Determination of Appropriate Methods for Determining if Topical Drugs are Bioequivalent

Bianca Verlangieri and Nick Koprowicz

May 01, 2017

Introduction

In order for generic drugs to hit the market, they must be approved by the FDA. That is, they must be shown to be safe and effective. One way that generic drugs can gain approval is if it is shown that the drug is *bioequivalent* to some other brand name drug that has already been approved. Two drugs are said to be bioequivalent, if they are, for all intents and purposes, the same.

Typically, to assess bioequivalence, two formulations of a drug are administered to subjects and the concentration of the drug is measured at various times. This is fairly easy to do with drugs that act through the bloodstream, as the drug can be measured through blood samples. Measuring concentration of topical drugs, however, is more difficult. To assess bioequivalence of topical drugs, a drug is applied to several sites on the skin. Some of these sites are used to assess uptake and others are used to assess clearance. We would like to determine a method to use these measurements to determine if two topical drugs are bioequivalent.

We will consider two specific characteristics in trying to evaluate whether the concentration curves of two topical drugs are similar: *rate* and *extent* of absorption. We consider the following five parameters to measure these characteristics:

Parameter	Measure
C_{max}	The maximum concentration. A measure of extent.
AUC	The area under the concentration curve. A measure of rate and extent.
AUC*	The area under the concentration curve during the clearance phase. A measure of rate and extent.
e^k	Where it is assumed that during the clearance phase, the concentration is $C(t) = C_0 e^{-kt}$. A measure of rate.
$C(1.5)$	The concentration of the drug at the beginning of the clearance phase. A measure of extent.

Since these parameters measure rate and extent, we can show that two formulations are bioequivalent by estimating these parameters for each formulation and showing that their ratio is approximately 1. Since tests are performed on a collection of subjects, we need a way to estimate the ratios of these parameters across a large group of subjects. We consider the following 5 ratios:

- Median ratio
- Ratio of means
- Mean log ratio
- Bias-corrected ratio
- Lognormal ratio

Additionally, since these parameters are based on measurements taken at different times, we also consider the following four times for taking measurements (measured in minutes).

- Time 1 = (0.25, 0.5, 1, 1.5, 4.5, 7.5, 10.5, 13.5)
- Time 2 = (0.1, 0.35, 0.75, 1.25, 3.0, 6.0, 9.0, 12.0)
- Time 3 = (0.35, 0.75, 1.25, 3.0, 6.0, 9.0, 12.0, 15.0)
- Time 4 = (1.5, 1.5, 1.5, 1.5, 10.5, 10.5, 10.5, 10.5)

Given these parameters, ratios, and times of measurement, there are many combinations that could be used to try to determine the bioequivalence of two formulations. It is not plausible that all of these combinations could be considered in any one experiment. Our goal then is to try to find the best set of parameters, ratios, and times to use in an experiment in which one wants to determine the bioequivalence of two formulations. We do this with simulated data.

Initial Simulations

Methods for Simulating Data

In order to determine which classification methods are best, we use simulation based on data collected on the absorption of tretinoin, an ingredient used to treat acne. The data set contains concentrations of two different formulations of tretinoin (Formula A and Formula B), collected during the times in Time 1.

We start by choosing a true ratio, r, a number of subjects, n, and a set of times. We then use the means, variances, and covariance matrix from Formula A from the tretinoin data to create a simulated Formula B. Our simulated data set should then have approximately the same distribution as the real data on Formula A, except that the ratio of the values will be approximately r, the chosen ratio. Good combinations of parameters, ratios, and times will classify simulated formulations with a ratio of approximately 1 as bioequivalent, and will classify simulated formulations with a ratio different from 1 as bioinequivalent.

For our initial analysis, we simulate 100 data sets, and use different methods to create 90% confidence intervals for the value of the true ratio of different parameters from Formulation A and the simulated Formula B. We use four methods to create confidence intervals. Two methods are based on bootstrapping, one is based on permutation, and another uses an actual formula for each kind of ratio. Based on these confidence intervals, we classify the data as bioequivalent, bioinequivalent, or say the test is inconclusive. If the confidence interval for any ratio is contained in the interval (0.8, 1.25), we classify the formulations as bioequivalent; if the confidence interval for any ratio is disjoint from (0.8, 1.25), we classify the formulations as bioinequivalent; otherwise, we classify the formulations as inconclusive.

For values of r close to 1, i.e. in the range (0.8, 1.25), we expect the two formulations to be classified as bioequivalent, and for values of r far from 1, i.e. outside of the range (0.8, 1.25), we expect the formulations to be classified as bioinequivalent. We would like to find the best combinations of parameters, ratios, and times, such that these expectations are met. Our results

from the initial set of simulated data, for time 1, are shown in Figure 1 in the Appendix. It is based on the following table. Each entry contains the percent of the 100 data sets that were classified as bioequivalent, bioinequivalent, or neither.

Table 1

	Formula	Bootstrap Method 1	Bootstrap Method 2	Permutation
C_{max}, median ratio				
C_{max}, ratio of means				
C_{max}, bias-corrected ratio				
C_{max}, lognormal ratio				
C_{max}, mean of log ratios				
AUC, median ratio				
AUC, ratio of means				
AUC, bias-corrected ratio				
AUC, lognormal ratio				
AUC, mean of log ratios				
AUC*, median ratio				

We create similar tables for time 2, time 3, and time 4, for $r = 1$ and $r = 1.5$, and for skewed and normal simulated data, giving us 16 tables in total.

Observations

We determine which parameter/ratio pairs work best using the following score, in which higher scores indicate that the formulas are classified accurately a larger percent of the time, i.e. the formulations are classified as bioequivalent for $r = 1$ and bioinequivalent for $r = 1.5$.

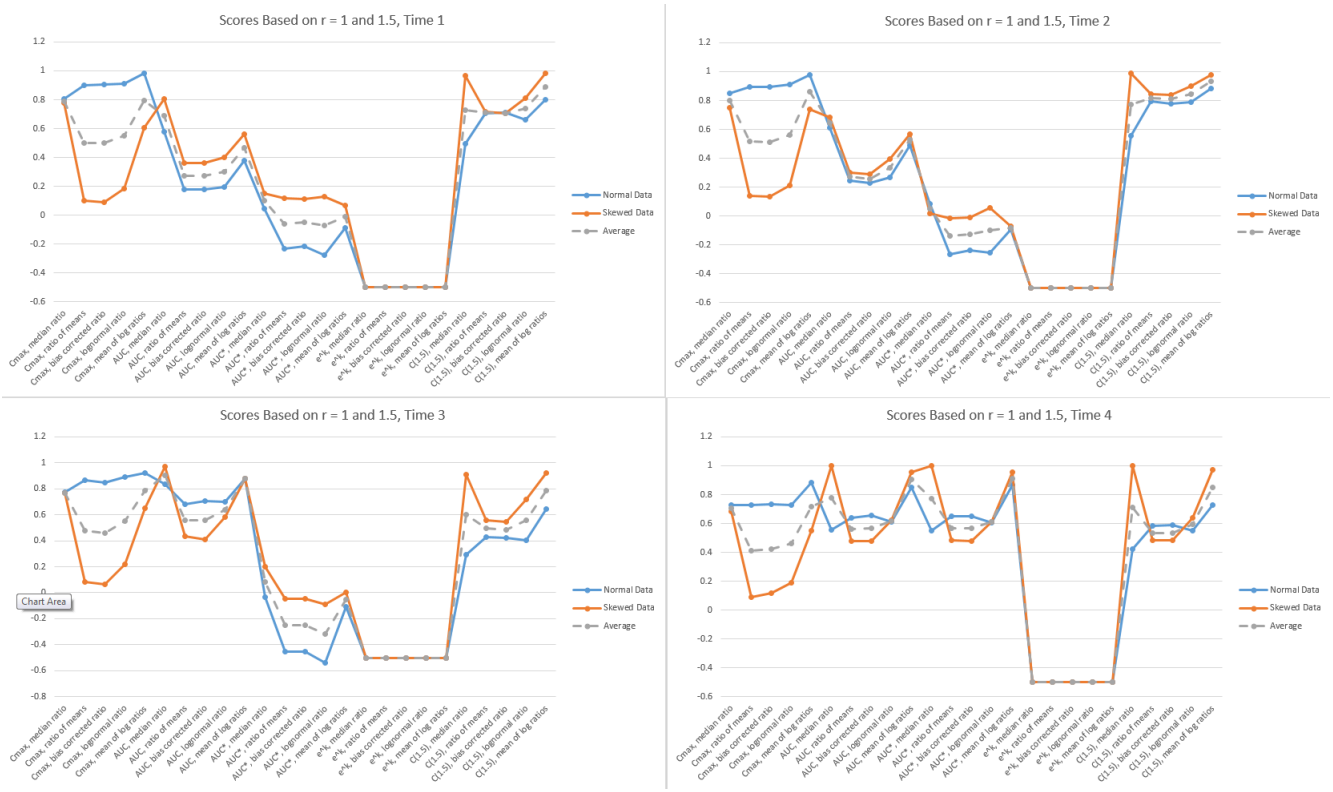
score = (percent of data classified as bioequivalent when $r = 1$) - 1.5*(percent of data classified as bioequivalent when $r = 1.5$)

In order to obtain one score for each ratio, we choose to average the results from our four methods of creating confidence intervals. For example, if a parameter/ratio pair classified formulations correctly as bioequivalent 90% of the time using bootstrap 1, 90% of the time using bootstrap 2, 80% of the time using permutations, and 80% of the time using formulas, we say the parameter/ratio pair classified the formulations correctly 85% of the time. We see that the following individual parameter/ratio pairs work best:

1. C(1.5) mean of log
2. Cmax mean of log
3. Cmax median ratio
4. AUC median ratio

5. C(1.5) median ratio
6. AUC mean of log
7. C(1.5) lognormal
8. C(1.5) ratio of means

The following figure justifies this decision:



Eliminating e^k

We notice that the ratios for the parameter e^k classify the formulations as bioequivalent 100% of the time, regardless of the value for r. Doing some algebra, we see that is it actually the rate k that depends on the ratio r in the way that we were anticipating. Thus we choose not to consider e^k as a parameter moving forward.

Secondary Simulations

Methods for Simulating Data

As mentioned previously, in order to classify a topical drug as bioequivalent or bioinequivalent, both rate and extent of absorption need to be the same. It follows that we need at least two parameter/ratio combinations to classify the formulations - one which measures rate, and another which measures extent. Using our eight highest scoring formulation/ratio combinations from the initial set of simulations, we choose subsets such that both rate and extent are measured. We then simulate 100 new sets of data and calculate a confidence interval for the parameter/ratio pairs as

we did in the first set of simulations to analyze how well parameter/ratio pairs work in conjunction with one another, that is, how often formulations are classified correctly when both *rate and extent* are taken into account. For each set, if both parameter/ratio pairs classify as bioequivalent, we say the formulations are bioequivalent. If one or both parameter/ratio pairs classify as bioinequivalent, we say the formulations are bioinequivalent. Otherwise the pair is considered to be inconclusive.

Since two of our eight selected parameter/ratio pairs measure rate and six measure extent, we test 12 pairs of classifiers:

1. AUC median ratio / C(1.5) mean of log
2. AUC median ratio / Cmax mean of log
3. AUC median ratio / Cmax median ratio
4. AUC median ratio / C(1.5) median ratio
5. AUC median ratio / C(1.5) lognormal
6. AUC median ratio / C(1.5) ratio of means
7. AUC mean of log / C(1.5) mean of log
8. AUC mean of log / Cmax mean of log
9. AUC mean of log / Cmax median ratio
10. AUC mean of log / C(1.5) median ratio
11. AUC mean of log / C(1.5) lognormal
12. AUC mean of log / C(1.5) ratio of means

Observations

We test these eight individual parameter/ratio pairs and 12 combinations with the new set of simulated data. The percent classified correctly, that is, classified as bioequivalent when $r = 1$ and percent classified as bioinequivalent when $r = 1.5$, can be shown in Figures 2 and 3 in the appendix. Using the same scoring method, we find that the following pairs perform best, given that data is collected at the same times as the tretinoin data (Time 1):

Best for Normal Data, Time 1:

Pair	Time	Score
AUC mean of log/Cmax mean of log	Time 1	.98
AUC median ratio/Cmax mean of log	Time 1	.915

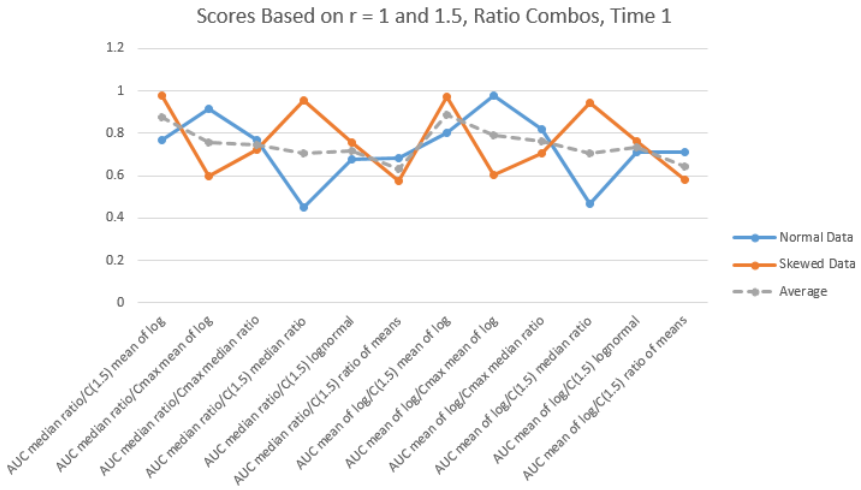
Best for Skewed Data, Time 1:

Pair	Time	Score
AUC median ratio/C(1.5) mean of log	Time 1	.98
AUC mean of log/C(1.5) mean of log	Time 1	.975

Best for Both Normal and Skewed Data, Time 1:

Pair	Time	Avg Score
AUC mean of log/C(1.5) mean of log	Time 1	.8875
AUC median ratio/C(1.5) mean of log	Time 1	.875

These results can be shown in the following figure:



Using the same scoring method across various times, we conclude that the following pairs performed best overall, and thus would be best to use in an experiment:

Best for Normal Data, All Times:

Pair	Time	Score
AUC mean of log/Cmax mean of log	Time 1	.98
AUC mean of log/Cmax mean of log	Time 2	.9725
AUC mean of log/Cmax mean of log	Time 3	.965
AUC median ratio/Cmax mean of log	Time 2	.9275

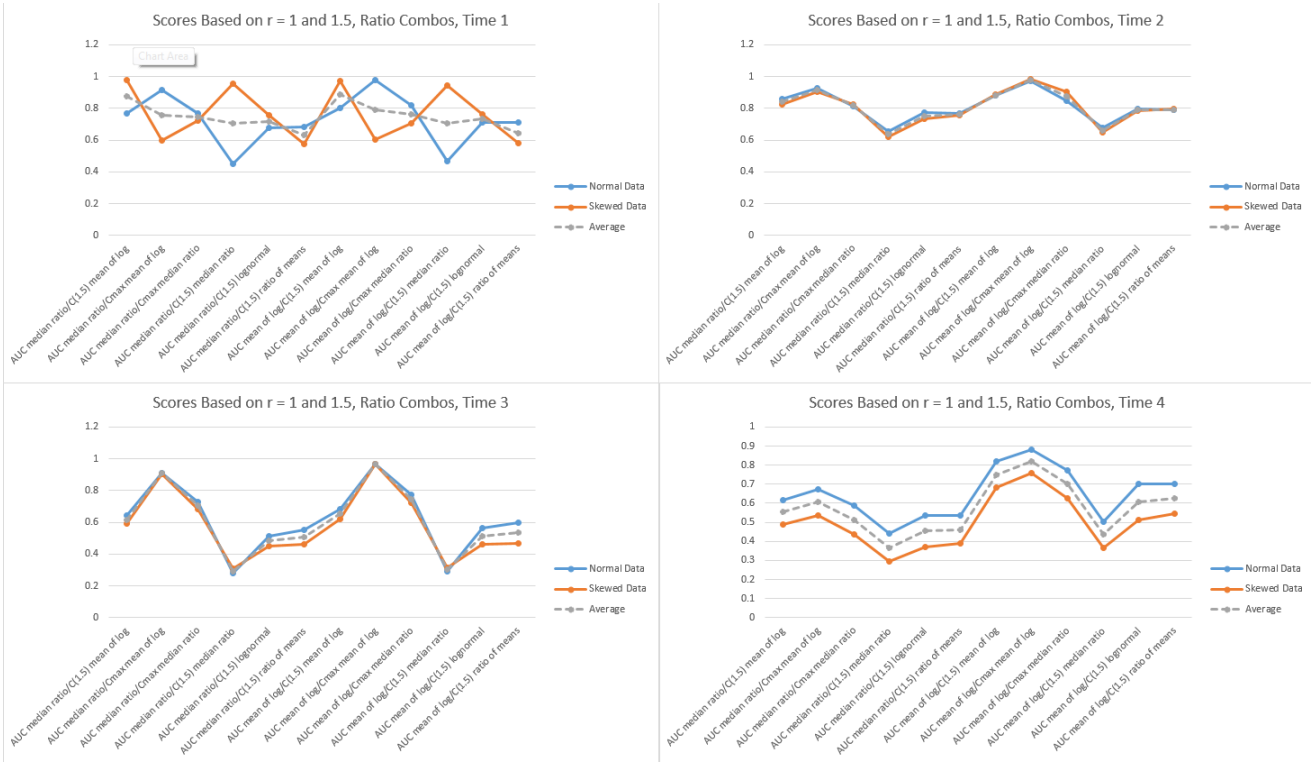
Best for Skewed Data, All Times:

Pair	Time	Score
AUC mean of log/Cmax mean of log	Time 2	.9825
AUC median ratio/C(1.5) mean of log	Time 1	.98
AUC mean of log/C(1.5) mean of log	Time 1	.975
AUC mean of log/Cmax mean of log	Time 3	.965

Best for Both Normal and Skewed Data, All Times:

Pair	Time	Avg Score
AUC mean of log/Cmax mean of log	Time 2	.9775
AUC mean of log/Cmax mean of log	Time 3	.965
AUC median ratio/Cmax mean of log	Time 2	.915
AUC median ratio/Cmax mean of log	Time 3	.9075

These results can be shown in the following figures:



These results suggest the optimal parameter/ratio pairs and times that should be used when conducting an experiment. To determine if two drugs are bioequivalent, we can suggest that the above ratios be calculated when the data is collected at the given times.

We also consider different values of r for the second round of simulations, namely $r = 0.8$ and $r = 1.2$. Values of $r = 0.8$ are right on the border between bioequivalent and bioinequivalent, while $r = 1.2$ is bioequivalent but just barely so. Thus, we find that when we simulate data using these values, the formulations are largely classified as inconclusive or bioequivalent. We want to see that our the methods that perform well for $r = 1$ and $r = 1.5$ also perform well on the boundary. We analyze our results by considering the percent of the time that the simulated formulation is not classified as bioinequivalent, since we expect the formulations to either be classified at bioinequivalent or inconclusive. These results are shown in Figures 4 and 5 in the appendix. We find that the methods that perform best for $r = 1$ and $r = 1.5$ also perform best at the boundary and we decide it is not necessary to re-score parameter/ratio pairs under $r = 0.8$ or $r = 1.2$.

Conclusions

For data collected at the same times as the tretinoin data, the combination of the AUC mean of log ratio and C(1.5) mean of log ratio is the most accurate classifier of bioequivalence and bioinequivalence. The second most accurate is the combination of the AUC median ratio and C(1.5) mean of log ratio. Both of these methods work better for skewed data rather than normal data. For only the normal data set, the combination of the AUC median ratio and Cmax mean of log ratio is the best classifier.

Across all four times, the combination of AUC mean of log and Cmax mean of log ratios at Time 2 is the best classifier of bioequivalence and bioinequivalence. This parameter/ratio works best for

skewed data, but overall still is a very accurate classifier for both distributions. This combination is also the second best classifier when measurements were taken at Time 3. The AUC median ratio and Cmax mean of log ratio at Time 2 and Time 3 are also excellent classifiers of bioequivalence.

Since we want the best classification methods regardless of distribution, and can take measurements at different times, we recommend the following:

Best for Both Normal and Skewed Data, All Times:

Pair	Time	Score
AUC mean of log/Cmax mean of log	Time 2	9
AUC mean of log/Cmax mean of log	Time 3	9
AUC median ratio/Cmax mean of log	Time 2	9
AUC median ratio/Cmax mean of log	Time 3	9

Appendix

Figure 1: Table for r = 1, Normal Data, Time 1

	Formula			Bootstrapping Method 1			Bootstrapping Method 2			Permutation		
	Bioeq	Bioineq	Neither	Bioeq	Bioineq	Neither	Bioeq	Bioineq	Neither	Bioeq	Bioineq	Neither
Cmax, median ratio	0.75	0	0.25	0.85	0	0.15	0.76	0	0.24	0.86	0	0.14
Cmax, ratio of means	0.91	0	0.09	0.91	0	0.09	0.9	0	0.1	0.89	0	0.11
Cmax, bias corrected ratio	0.91	0	0.09	0.92	0	0.08	0.91	0	0.09	0.88	0	0.12
Cmax, lognormal ratio	0.94	0	0.06	0.91	0	0.09	0.91	0	0.09	0.89	0	0.11
Cmax, mean of log ratios	0.99	0	0.01	0.98	0	0.02	0.99	0	0.01	0.98	0	0.02
AUC, median ratio	0.92	0	0.08	0.95	0	0.05	0.9	0	0.1	0.96	0	0.04
AUC, ratio of means	0.92	0	0.08	0.92	0	0.08	0.94	0	0.06	0.91	0	0.09
AUC, bias corrected ratio	0.92	0	0.08	0.91	0	0.09	0.94	0	0.06	0.93	0	0.07
AUC, lognormal ratio	0.93	0	0.07	0.92	0	0.08	0.96	0	0.04	0.95	0	0.05
AUC, mean of log ratios	1	0	0	1	0	0	1	0	0	0.99	0	0.01
AUC*, median ratio	0.78	0	0.22	0.85	0	0.15	0.71	0	0.29	0.89	0	0.11
AUC*, ratio of means	0.78	0	0.22	0.77	0	0.23	0.8	0	0.2	0.78	0	0.22
AUC*, bias corrected ratio	0.77	0	0.23	0.77	0	0.23	0.8	0	0.2	0.77	0	0.23
AUC*, lognormal ratio	0.77	0	0.23	0.79	0	0.21	0.81	0	0.19	0.79	0	0.21
AUC*, mean of log ratios	0.96	0	0.04	0.96	0	0.04	0.96	0	0.04	0.96	0	0.04
e^k, median ratio	1	0	0	1	0	0	1	0	0	1	0	0
e^k, ratio of means	1	0	0	1	0	0	1	0	0	1	0	0
e^k, bias corrected ratio	1	0	0	1	0	0	1	0	0	1	0	0
e^k, lognormal ratio	1	0	0	1	0	0	1	0	0	1	0	0
e^k, mean of log ratios	1	0	0	1	0	0	1	0	0	1	0	0
C(1.5), median ratio	0.45	0	0.55	0.57	0	0.43	0.42	0	0.58	0.55	0	0.45
C(1.5), ratio of means	0.72	0	0.28	0.71	0	0.29	0.74	0	0.26	0.65	0	0.35
C(1.5), bias corrected ratio	0.73	0	0.27	0.73	0	0.27	0.74	0	0.26	0.64	0	0.36
C(1.5), lognormal ratio	0.66	0	0.34	0.64	0	0.36	0.7	0	0.3	0.65	0	0.35
C(1.5), mean of log ratios	0.78	0	0.22	0.8	0	0.2	0.8	0	0.2	0.81	0	0.19

Figure 2: Percent Bioequivalent when r = 1

Figure 3: Percent Bioinequivalent when r = 1.5

Figure 4: Percent not Bioinequivalent for r = .8

Figure 5: Percent not Bioinequivalent for r = 1.2

R Code

Functions

```
##########################################################
##########################################################
############ Calculate parameters:

Cmax = function(dataA, dataB, n, time){
  CmaxA = apply(dataA, 1, max)
  CmaxB = apply(dataB, 1, max)
  return(list(CmaxA, CmaxB))
}

AUC = function(FormA, FormB, n, time){
  w = length(time)
  AUCA = 0
  AUCB = 0

  for (i in 2:w){
    AUCA = AUCA + (time[i] - time[i - 1])*(FormA[,i] + FormA[,i- 1])/2
  }

  for (i in 2:w){
    AUCB = AUCB + (time[i] - time[i - 1])*(FormB[,i] + FormB[,i- 1])/2
  }

  return(list(AUCA, AUCB))
}

AUC2 = function(FormA, FormB, n, time){

  w = length(time)
  p = 4

  AUC2A = 0
  AUC2B = 0

  for (i in (p + 1):w){
    AUC2A = AUC2A + (time[i] - time[i - 1])*(FormA[,i] + FormA[,i - 1])/2
  }

  for (i in (p + 1):w){
    AUC2B = AUC2B + (time[i] - time[i - 1])*(FormB[,i] + FormB[,i - 1])/2
  }

  return(list(AUC2A, AUC2B))
}

ek = function(FormA, FormB, n, time){

   p = 4

  time1 = time[-c(1:p-1)]
  Form1A = FormA[,-c(1:p-1)]
  Form1B = FormB[,-c(1:p-1)]

  kA = rep(0, n)
  kB = rep(0, n)
  for (i in 1:n){
    kA[i] = -cov(log(Form1A[i,]), time1)/sd(time1)^2
  }

  for (i in 1:n){
    kB[i] = -cov(log(Form1B[i,]), time1)/sd(time1)^2
  }

  return(list(exp(kA), exp(kB)))
}

CO = function(FormA, FormB, n, time){

  p = 4

  time1 = time[-c(1:p-1)]
  Form1A = FormA[,-c(1:p-1)]
  Form1B = FormB[,-c(1:p-1)]

  kA = rep(0, n)
  kB = rep(0, n)
  #for (i in 1:n){
  kA = apply(Form1A, 1, function(x) -cov(log(x), time1)/sd(time1)^2)
  #}

  #for (i in 1:n){
  kB = apply(Form1B, 1, function(x) -cov(log(x), time1)/sd(time1)^2)
  #}

  CA = rep(0, n)

  #for (i in 1:n){
  CA = apply(Form1A, 1, function(x) mean(log(x))) + kA*mean(time1 - time1[1])
  #}
```

```
  COA = exp(CA)

  CB = rep(0, n)

  # for (i in 1:n){
  CB = apply(Form1B, 1, function(x) mean(log(x))) + kB*mean(time1 - time1[1])
  #}

  COB = exp(CB)

  return(list(COA, COB))
}

############################################################
############################################################
############ Calculate ratios:

medianratio = function(storeA, storeB){
  return(median(storeA/storeB))
}

ratiomeans = function(storeA, storeB){
  return(mean(storeA)/mean(storeB))
}

biasratio = function(storeA, storeB){

  xbar = mean(storeA)
  ybar = mean(storeB)
  sy = sd(storeB)
  sxy = cov(storeA, storeB)

  return (xbar/ybar - 1/50*(xbar*sy^2/ybar^3 - sxy/ybar^2))
}

lognormalratio = function(storeA, storeB){
  u = log(storeA)
  ubar = mean(u)
  v = log(storeB)
  vbar = mean(v)
  su = sd(u)
  sv = sd(v)

  return (exp(ubar - vbar + su^2/2 - sv^2/2))
}

meanlogratio = function(storeA, storeB){
  return(mean(log(storeA/storeB)))
}

############################################################
############################################################
############ Formulas (For "Formulas Column"):

# Function returns a 90% CI for median ratio

formula_medratio = function(data1, data2, n){
  vecratio = sort(data1/data2)
  k = 0.5*n - 1.645*sqrt(n*.5*.5)
  return(c(vecratio[k], vecratio[ceiling(n - k + 1)]))
}

# Function returns a 90% CI for the ratio of means

formula_ratiomeans = function(data1, data2, n){

  z = 1.645

  xbar = mean(data1)
  ybar = mean(data2)

  sx = sd(data1)
  sy = sd(data2)

  sxy = cov(data1, data2)

  lower = xbar/ybar - z/sqrt(n)*sqrt(sx^2/ybar^2 + xbar^2*sy^2/ybar^4 - 2*xbar*sxy/ybar^3)
  upper = xbar/ybar + z/sqrt(n)*sqrt(sx^2/ybar^2 + xbar^2*sy^2/ybar^4 - 2*xbar*sxy/ybar^3)

  return(c(lower, upper))
}

# Function returns a 90% CI for the bias-corrected ratio

formula_biasratio = function(data1, data2, n){

  z = 1.645

  xbar = mean(data1)
  ybar = mean(data2)


  sx = sd(data1)
  sy = sd(data2)

  sxy = cov(data1, data2)
```

```
  lower = xbar/ybar - 1/n*(xbar*sy^2/ybar^3 - sxy/ybar^2) - z/sqrt(n)*sqrt(sx^2/ybar^2 + xbar^2*sy^2/ybar^4 - 2*xbar*sxy/ybar^3)
  upper = xbar/ybar - 1/n*(xbar*sy^2/ybar^3 - sxy/ybar^2) + z/sqrt(n)*sqrt(sx^2/ybar^2 + xbar^2*sy^2/ybar^4 - 2*xbar*sxy/ybar^3)

  return(c(lower, upper))
}

# Function returns a 90% CI for the lognormal ratio

formula_ratiologmeans = function(data1, data2, n){
  z = 1.645

  u = log(data1)
  v = log(data2)

  ubar = mean(u)
  vbar = mean(v)

  su = sd(u)
  sv = sd(v)

  suv = cov(u, v)

  lower = exp(ubar - vbar + su^2/2 - sv^2/2 - z*sqrt(su^2/n + sv^2/n - 2*suv/n + su^4/(2*n) + sv^4/(2*n)))
  upper = exp(ubar - vbar + su^2/2 - sv^2/2 + z*sqrt(su^2/n + sv^2/n - 2*suv/n + su^4/(2*n) + sv^4/(2*n)))

  return(c(lower, upper))
}

# Function returns a 90% CI for the mean of log ratios

formula_logratios = function(data1, data2, n){

  data1 = log(data1)
  data2 = log(data2)

  D = data1 - data2
  Dbar = mean(D)
  std = sd(D)

  t = qt(.95, n - 1)

  lower = Dbar - t*std/sqrt(n)
  upper = Dbar + t*std/sqrt(n)

  return(c(lower, upper))
}

# Function takes computes all the different formula confidence intervals when given
# formulation A and formulation B and the sample size:

formula_intervalcalc = function(FormA, FormB, n){
  return(matrix(c(formula_medratio(FormA, FormB, n), formula_ratiomeans(FormA, FormB, n),
                  formula_biasratio(FormA, FormB, n), formula_ratiologmeans(FormA, FormB, n),
                  formula_logratios(FormA, FormB, n)), nrow = 5, ncol = 2, byrow = TRUE))
}

############################################################
############################################################
############# Bootstrap 1:

# Function generates a bootstrap sample from the data:

shuffledat = function(dataA, dataB, n){

  tempA = matrix(0, nrow = n, ncol = 8)
  tempB = matrix(0, nrow = n, ncol = 8)

  for (i in 1:n){

    a = floor(runif(1, 1, n + 1))

    tempA[i, ] = dataA[a, ]
    tempB[i, ] = dataB[a, ]
  }
  return(list(tempA, tempB))
}

# Function returns a 90% CI for bootstrap 1

bootint1 = function(dataA, dataB, n, time, fun, ratio){

  stuff = fun(dataA, dataB, n, time)

  Ra = ratio(stuff[[1]], stuff[[2]])
  R = c()

  for(r in 1:1000){
    k = length(time)

    temp = shuffledat(dataA, dataB, n)

    matA = temp[[1]]
    matB = temp[[2]]

    a = fun(matA, matB, n, time)
```

```
    storeA = a[[1]]
    storeB = a[[2]]


    R[r] = ratio(storeA, storeB)


  }
  R = sort(R)
  return(c(R[50], R[950]))
}

############################################################
############################################################
############# Bootstrap 2:

bootint2 = function(dataA, dataB, n, time, fun, ratio){

  stuff = fun(dataA, dataB, n, time)

  Ra = ratio(stuff[[1]], stuff[[2]])
  R = c()

  for(r in 1:1000){
    k = length(time)

    temp = shuffledat(dataA, dataB, n)

    matA = temp[[1]]
    matB = temp[[2]]

    a = fun(matA, matB, n, time)

    storeA = a[[1]]
    storeB = a[[2]]

    R[r] = ratio(storeA, storeB)

  }
  R = sort(R)
  return(c(2*Ra - R[950], 2*Ra - R[50]))
}

############################################################
############################################################
############# Permutation:

# Function shuffles data sets and returns two shuffled sets of data:

shuffledat2 = function(dataA, dataB, n){
    matA = matrix(0, nrow = n, ncol = 8)
    matB = matrix(0, nrow = n, ncol = 8)

    for (i in 1:n){

      a = runif(8)
      for (j in 1:8){
        if(a[j] > 0.5){
          matA[i, j] = dataB[i, j]
          matB[i, j] = dataA[i, j]
        }

        else{
          matA[i, j] = dataA[i, j]
          matB[i, j] = dataB[i, j]
        }
      }
    }
    return(list(matA, matB))
}

# Function returns a 90 % CI for bootstrap 2

permutation = function(dataA, dataB, n, time, fun, ratio){

  stuff = fun(dataA, dataB, n, time)
  Ra = ratio(stuff[[1]], stuff[[2]])
  if(identical(ratio, meanlogratio)){
    Ra = exp(Ra)
  }
  R = c()

  for(r in 1:1000){
    k = length(time)

    temp = shuffledat2(dataA, dataB, n)

    matA = temp[[1]]
    matB = temp[[2]]

    a = fun(matA, matB, n, time)

    storeA = a[[1]]
    storeB = a[[2]]
```

```
    if (identical(ratio, meanlogratio)){
      R[r] = exp(mean(log(storeA) - log(storeB)))
    }
    else{
      R[r] = ratio(storeA, storeB)
    }

  }
  R = sort(R)

  if(identical(ratio, meanlogratio)){
    return(c(log(Ra/R[950]), log(Ra/R[50])))
  }
  else{
    return(c(Ra/R[950], Ra/R[50]))
  }
}
```

Project Code

```
library("Hmisc")
require(MASS)

set.seed(Sys.time())

dataA = read.table("http://inside.mines.edu/~wnavidi/math482/tretinoinA", header = TRUE)
dataA = as.matrix(dataA)
dataA[dataA == 0] = 3
dataA = log(dataA)

dataB = read.table("http://inside.mines.edu/~wnavidi/math482/tretinoinB", header = TRUE)
dataB = as.matrix(dataB)
dataB[dataB == 0] = 3
dataB = log(dataB)

require(expm)

timea = c(0.25, 0.5, 1, 1.5, 4.5, 7.5, 10.5, 13.5)
timeb = c(0.1, 0.35, 0.75, 1.25, 3.0, 6.0, 9.0, 12.0)
timec = c(0.35, 0.75, 1.25, 3.0, 6.0, 9.0, 12.0, 15.0)
timed = c(1.5, 1.5, 1.5, 1.5, 10.5, 10.5, 10.5, 10.5)

time = timea

######################################################################
######################################################################
# Find means for formulations A and B

meansA = c()

for(i in 1:length(dataA[1,])){
  if (any(timea == time[i])){
    t = which(timea %in% time[i])
    meansA[i] = mean(dataA[,t])
  }
  else if(time[i] < timea[1]){
    meansA[i] = approxExtrap(c(timea[1], timea[2]), c(mean(dataA[,1]), mean(dataA[,2])), time[i])$y
  }
  else if(time[i] > timea[8]){
    meansA[i] = approxExtrap(c(timea[7], timea[8]), c(mean(dataA[,7]), mean(dataA[,8])), time[i])$y
  }
  else{
    hight = time[time[i] < timea]
    lowt = time[time[i] > timea]
    a = which(time %in% min(hight))
    b = which(time %in% max(lowt))
    meansA[i] = mean(c(mean(dataA[,a]), mean(dataA[,b])))
  }
}

r = 1.5 #Take the ratio to be one
r2 = 1.5

meansB = c()
meansB[1:4] = meansA[1:4] + log(r)
for(t in 5:8){
  meansB[t] = meansA[4] - r2*(meansA[4] - meansA[t]) + log(r)
}

# Find variances for formulations A and B

varianceA = c()
varianceB = c()

for(i in 1:length(dataA[1,])){
  if (any(timea == time[i])){
    t = which(timea %in% time[i])
    varianceA[i] = var(dataA[,t])
    varianceB[i] = var(dataB[,t])
  }
  else if(time[i] < timea[1]){
    varianceA[i] = var(dataA[,1])
    varianceB[i] = var(dataB[,1])
  }
  else if(time[i] > timea[8]){
    varianceA[i] = var(dataA[,8])
    varianceB[i] = var(dataB[,8])
  }
  else{
    hight = time[time[i] < timea]
    lowt = time[time[i] > timea]
    a = which(time %in% min(hight))
    b = which(time %in% max(lowt))
    varianceA[i] = mean(c(var(dataA[,a]), var(dataA[,b])))
    varianceB[i] = mean(c(var(dataB[,a]), var(dataB[,b])))
  }
}

######################################################################
######################################################################

l = 0.4 #The correlation between any pair of log measurements
n = 50 #The number of subjects

# Compute the covariance matrix of the log of the tretinoin data
```

```
variance = c(varianceA, varianceB)

covmat = matrix(0, nrow = 16, ncol = 16)
for(i in 1:16){
  for(j in 1:16){
    if(i == j){
      covmat[i, j] = variance[i]
    }
    else{
      covmat[i, j] = l*sqrt(variance[i])*sqrt(variance[j])
    }
  }
}

######################################################################
######################################################################

#################### For normal data
newmat =  mvrnorm(50, rep(0, 16), covmat)

#################### For skewed data
# newmat = mvrnorm(50, rep(0, 16), diag(1, nrow = 16, ncol = 16))
# newmat = newmat^2
# newmat = 0.5*newmat - 0.5
# newmat = newmat%*%sqrtm(covmat)

# Add the true mean for formulation A to the first 8 columns and the true mean for
# formulation B to the second 8 columns

means = c(meansA, meansB)

for(i in 1:n){
  newmat[i,] = newmat[i,] + means
}

newmat = exp(newmat)

# Estimate Cmax for each subject

FormA = newmat[,1:8]
FormB = newmat[,9:16]

# Make vectors os functions

formulas = c(formula_medratio, formula_ratiomeans, formula_biasratio, formula_ratiologmeans, formula_logratios)
ratio = c(medianratio, ratiomeans, biasratio, lognormalratio, meanlogratio)
params = c(Cmax, AUC, AUC2, ek, C0)
methods = c(bootint1, bootint2, permutation, formulas)


#############################################################
######################## Start Below! ######################
#############################################################

bioequivalent = array(0, c(1000, 4, 8))
bioinequivalent = array(0, c(1000, 4, 8))
neither = array(0, c(1000, 4, 8))

for(h in 1:100){

  newmat =  mvrnorm(50, rep(0, 16), covmat)

  #################### Below is for skewed data
  # newmat = mvrnorm(50, rep(0, 16), diag(1, nrow = 16, ncol = 16))
  # newmat = newmat^2
  # newmat = 0.5*newmat - 0.5
  # newmat = newmat%*%sqrtm(covmat)

  means = c(meansA, meansB)

  for(i in 1:n){
    newmat[i,] = newmat[i,] + means
  }
  newmat = exp(newmat)

  FormA = newmat[,1:8]
  FormB = newmat[,9:16]

  for (p in 1:8){


    if(p == 1){
    j = 2
    k = 1
    }
    else if (p == 2){
    j = 2
    k = 5
    }
    else if (p == 3){
    j = 5
    k = 5
    }
    else if (p == 4){
    j = 1
```

```
    k = 5
    }
    else if (p == 5){
    j = 1
    k = 1
    }
    else if (p == 6){
    j = 5
    k = 1
    }
    else if (p == 7){
    j = 5
    k = 4
    }
    else if (p == 8){
    j = 5
    k = 2
    }


  for (i in 1:4){

        if (i == 4){

          A = params[[j]](FormA, FormB, 50, time)[[1]]
          B = params[[j]](FormA, FormB, 50, time)[[2]]

          if (identical(ratio[[k]], meanlogratio)){
            if(exp(formulas[[k]](A, B, 50)[1]) > 0.8 & exp(formulas[[k]](A, B, 50)[2]) < 1.25){
              bioequivalent[h, i, p] = bioequivalent[h, i, p] + 1}
            else if(exp(formulas[[k]](A, B, 50)[1]) > 1.25 || exp(formulas[[k]](A, B, 50)[2]) < 0.8){
              bioinequivalent[h, i, p] = bioinequivalent[h, i, p] + 1}
            else{
              neither[h, i, p] = neither[h, i, p] + 1}
          }
          else {
            if(formulas[[k]](A, B, 50)[1] > 0.8 & formulas[[k]](A, B, 50)[2] < 1.25){
              bioequivalent[h, i, p] = bioequivalent[h, i, p] + 1}
            else if(formulas[[k]](A, B, 50)[1] > 1.25 || formulas[[k]](A, B, 50)[2] < 0.8){
              bioinequivalent[h, i, p] = bioinequivalent[h, i, p] + 1}
            else{
              neither[h, i, p] = neither[h, i, p] + 1}
          }
        }
        else if (identical(ratio[[k]], meanlogratio)){
          if(exp(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[1]) > 0.8 & exp(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[2]) < 1.25){
            bioequivalent[h, i, p] = bioequivalent[h, i, p] + 1}
          else if(exp(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[1]) > 1.25 || exp(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[2]) < 0.8){
            bioinequivalent[h, i, p] = bioinequivalent[h, i, p] + 1}
          else{
            neither[h, i, p] = neither[h, i, p] + 1}
        }
        else {
          if(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[1] > 0.8 & methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[2] < 1.25){
            bioequivalent[h, i, p] = bioequivalent[h, i, p] + 1}
          else if(methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[1] > 1.25 || methods[[i]](FormA, FormB, 50, time, params[[j]], ratio[[k]])[2] < 0.8){
            bioinequivalent[h, i, p] = bioinequivalent[h, i, p] + 1}
          else{
            neither[h, i, p] = neither[h, i, p] + 1}
        }
        print(bioequivalent[c(1:20),,])
      }
  }
}

table1 = array(0, c(8,12)) # stores all bioeq/bioin/neither percents for 8 individual ratios
table2 = array(0, c(12,12)) # stores all bioeq/bioin/neither percents for 12 ratio combos


for (w in 3:8) {
  for (x in 1:4) {
    for (y in 1:100) {
      if (bioequivalent[y,x,w] == 1 && bioequivalent[y,x,1] == 1) {
        table2[w-2,(x*3)-2] = table2[w-2,(x*3)-2] + 1
      }
      if (bioequivalent[y,x,w] == 1 && bioequivalent[y,x,2] == 1) {
        table2[w+4,(x*3)-2] = table2[w+4,(x*3)-2] + 1
      }
      if (bioinequivalent[y,x,w] == 1 || bioinequivalent[y,x,1] == 1) {
        table2[w-2,(x*3)-1] = table2[w-2,(x*3)-1] + 1
      }
      if (bioinequivalent[y,x,w] == 1 || bioinequivalent[y,x,2] == 1) {
        table2[w+4,(x*3)-1] = table2[w+4,(x*3)-1] + 1
      }
      if ((neither[y,x,w] == 1 && bioequivalent[y,x,1] == 1) || (bioequivalent[y,x,w] == 1 && neither[y,x,1] == 1 ) || (neither[y,x,w] == 1 && neither[y,x,1] == 1 )) {
        table2[w-2,(x*3)] = table2[w-2,(x*3)] + 1
      }
      if ((neither[y,x,w] == 1 && bioequivalent[y,x,2] == 1) || (bioequivalent[y,x,w] == 1 && neither[y,x,2] == 1 ) || (neither[y,x,w] == 1 && neither[y,x,2] == 1 )) {
        table2[w+4,(x*3)] = table2[w+4,(x*3)] + 1
      }
    }
  }
}
table2 = table2/100
df2 = data.frame(table2)
colnames(df2) <- c("Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither")
```

```
rownames(df2) <- c("AUC median ratio/C(1.5) mean of log","AUC median ratio/Cmax mean of log","AUC median ratio/Cmax median ratio","AUC median ratio/C(1.5) median ratio","AUC m

# Sum all rows and put into table1 for 8 individual ratios
for (w in 1:8) {
  for (x in 1:4) {
    for (y in 1:100) {
      if (bioequivalent[y,x,w] == 1) {
        table1[w,(x*3)-2] = table1[w,(x*3)-2] + 1
      }
      if (bioinequivalent[y,x,w] == 1) {
        table1[w,(x*3)-1] = table1[w,(x*3)-1] + 1
      }
      if (neither[y,x,w] == 1) {
        table1[w,(x*3)] = table1[w,(x*3)] + 1
      }
    }
  }
}
table1 = table1/100
df1 = data.frame(table1)
colnames(df1) <- c("Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither","Bioeq","Bioineq","Neither")
rownames(df1) <- c("AUC median ratio","AUC mean of log","C(1.5) mean of log","Cmax mean of log","Cmax median ratio","C(1.5) median ratio","C(1.5) log normal","C(1.5) ratio of
```