	100
20	16	0.95	100
20	32	0.95	100

Table 7: The percent of test objects classified as belonging to the training objects set, given fluctuating numbers of training objects, number of dimensions, and ν parameter.

C.5 SVM and Atypicality Results when the NULL hypothesis is false

# of training obs	# of dimensions	Test obj vector	% Atyp	% SVM
4	1	0.0	78	33
4	1	0.5	72	42
4	1	1.0	55	22
4	1	1.5	41	9
4	1	2.0	37	5
4	1	2.5	28	0

4	1	3.0	21	0
4	1	3.5	16	0
4	1	4.0	12	0
4	1	4.5	9	0
4	1	5.0	5	0
4	1	5.5	3	0
4	1	6.0	4	0
4	1	6.5	1	0
4	1	7.0	1	0
4	2	0.0	88	53
4	2	0.5	81	33
4	2	1.0	60	13
4	2	1.5	52	1
4	2	2.0	41	0
4	2	2.5	22	0
4	2	3.0	18	0
4	2	3.5	8	0
4	2	4.0	4	0
4	2	4.5	2	0
4	2	5.0	1	0
4	2	5.5	0	0
4	4	0.0	74	48
4	4	0.5	76	21
4	4	1.0	74	2
4	4	1.5	45	0
4	4	2.0	28	0
4	4	2.5	8	0

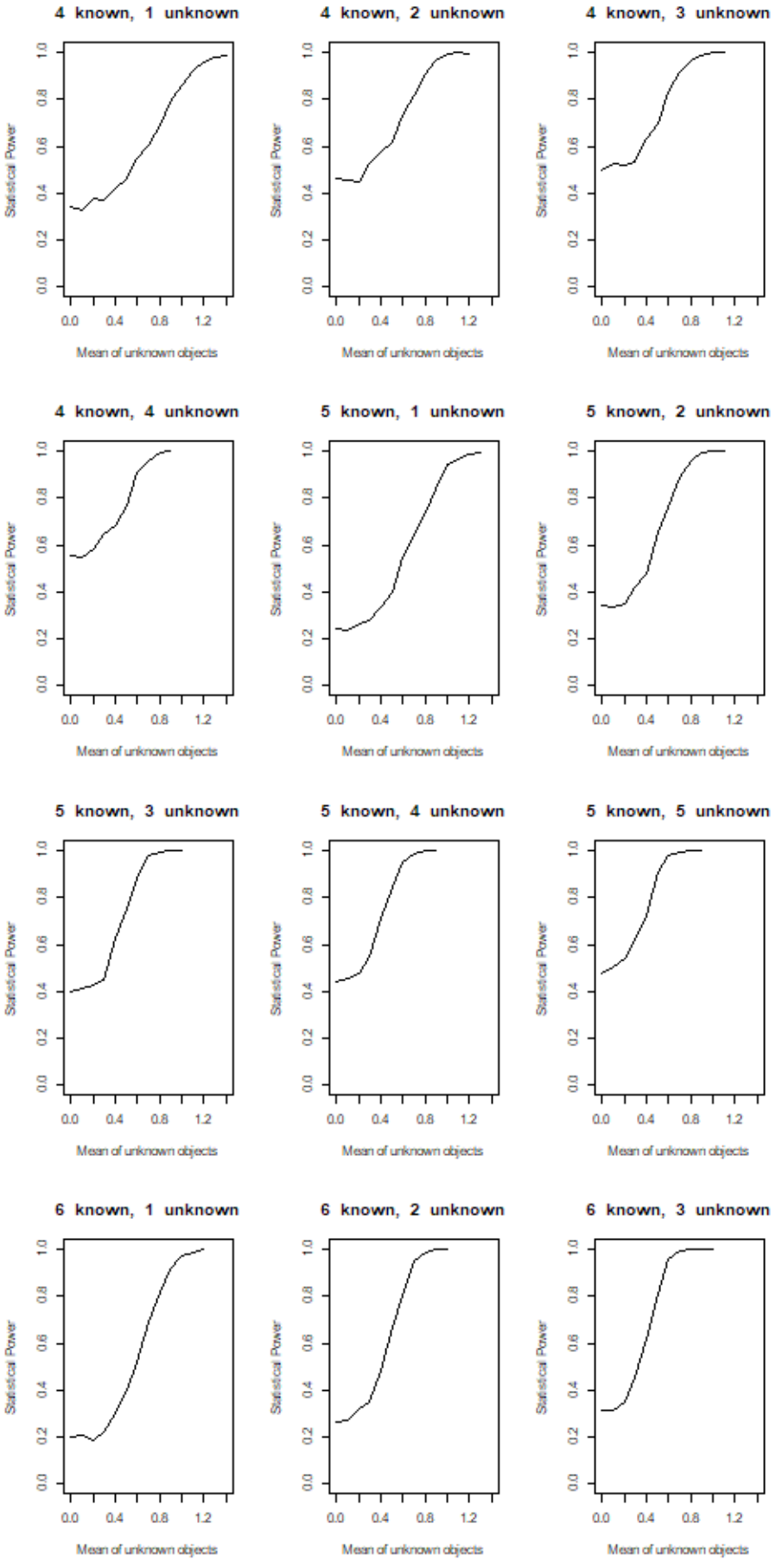
4	4	3.0	7	0
4	4	3.5	3	0
4	4	4.0	2	0
4	8	0.0	73	34
4	8	0.5	68	6
4	8	1.0	64	0
4	8	1.5	37	0
4	8	2.0	17	0
4	8	2.5	9	0
4	8	3.0	2	0
4	16	0.0	48	15
4	16	0.5	56	1
4	16	1.0	67	0
4	16	1.5	34	0
4	16	2.0	8	0
4	16	2.5	3	0
4	32	0.0	32	11
4	32	0.5	31	0
4	32	1.0	74	0
4	32	1.5	15	0
4	32	2.0	0	0
4	64	0.0	15	0
4	64	0.5	19	0
4	64	1.0	71	0
4	64	1.5	6	0
4	64	2.0	0	0
10	1	0.0	99	63

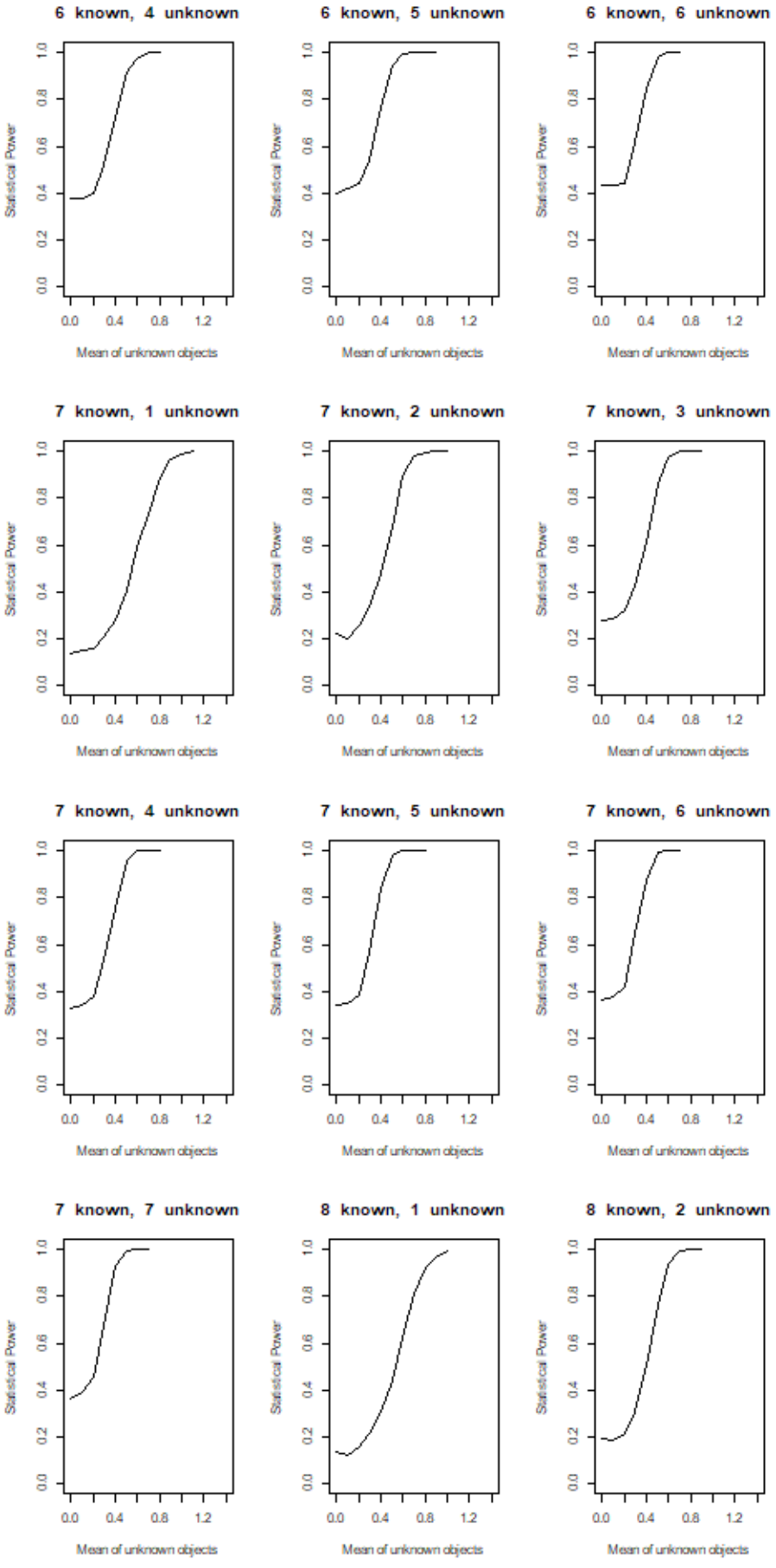
10	1	0.5	93	59
10	1	1.0	79	54
10	1	1.5	50	38
10	1	2.0	33	13
10	1	2.5	19	2
10	1	3.0	8	0
10	1	3.5	8	0
10	1	4.0	7	0
10	2	0.0	99	82
10	2	0.5	94	65
10	2	1.0	86	18
10	2	1.5	63	3
10	2	2.0	33	1
10	2	2.5	22	0
10	2	3.0	14	0
10	2	3.5	9	0
10	4	0.0	100	99
10	4	0.5	99	75
10	4	1.0	94	10
10	4	1.5	59	0
10	4	2.0	30	0
10	4	2.5	13	0
10	4	3.0	1	0
10	8	0.0	97	98
10	8	0.5	99	81
10	8	1.0	90	1
10	8	1.5	58	0

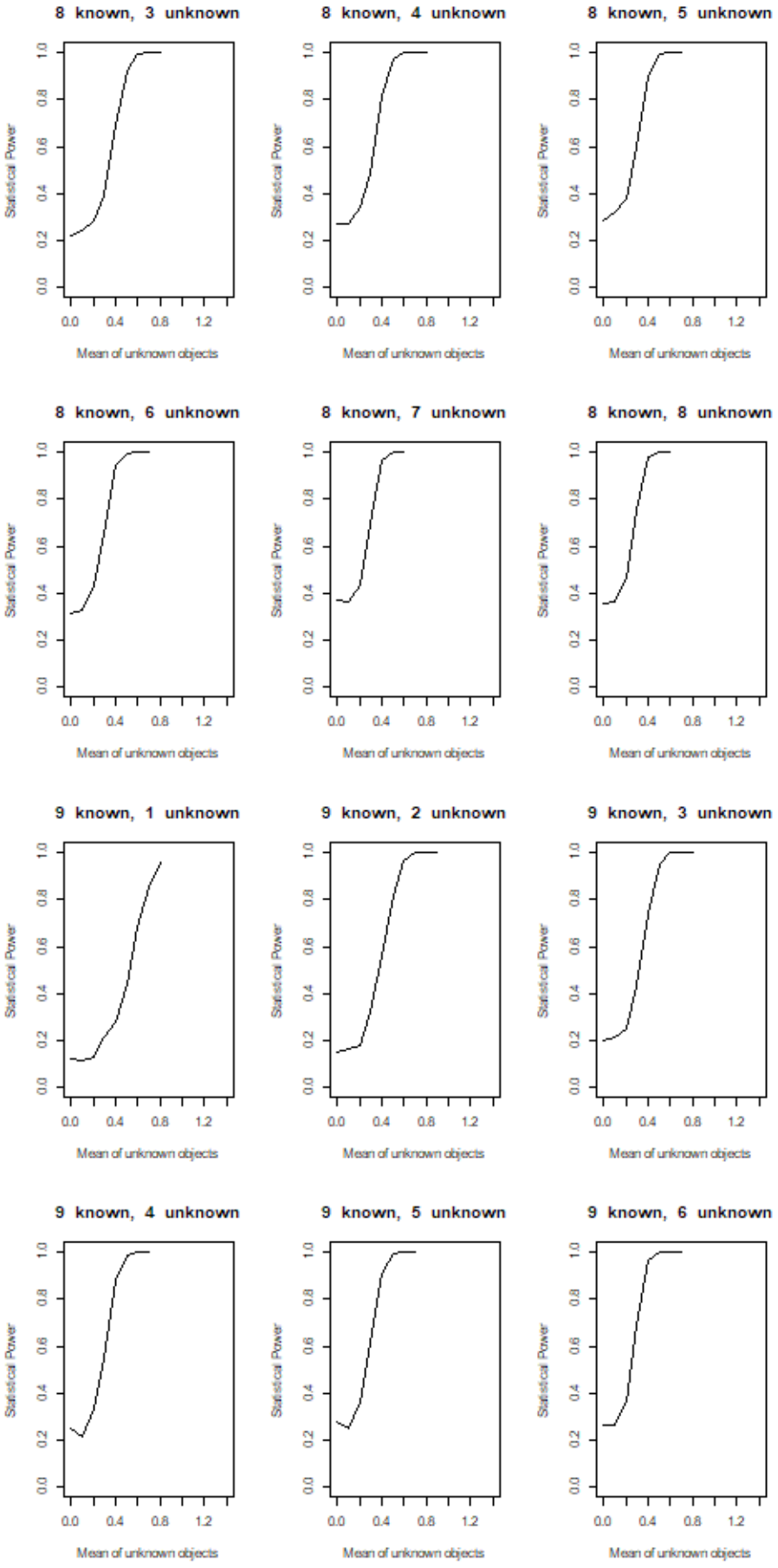
10	8	2.0	21	0
10	16	0.0	94	100
10	16	0.5	94	83
10	16	1.0	86	0
10	16	1.5	44	0
10	16	2.0	14	0
10	32	0.0	77	100
10	32	0.5	82	86
10	32	1.0	95	0
10	32	1.5	20	0
10	64	0.0	1	1
10	64	0.5	67	88
10	64	1.0	88	0
10	64	1.5	2	0
20	1	0.0	100	73
20	1	0.5	97	63
20	1	1.0	74	69
20	1	1.5	38	66
20	1	2.0	25	19
20	1	2.5	19	0
20	1	3.0	4	0
20	2	0.0	100	68
20	2	0.5	99	87
20	2	1.0	91	58
20	2	1.5	58	11
20	2	2.0	30	0
20	2	2.5	21	0

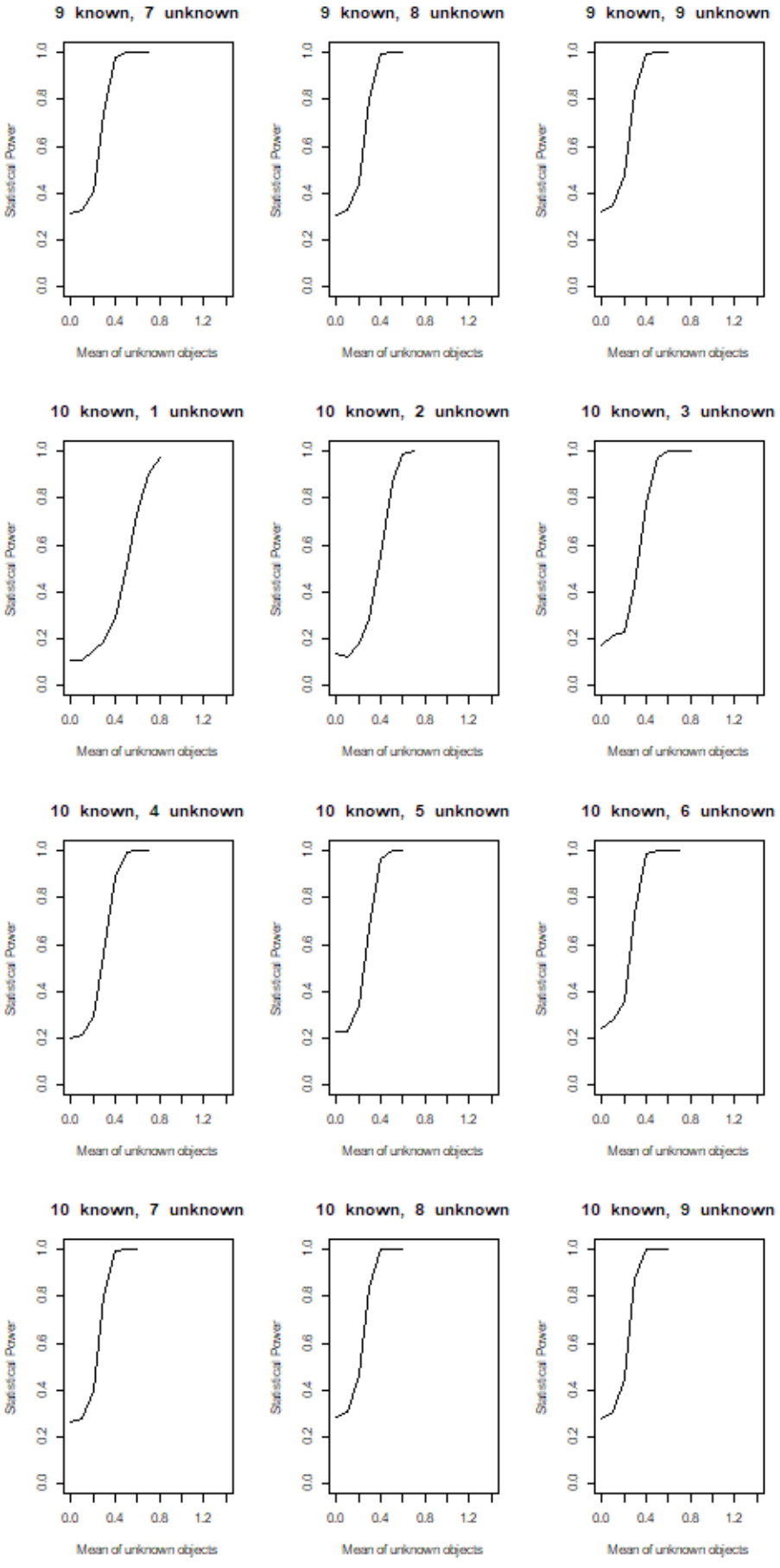
20	2	3.0	2	0
20	4	0.0	100	100
20	4	0.5	100	98
20	4	1.0	89	21
20	4	1.5	58	0
20	4	2.0	43	0
20	4	2.5	6	0
20	8	0.0	100	100
20	8	0.5	100	99
20	8	1.0	92	5
20	8	1.5	57	0
20	8	2.0	30	0
20	16	0.0	100	100
20	16	0.5	100	100
20	16	1.0	89	0
20	16	1.5	48	0
20	32	0.0	100	100
20	32	0.5	100	100
20	32	1.0	93	0
20	32	1.5	46	0

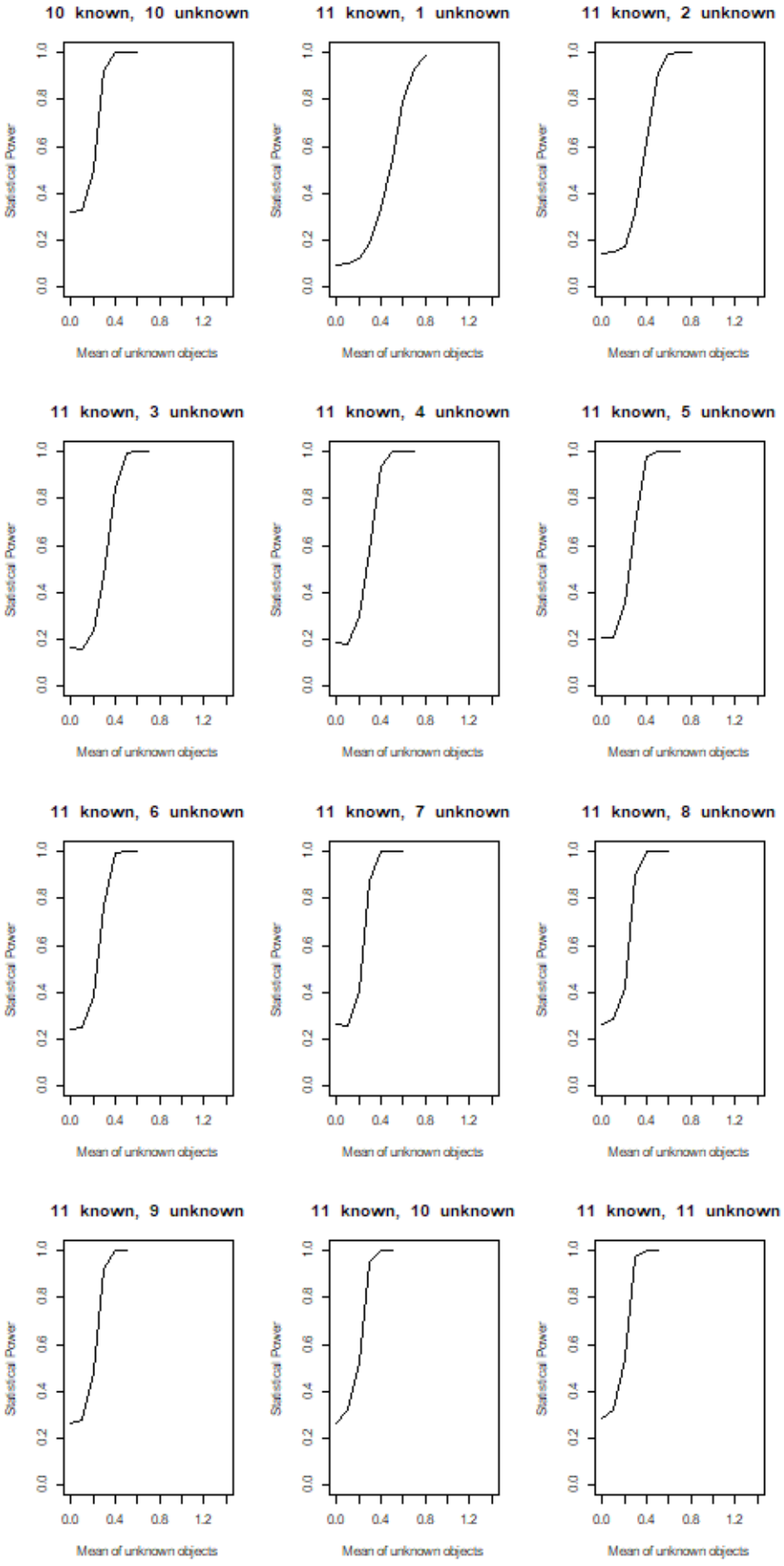
Table 8: The percent of test objects classified as belonging to the training objects set, given fluctuating numbers of training objects, number of dimensions, and training coordinates.

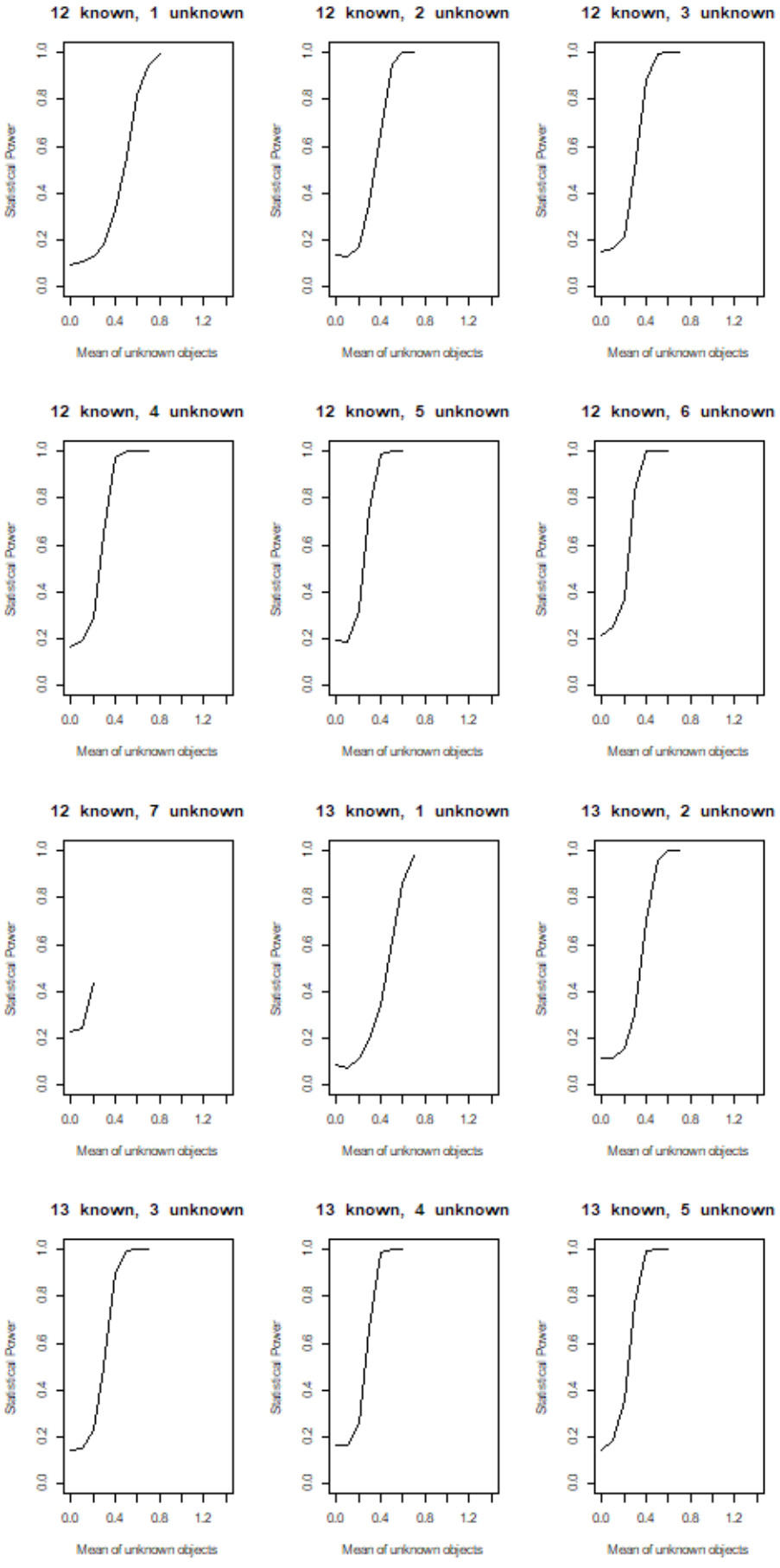

4 known, 1 unknown
4 known, 2 unknown
4 known, 3 unknown
4 known, 4 unknown
5 known, 1 unknown
5 known, 2 unknown
5 known, 3 unknown
5 known, 4 unknown
5 known, 5 unknown
6 known, 1 unknown
6 known, 2 unknown
6 known, 3 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2

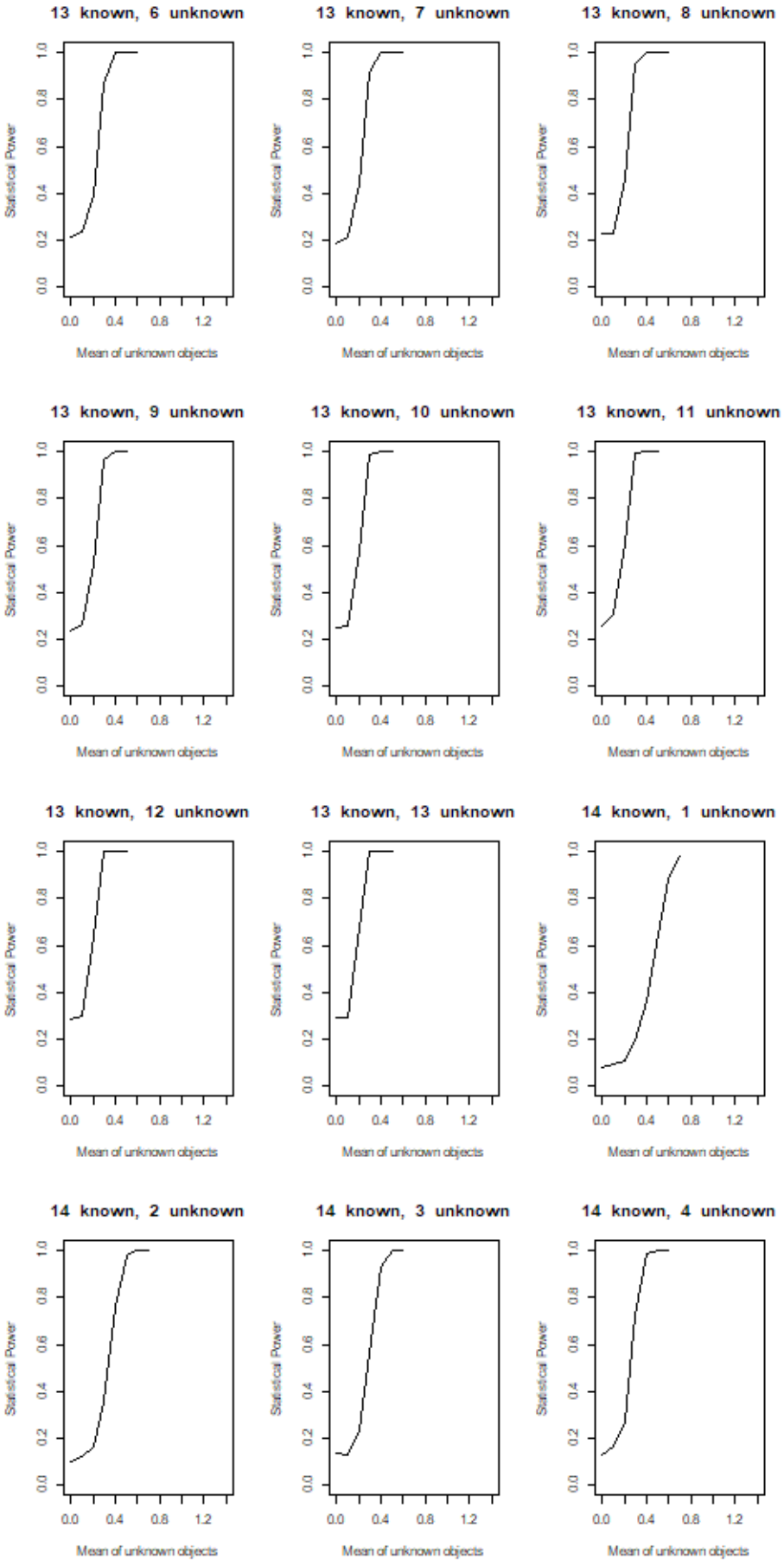

6 known, 4 unknown
6 known, 5 unknown
6 known, 6 unknown
7 known, 1 unknown
7 known, 2 unknown
7 known, 3 unknown
7 known, 4 unknown
7 known, 5 unknown
7 known, 6 unknown
7 known, 7 unknown
8 known, 1 unknown
8 known, 2 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2

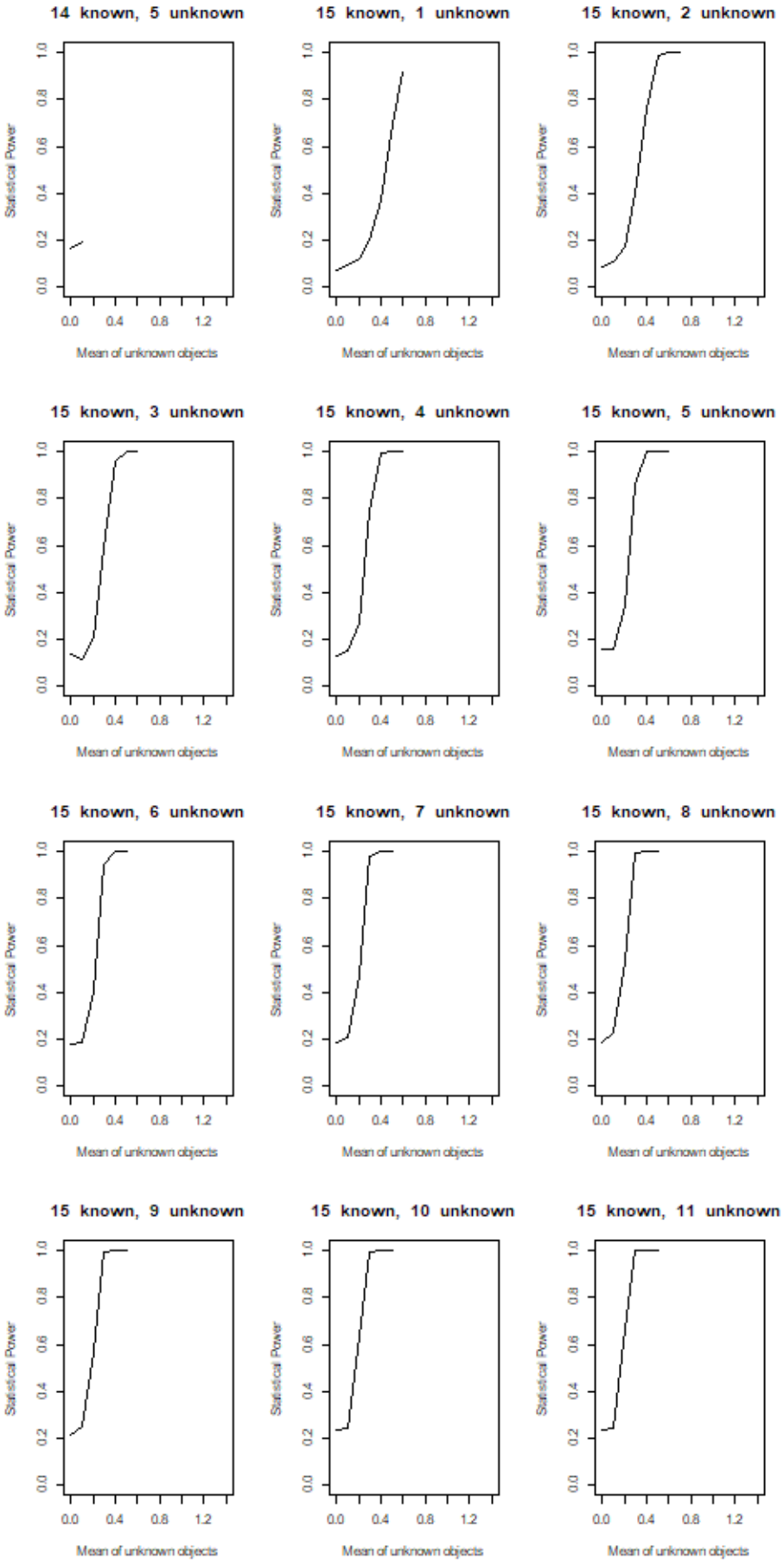

8 known, 3 unknown
8 known, 4 unknown
8 known, 5 unknown
8 known, 6 unknown
8 known, 7 unknown
8 known, 8 unknown
9 known, 1 unknown
9 known, 2 unknown
9 known, 3 unknown
9 known, 4 unknown
9 known, 5 unknown
9 known, 6 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2

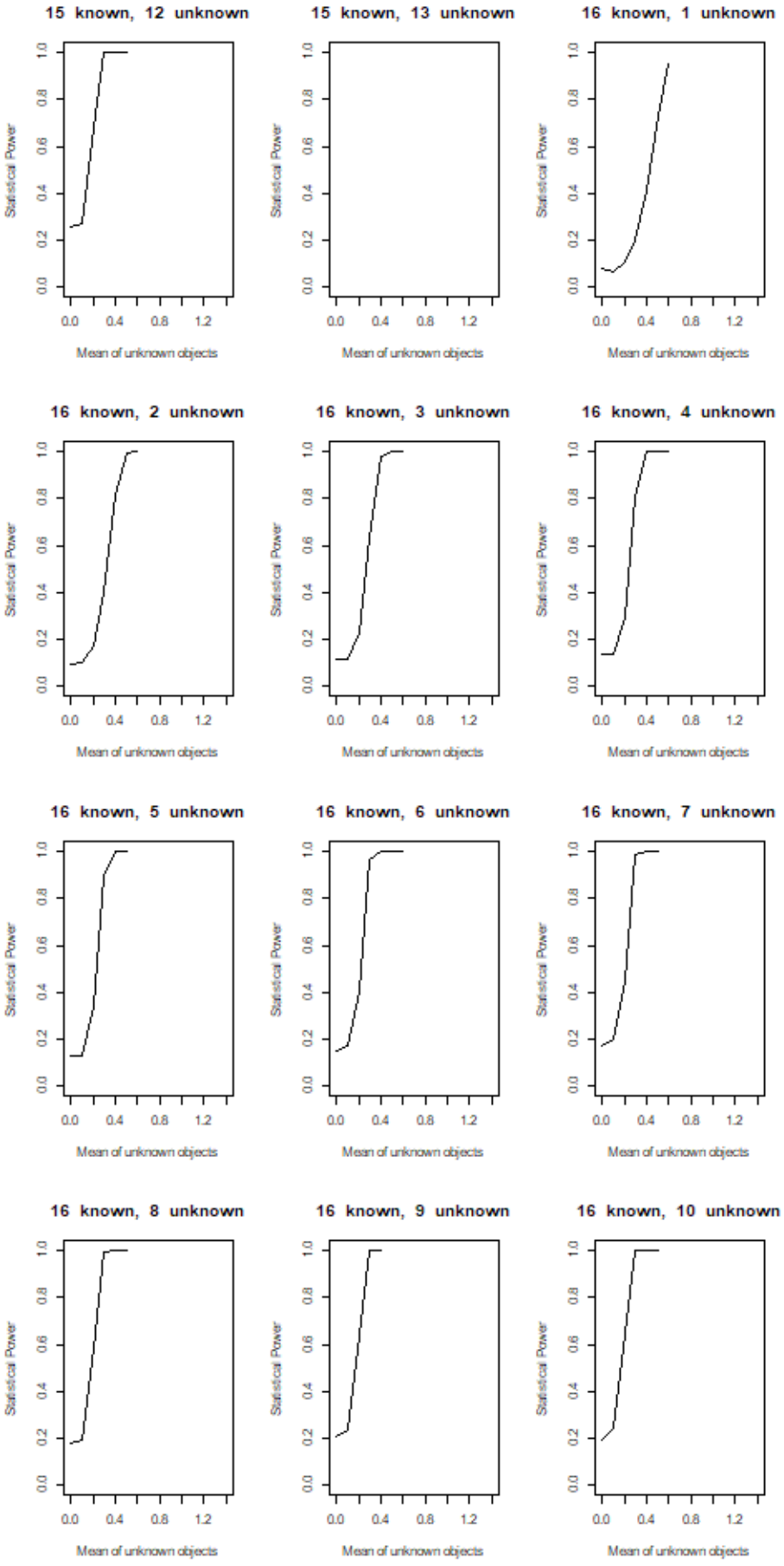

9 known, 7 unknown
9 known, 8 unknown
9 known, 9 unknown
10 known, 1 unknown
10 known, 2 unknown
10 known, 3 unknown
10 known, 4 unknown
10 known, 5 unknown
10 known, 6 unknown
10 known, 7 unknown
10 known, 8 unknown
10 known, 9 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2

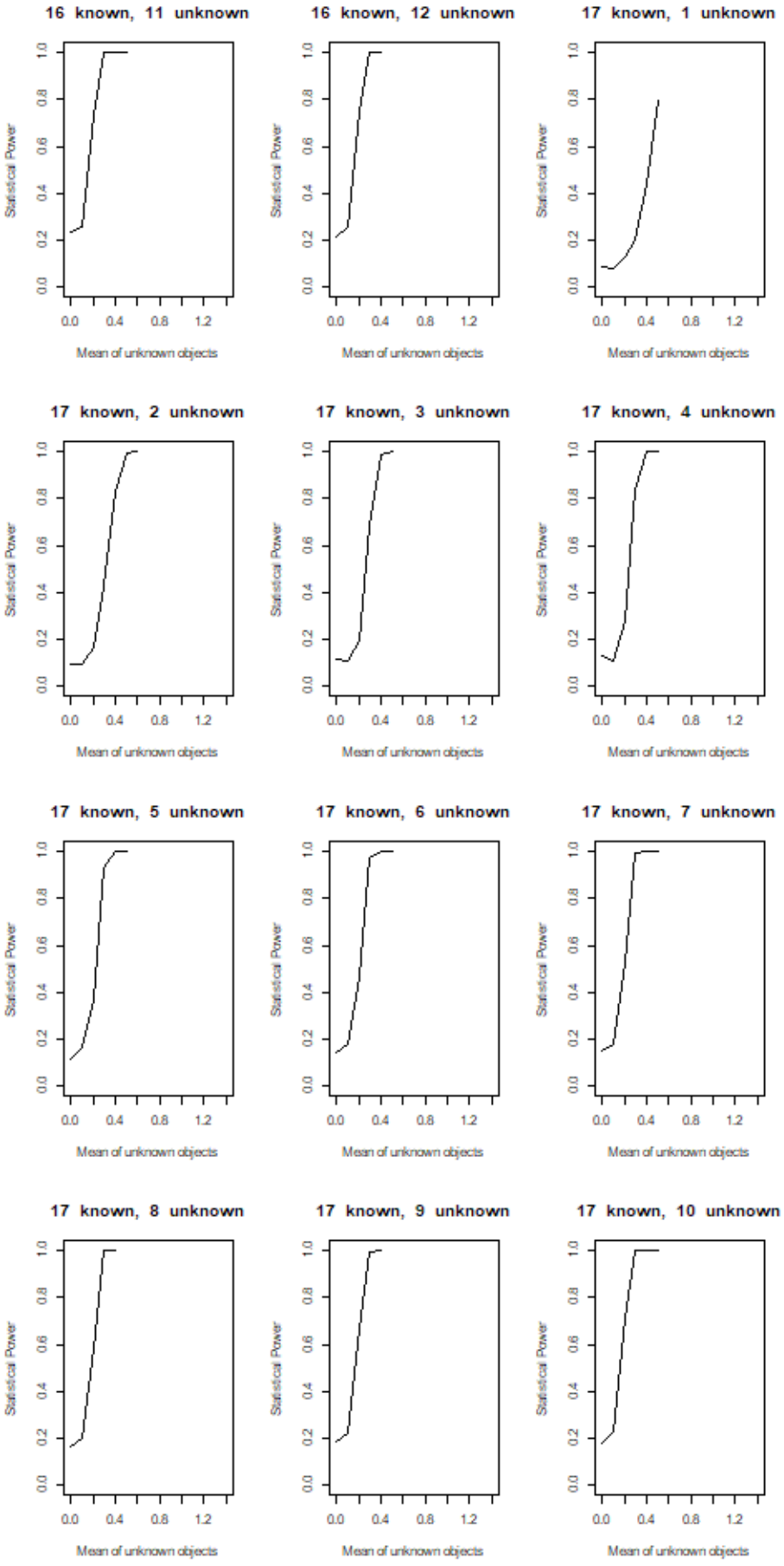

10 known, 10 unknown
11 known, 1 unknown
11 known, 2 unknown
11 known, 3 unknown
11 known, 4 unknown
11 known, 5 unknown
11 known, 6 unknown
11 known, 7 unknown
11 known, 8 unknown
11 known, 9 unknown
11 known, 10 unknown
11 known, 11 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2

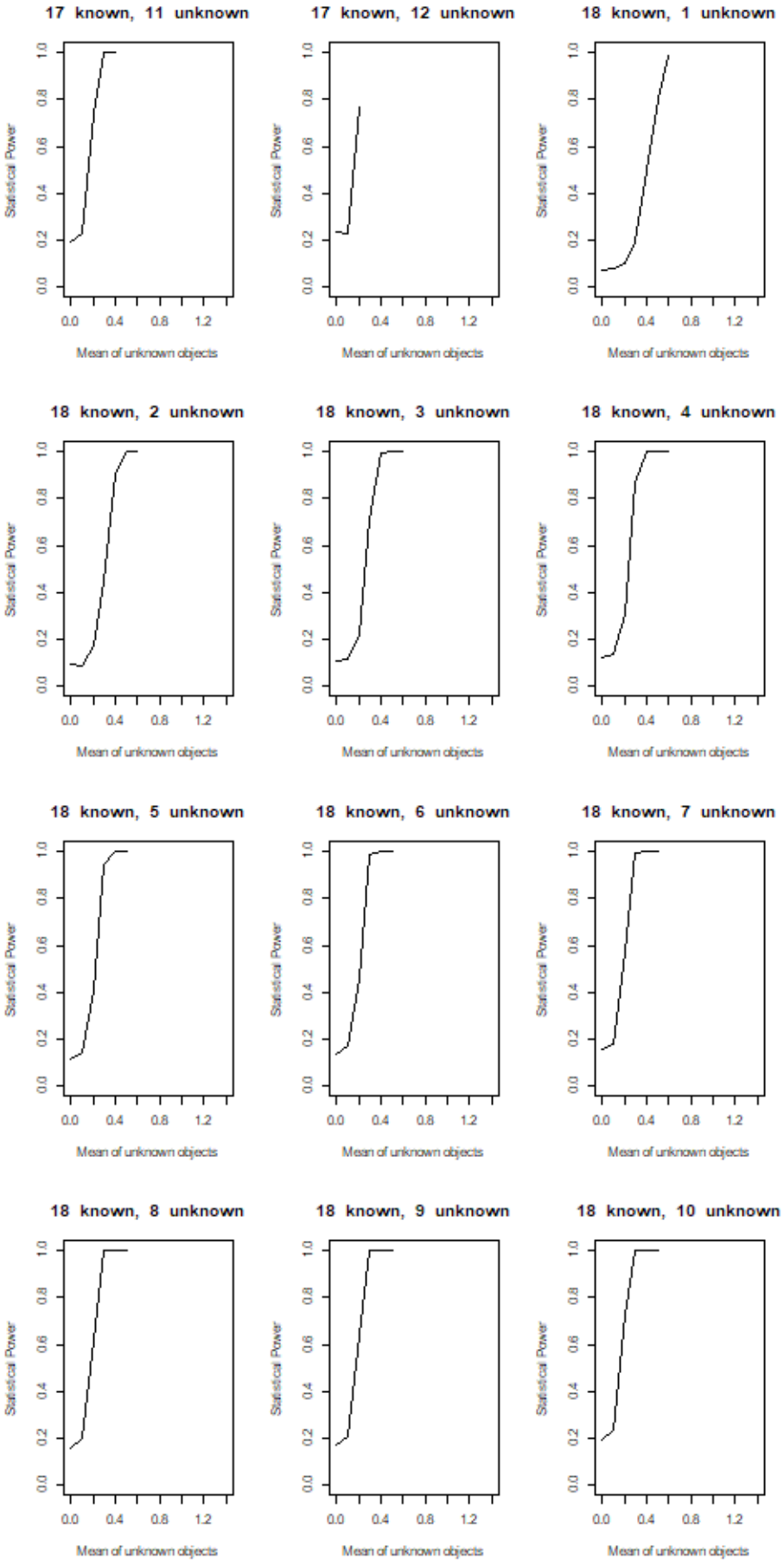

12 known, 1 unknown
12 known, 2 unknown
12 known, 3 unknown
12 known, 4 unknown
12 known, 5 unknown
12 known, 6 unknown
12 known, 7 unknown
13 known, 1 unknown
13 known, 2 unknown
13 known, 3 unknown
13 known, 4 unknown
13 known, 5 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2

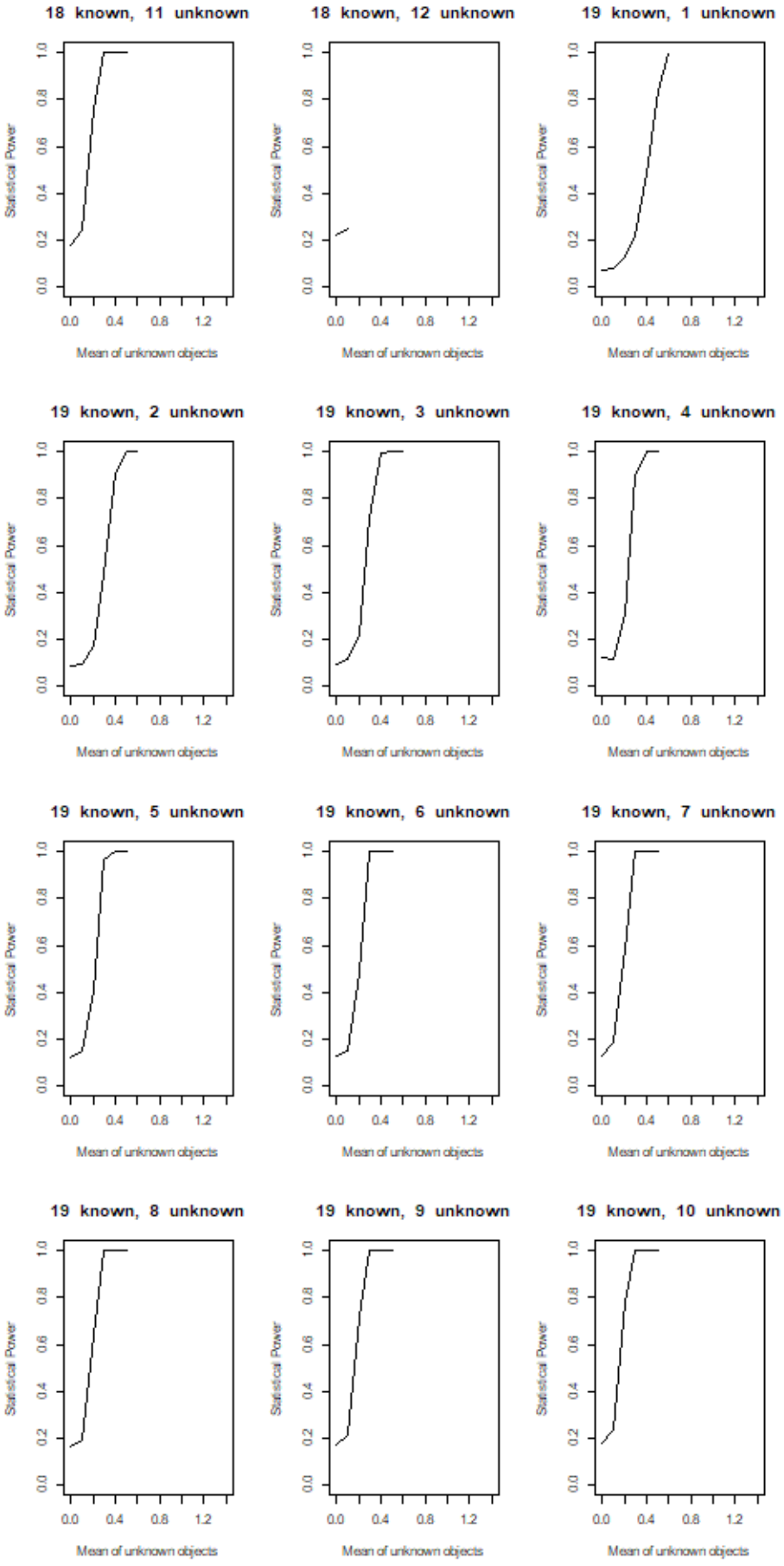

13 known, 6 unknown
13 known, 7 unknown
13 known, 8 unknown
13 known, 9 unknown
13 known, 10 unknown
13 known, 11 unknown
13 known, 12 unknown
13 known, 13 unknown
14 known, 1 unknown
14 known, 2 unknown
14 known, 3 unknown
14 known, 4 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2

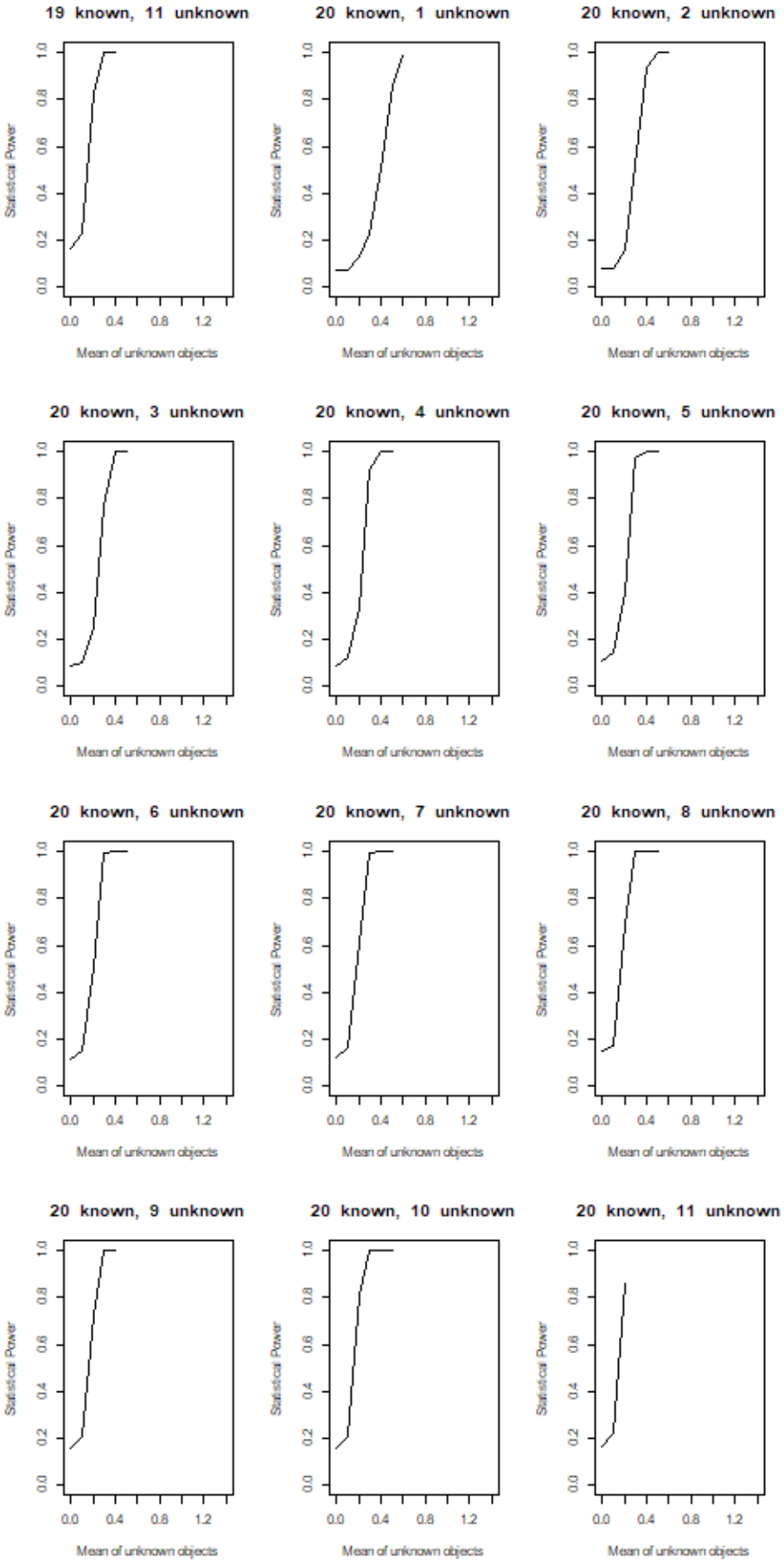

14 known, 5 unknown
15 known, 1 unknown
15 known, 2 unknown
15 known, 3 unknown
15 known, 4 unknown
15 known, 5 unknown
15 known, 6 unknown
15 known, 7 unknown
15 known, 8 unknown
15 known, 9 unknown
15 known, 10 unknown
15 known, 11 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2


15 known, 12 unknown
15 known, 13 unknown
16 known, 1 unknown
16 known, 2 unknown
16 known, 3 unknown
16 known, 4 unknown
16 known, 5 unknown
16 known, 6 unknown
16 known, 7 unknown
16 known, 8 unknown
16 known, 9 unknown
16 known, 10 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2


16 known, 11 unknown
16 known, 12 unknown
17 known, 1 unknown
17 known, 2 unknown
17 known, 3 unknown
17 known, 4 unknown
17 known, 5 unknown
17 known, 6 unknown
17 known, 7 unknown
17 known, 8 unknown
17 known, 9 unknown
17 known, 10 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2


17 known, 11 unknown
17 known, 12 unknown
18 known, 1 unknown
18 known, 2 unknown
18 known, 3 unknown
18 known, 4 unknown
18 known, 5 unknown
18 known, 6 unknown
18 known, 7 unknown
18 known, 8 unknown
18 known, 9 unknown
18 known, 10 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.4
0.8
1.2


18 known, 11 unknown
18 known, 12 unknown
19 known, 1 unknown
19 known, 2 unknown
19 known, 3 unknown
19 known, 4 unknown
19 known, 5 unknown
19 known, 6 unknown
19 known, 7 unknown
19 known, 8 unknown
19 known, 9 unknown
19 known, 10 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2


19 known, 11 unknown
20 known, 1 unknown
20 known, 2 unknown
20 known, 3 unknown
20 known, 4 unknown
20 known, 5 unknown
20 known, 6 unknown
20 known, 7 unknown
20 known, 8 unknown
20 known, 9 unknown
20 known, 10 unknown
20 known, 11 unknown
Statistical Power
Mean of unknown objects
0.0
0.2
0.4
0.6
0.8
1.0
1.2