Phylogenetic Trees Python Code User Manual: RCC and RCC1

by
Wesley Johnston

Last Updated
14 July 2020

Contents

Section 1

About Phylogenetic Trees and the Python Code

Bill Howard

The late Astronomer William E. Howard III developed and freely shared his method for generating phylogenetic trees from Y-DNA STR data. He published his first two papers in 2009 in the *Journal of Genetic Genealogy*:

- "The Use of Correlation Techniques for the Analysis of Pairs of Y-STR Haplotypes, Part 1: Rationale, Methodology and Genealogy Time Scale"
- "The Use of Correlation Techniques for the Analysis of Pairs of Y-STR Haplotypes, Part 2: Application to Surname and Other Haplotype Clusters".[1]

He went on to publish many more papers, with various co-authors on some of them. He sought to apply his method as much as possible, exploring the potential and the limits of his method. The subtitle of his 2014 paper "Using Y-DNA Haplotypes to Estimate Their Dates of Origin: Pitfalls and Prospects" reflects this well. He was very willing to share his work, carefully explaining precisely how the method work so that anyone wishing to follow in his footsteps could replicate his method.

Bill Howard passed away 25 July 2016. The state of the art of Y-DNA has changed greatly. Family Tree DNA now has the Big Y-700 test and uses SNPs to cluster kits into buckets on the Big Y Block Tree. This recent focus on SNPs has led to some people discounting the value of STRs in Y-DNA genetic genealogy. But STRs give a level of granularity in recent branching that SNPs cannot give. Both are needed.

Bill Howard's phylogenetic trees provide a track record of many applications over many years. Although limitations appeared in some applications, it works very well in other situations.

My own sense is that phylogenetic trees work well with kits that cluster in the same bucket (or even adjacent buckets) at the lowest level of the Big Y Block Tree. In those cases, phylogenetic trees provide a time-gradient DNA-based estimate of when branches separated from each other, a level of specificity that the SNPs cannot provide. For surname projects, it may be that only certain

[1] All of Bill's papers and other sharing that he did that I have been able to find are gathered on the web page at http://www.wwjohnston.net/famhist/bill-howard-ydna.htm

subsets of the kits really belong together. For haplogroup projects, my sense is that the method may not work for that broad a population of kits.

RCC (Revised Correlation Coefficient)

Bill's fundamental measure for comparing pairs of kits to generate the phylogenetic trees is what he called the Revised Correlation Coefficient (RCC):

RCC = 10000 * ((1/CC) -1)

where CC is the Pearson Correlation Coefficient of the values of the Y-STR markers in a specific pair of kits. He builds a matrix of all possible pairs of kits, in which the value of each cell in the matrix is the RCC value for that pair, with 0 for a perfect match and higher values meaning higher distance from each other.

The Python Code

The Johnstons of Annandale project, administered by Clifford Johnston, has relied on Bill Howard's phylogenetic trees for many years. This is a very specific subset of the Johnstons. The kits cluster nicely in adjacent buckets of the Big Y Block Tree, so that the SNPs align well with the STRs. The STRs can then take the analysis the next step beyond what the SNPs can show.

Bill Howard began his work using Excel spreadsheet formulas. He eventually implemented it in R code, run in MatLab. After Bill's death, some in the Johnston project grew concerned that Bill's method should remain available to the project. Since Bill was very transparent in his papers, in 2018 Wesley Johnston and George El Zakhem in the United States and David Johnston in Australia began developing code to replicate Bill Howard's method. George El Zakhem developed the first implementation in Mathematica. David Johnston in Australia then developed a robust Python implementation, which became the core program. George El Zakhem then worked with that to make the first runs on the combined kits of the Eastlick and Lake projects, which Wesley Johnston then augmented. The code now available thus represents the culmination of the work of Bill Howard, David Johnston, George El Zakhem and Wesley Johnston.

The RCC1 Version

During work on the Lake DNA kits, Wesley Johnston discovered that the phylogenetic tree placed one kit on a branch that conflicted with where Big Y SNP results, autosomal results and documented family trees placed it. Examination of the markers for that kit revealed that one marker had a 3-step mutation. Because the RCC method relies on the correlation coefficient of actual marker values in pairs of kits, this 3-step mutation had caused the misplacement of the kit in the tree.

After considering alternatives of how to handle this situation, Wesley relied on developing a method, which he called RCC1, analogous to how simple genetic distance is calculated. If a marker's value differs by 3 for two people, stepwise genetic distance counts that as a distance of 3, but simple genetic distance counts differences of any size as 1.

In work with other family project data, Wesley realized that RCC1 mutes the impact of any multi-step mutations. This meant that the person running the program could use the same data used in the RCC program and not have to deal with discovering and finding a workaround for specific markers.

As of this writing, Wesley has run the RCC1 version for two different family projects. In both cases, the RCC1-generated phylogenetic tree more accurately reflected what other DNA evidence and documents had found as where the kits belonged in relation to each other.

The details of running the two versions are identical. Sections 2-5 apply to both versions, and Section 6 has the specifics about the RCC1 version.

Wesley's **standard procedure now includes running the same data through both the RCC and the RCC1 versions**. He then examines the generated trees to see whether they place specific kits differently. Such likely have multi-step mutations in at least one marker.

Section 2

Input Data

Data Format

The input data comes into the program in the form of a CSV (comma-separated variables) file. The format is very simple.

Rows

- The first row is the header row.
- The remaining rows each represent a single Y-DNA test kit's results.

Columns

- The left column is the name you want to appear on the phylogenetic tree for that kit. For the first row, enter the label "Name". If you do not want any name to appear, it is best to enter a period in each subsequent row. Otherwise, Python will default to a text string that it uses for missing values.
- The second column from the left is the kit number column. For the first row, enter the label "Kit". Each subsequent row should have the kit number so that you can clearly identify that kit on the phylogenetic tree.
- The third and subsequent columns are for the values (alleles) of the STR markers. The first row has the labels for the markers.

Multi-valued Markers

Family Tree DNA presents the multi-valued markers as a single marker with multiple values. For example, the value of DYS385 may appear as 11-14, or DYS464 may appear as 13-15-16-17.[2]

When the results are exported from Family Tree DNA or copied from the project results screen into a spreadsheet, one of two things can happen. If the receiving fields in the spreadsheet are pre-defined as Text, then the values will appear as they do in Family Tree DNA. Otherwise, the values that can be considered dates, will show either as the date or will be converted to the internal number that the spreadsheet uses to represent a date.

---

[2] Bill Howard addressed what he called multi-copy markers on page 27 of his 25 February 2014 paper "Uniting the Time Scales of Genealogy and Genetics Using Correlation Techniques to Explore Y-DNA". He studied permutations of the multi-valued markers and concluded that "the errors introduced through the use of multi-copy markers will be minimal or non-existent."

For example, the value 11-14 will appear correctly in a pre-defined Text cell. But it will be converted into the date 14 November of the current year, so that in 2020 the cell will show either the date ("14-Nov") or the internal value ("44149").

The format needed for input to the Python code requires that all multi-valued markers be separated into individual columns. So, you have to go through the imported kits in your spreadsheet and separate them: the one column with "11-14" must become two columns, one with "11" and the other with "14". This must be done for all mutli-valued markers.

Extra Values in Multi-valued Markers

On rare occasions, a multi-valued marker that normally has only two values will appear with three values. For example, CDY may normally show "37-38" or "35-40", but it may sometimes show "34-35-36". I believe that in one of his papers, Bill advised to simply take the first two values in such a case (i.e. "34-35"). However, this is a subjective choice, and my own preference is to take the extreme values, thus "34-36" in the case of "34-35-36". I have not conducted any study of this, and I do not know that Bill did. So, this is an area open for research. But these cases are so rare that it has little impact in most surname-specific projects.

Example Data Input

Here is how the data will look in a spreadsheet:

	A	B	C	D	E	F	G	H	I	
1	**Name**	**Kit**	**DYS393**	**DYS390**	**DYS19**	**DYS391**	**DYS385a**	**DYS385b**	**DYS426**	**DY**
2	Lake	L25404	13	24	14	11	11	14	12	
3	Lake	L25424	13	24	14	12	11	14	12	
4	Lake	L36265	13	25	14	11	11	15	12	

Here is how the same data will look in the CSV file (with the rows truncated to correspond to the spreadsheet view):

```
Name,Kit,DYS393,DYS390,DYS19,DYS391,DYS385a,DYS385b,DYS426,DY
Lake,L25404,13,24,14,11,11,14,12,
Lake,L25424,13,24,14,12,11,14,12,
Lake,L36265,13,25,14,11,11,15,12,
```

Input Gotcha

One error that sometimes shows up is when you have blank rows in the spreadsheet after the data rows. This will lead to an input error when you run the program. In that case, delete the extra blank rows in your input data file.

Proper Grouping of Input Kits

The input data must have the same number of columns for every row. Thus, you cannot run a 12-marker test in a run with 37-marker kits. The calculation of the correlation coefficient, the fundamental calculation in Bill Howard's method, will fail if every kit does not have the same number of markers.

If you wanted to run 37-marker kits with 67 and 111-marker kits, you would need to truncate the results of the 67 and 111-marker kits to their first 37 markers, so that all rows have exactly 37 markers.

Here is a note that Bill Howard posted 30 Nov 2013 on the Rootsweb Genealogy-DNA e-mail list:[3]

"One last point, please try to make the haplotype set as pure as possible. By that I mean, if it is an M222 SNP, be sure all haplotypes have been SNP-tested. If it is a surname set, I would prefer that they all are in the same haplogroup (otherwise the tree gets unwieldy, long, and will lack time resolution in regions of genealogical interest). The larger the set, the better the optimization process will be when the tree is produced …"

See Appendix 4 "Convergence, Reverse and Parallel Mutations" for more information.

Ordering of Input Kits

Tests run on various orderings of the sequence in which the kits were presented to the program all resulted in the same phylogenetic tree output. So there does not appear to be any bias in the clustering from the order in which the kits are presented. See the Appendices for more on this.

Entering the File Name in the Program

Enter the file name of your input CSV file in single quotes as the value of the variable I in the first line of code after the importing of the external functions. So, if your CSV file is "37-marker-kits.csv", then after entering the file name as the value of I, the top lines of code in the program will look like this:

```
from scipy.stats import pearsonr
from scipy.cluster.hierarchy import dendrogram, linkage
#from pprint import pprint
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist
from matplotlib import pyplot as plt
import math
```

[3] Rootsweb terminated support for e-mail lists in 2020. The Genealogy-DNA list successor for the broadest group of former Rootsweb members is now at https://groups.io/g/genealogy-dna

```
I = 37-marker-kits'
#Specify the file name of the CSV file.
# This will also become the title of the phylogenetic tree and the name of the output
file with the tree.
```

As noted in the comment, the title of the generated phylogenetic tree will have this same name, and so will the file that contains the image of the tree.

Section 3

Setting the Control Parameters

Rscale

Rscale is the ratio of years per RCC (or RCC1 in the RCC1 version). Rscale is used to calculate the year in which the phylogenetic tree has a split of two branches from each other.

Calibrating the value of Rscale (years/RCC or years/RCC1 ratio)

Ideally, the value should be calibrated from your own data. To do so, you need one or more pairs of kits for whom you know their common ancestor. You then calculate the years from the birth of the ancestor to the birth of the test taker for each person in the pair and then divide that by the RCC (or RCC1) value of the pair. Then use the average of the individual ratios as the value of Rscale.

It is important to not use kits that are too close to each other. This is because they will have a small RCC (or RCC1), which will lead to a large years/RCC ratio (or years/RCC1). So how close is too close? There is no definitive answer. I start by ruling out of the calibration pairs that have RCC (or RCC1) of 5 or less. But when you see several pairs generating years/RCC (or years/RCC1)ratios around 35 or 40 when another pair generates a ratio of 114, then that pair at 114 is almost certainly too close to include in the calibration.

The calibration must be made for each level of markers. The years/RCC (or years/RCC1) ratio for a set of kits at 37 markers will almost certainly not be the same as the years/RCC (years/RCC1) ratio for the same kits at 111 markers.[4]

It may also be worth calibrating pairs of kits with a distant Most Recent Common Ancestor (MRCA) separately from those who descend from a more recent MRCA and seeing how the two different ratios estimate the dates for the different parts of the tree.

What If Calibration is Impossible?

If you have no pairs of sufficiently distant kits for whom the common ancestor is known, then you cannot do the calibration calculation. In this case, start with a value of 40 for 111 markers, 37 for 67 markers and 34 for 37 markers. These will probably place you in the right ballpark. But keep trying to find a pair of kits that you can use to do your own calibration.

[4] Bill Howard did a lot of research on the calibration, as he worked on different applications of his method to specific families. Several of his papers have important information about these experiences.

Byear

Byear is the approximate birth year of present-day test takers to use in the calculation of the years in which branch splits happened. So, you need at least a rough idea of the birth years of most of your test takers. Setting Byear to 1950 may be about right if there is a mixture of older and younger test takers.

Here is the formula used to calculate the year of a branch split from the values of Rscale, Byear and RCC (or RCC1 in the RCC1 version).

Split year = 10 * INT([Byear – INT(RCC * Rscale)]/10)

where INT is the INTEGER function that eliminates all decimal places (e.g. INT(3.141) = 3).

So if Rscale is 38 and Byear is 1950 and the branch split happened at RCC =7.27, then the calculation goes as follows:

RCC * Rscale = 38 * 7.27 = 276.26
INT(RCC * Rscale) = INT(276.26) = 276
Byear – INT(RCC * Rscale) = 1950 – 276 = 1674
(Byear – INT(RCC * Rscale))/10 = 1674/10 = 167.4
INT[(Byear – INT(RCC * Rscale))/10] = INT(167.4) = 167
10 * INT[(Byear – INT(RCC * Rscale))/10] = 10 * 167 = 1670

So, the branch split happened in about the year 1670. Note that the process always generates a split year estimate truncated to the next lowest decade. So instead of displaying the split year as 1674, it shows it as 1670.

Setting Rscale and Byear

The Rscale and Byear variables are set immediately after the input I variable. So if you set Rscale top 34.65 and Byear to 1951, then the highlighted lines below are all that you need to do to set up a run of your data through the program.

```
from scipy.stats import pearsonr
from scipy.cluster.hierarchy import dendrogram, linkage
#from pprint import pprint
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist
from matplotlib import pyplot as plt
import math

I = 37-marker-kits'
#Specify the file name of the CSV file.
# This will also be the phylogenetic tree's title and file name.

Rscale = 34.65
Byear = 1950
```

```
# Rscale is the years/RCC ratio, calibrated to this family for the number of markers.
# Byear is the Birth Year used to compute the split years.
# Set it at about the birth year of the test takers used to calibrate Rscale.
```

Setting RCC1added (RCC1 version only)

In the RCC1 version, the RCC1added variable determines the fill value for the fabricated z1 vector and thus indirectly for the values of the fabricated z2 vector. RCC1added normally has the value 13 since that is the average value across all 111 markers in most kits. Altering the value of RCC1added probably requires calibrating the years/RCC1 ration.

RCC1added is set in the RCC1 version immediately after the Rscale and Byear variable setting lines.

Setting RCC1markers (RCC1 version only)

In the RCC1 version, fabricated vectors z1 and z2 must be set to the correct number of markers. So, if the input kits are 67-marker results, set RCC1markers to 67. If the input kits are 111-marker results, set RCC1markers to 111.

RCC1markers is set in the RCC1 version immediately after the Rscale and Byear variable setting lines.

Section 4

Managing the Output

Overview

Family Tree DNA measures genetic distance (GD) stepwise, so that a difference in value of 2 between two kits counts as 2 in the genetic distance (as opposed to a difference-only genetic distance measure that would count such a difference as 1). Family Tree DNA also uses thresholds at each level of markers to display matches.[5] If another kit differs from your kit by more than the threshold, Family Tree DNA does not display that kit in your match list.

Bill Howard's method works most accurately when the kits included have reasonably close genetic distances. If all pairs of kits in your input data have genetic distances of 10 or less at 111 markers, 7 or less at 67 markers, or 5 or less at 37 markers, then your kits likely fall within the limits of applicability for Bill Howard's method.

This is important because the version of the Python code in the appendix expects such pairings. And the formatting of the output, the phylogenetic tree thus works well for such pairings. You will probably not have to worry about managing the output, since the default version will probably work well for you.

Runs with Higher Genetic Distances

When you first begin a study, you may want to generate a phylogenetic tree with kits whose genetic distances are greater. While the generated phylogenetic tree may not be accurate as to the split years and may even have some structure problems, such a run can identify separate clusters for which you may want to follow up with normal runs that use only the kits from one of those clusters.[6]

The inclusion of kits for which the splits of branches go back 1,000 years or more cannot be handled by the default version of the Python program. You need to modify the output format parameters for such runs.

The critical line in the Python code is this as the default:

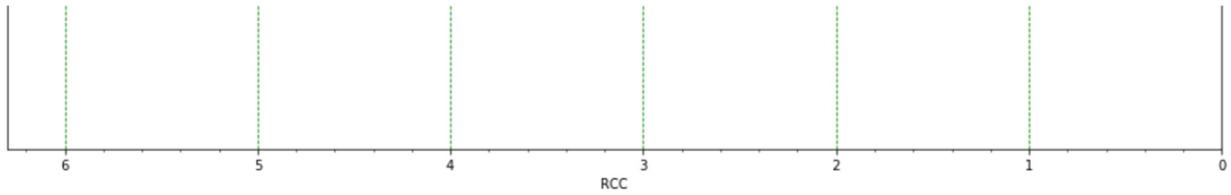
```
plt.xticks(np.arange(0,800,1))
```

[5] See https://learn.familytreedna.com/y-dna-testing/y-str/expected-relationship-match/ for the thresholds and expected levels of relationship.

[6] Beware of marker DYS481 in all your work but especially when dealing with high-GD collections of kits. DYS481 has very high DNA "stutter". If your data shows multi-step mutation on DYS481, you should probably eliminate DYS481 from your analysis. See Section 5 for more on DYS481.

This determines the X-axis (the RCC axis) vertical grid lines (ticks). It starts at zero, at the bottom right corner of the plot and then steps back 1 step at a time for up to 800 steps. The number 800 is chosen arbitrarily as a large number that is never expected to be reached. Each unit is 1 RCC.

Here is an example of the X-axis for a group of kits that go back 6 RCC:



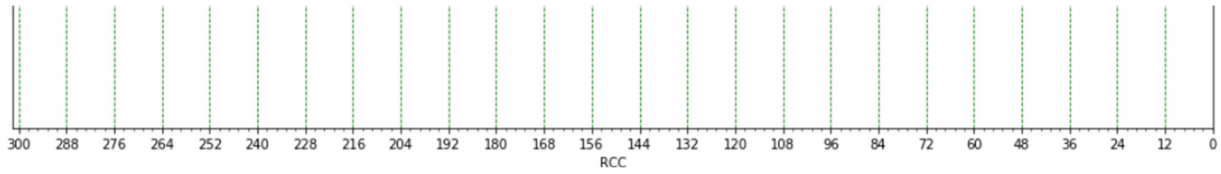
The dashed green lines are the X-axis grid lines, counting, from right to left, back in time up to 6 RCC back in time for the first branch split. When the Rscale value is 40, this is about 240 years into the past.

But when the program determines the first branch split on a collection of distant kits was at RCC = 85 or some other high value, the attempt to plot the resulting phylogenetic tree may fail because of a MAXTICKS error.

One way to deal with this is to increase the number of RCCs between grid lines, such as the following:

```
plt.xticks(np.arange(0,800,12))
```

This places the grid lines 12 RCCs apart: 0, 12, 24, etc. Here is an example of the X-axis for a run in which the first branch split happens at RCC = 287.09:



Attempting to use the default of one RCC per grid line results in an error that points to a different instruction:

```
RuntimeError                              Traceback (most recent call last)
<ipython-input-3-e1f4e3ea296d> in <module>()
    137
    138 F.subplots_adjust(left=0.05, right=0.85, top=0.97, bottom=0.05)
--> 139 plt.savefig("{input}.jupyter.png".format(input=I))
    140
```

But when you trace down to the error at the actual instruction in the called subroutine, the error is a MAXTICKS error:

```
   1470             raise RuntimeError("Locator attempting to generate {} ticks from "
   1471                                "{} to {}: exceeds Locator.MAXTICKS".format(
-> 1472                                    len(locs), locs[0], locs[-1]))
   1473         return locs
   1474
```

```
RuntimeError: Locator attempting to generate 1206 ticks from 0.2 to 301.4: exceeds Locator.MAXTICKS
```

So it is not the flagged instruction that caused the problem but the plt.xticks instruction. And the solution is to increase the value of the parameter that controls the number of RCCs between grid lines (ticks).

Section 5

Troubleshooting

Run-Time Problems

We encountered many different run-time problems, all of which traced back to our data, once the python code was well established. This section gives some specific cases.

Invalid Value Encountered in Double Scalars (GD limitation)

The error report looks like this in an Anaconda Jupyter Notebook, sometime without the boxed “invalid value” message at the top:

```
C:\ProgramData\Anaconda2\lib\site-packages\scipy\stats\stats.py:3047: RuntimeWarning: invalid value encountered in doub
le_scalars
  t_squared = r**2 * (df / ((1.0 - r) * (1.0 + r)))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-20eea2b7af14> in <module>()
     53
     54 # perform the agglomerative clustering using the average method as per Bill Howards paper
---> 55 Z = linkage(Y,method='weighted',optimal_ordering=False)
     56
     57 # perform linkage reordering such that the shorter branch is first, the longer branch second

C:\ProgramData\Anaconda2\lib\site-packages\scipy\cluster\hierarchy.pyc in linkage(y, method, metric, optimal_ordering)
   1107
   1108     if not np.all(np.isfinite(y)):
-> 1109         raise ValueError("The condensed distance matrix must contain only "
   1110                          "finite values.")
   1111

ValueError: The condensed distance matrix must contain only finite values.
```

The problem turned out to be that the stepwise genetic distances between some pairs of kits were too great. While systematic testing has not been done, limited testing found that GD of 46 or less worked, while adding just one more kit with a GD of 66 from a kit defined as the standard resulted in the error message above (without the top box).

Number of Kits (Plot readability limitation)

Test runs with as many as 216 kits worked successfully, as long as the stepwise genetic distances were within acceptable limits. The problem becomes the readability of the phylogenetic tree, since so many cases are force-fit onto a single page.

A run with 1080 within-GD-limits kits took a long time to run and then generated this error (showing just the top and bottom level errors in the error stack):

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-8-5ddf3d4e6d4f> in <module>()
    117 plt.annotate("RCC = {rscale} years".format(rscale=Rscale),(0,0),xytext=(0,-5))
    118 F.subplots_adjust(left=0.05, right=0.85, top=0.97, bottom=0.05)
--> 119 plt.savefig("{input}.jupyter.png".format(input=I))
    120
    121 # cite: http://www.jogg.info/pages/72/files/Howard.htm


C:\ProgramData\Anaconda2\lib\site-packages\matplotlib\ticker.pyc in raise_if_exceeds(self, locs)
   1470             raise RuntimeError("Locator attempting to generate {} ticks from "
   1471                                "{} to {}: exceeds Locator.MAXTICKS".format(
-> 1472                                    len(locs), locs[0], locs[-1]))
   1473         return locs
   1474

RuntimeError: Locator attempting to generate 4320 ticks from 1.0 to 10799.0: exceeds Locator.MAXTICKS


<Figure size 1152x1440 with 1 Axes>
```

The more kits in the input data, the longer the time will be to do the run. The complexity of the processing increases. As more kits are added to the dataset, the matrix calculations must then include calculation of the RCC for this new kit in relation to every kit already in the dataset.

Section 6

Special Considerations and RCC1

DNA Stutter

When the combination of Big Y, autosomal DNA and documented family history led to the expectation that one kit should align with three others in the phylogenetic tree, we sought the cause of the placement in the tree. It turned out that going back more than a decade, forensic DNA researchers had identified marker DYS481 (which is in the 38-67 marker panels and thus is in 67 marker and 111 marker tests) as having high "stutter":

"During the PCR amplification process, the polymerase can lose its place when copying a strand of DNA, usually slipping forwards or backwards four base pairs. The result is a small number of DNA fragment copies that are either one repeat larger or smaller than the true fragment being amplified."[7]

"Stutter is the most common instrumental artefact and is caused by DNA slippage during amplification. Stutter occurs in between 6-10% of amplification products."[8]

To be clear, DNA stutter happens in the laboratory and does not reflect the actual value of a person's Y-DNA. When stutter occurs in those relatively few cases, the value shown for the person for that marker is not the true value. It seems that it is impossible to know at the time of amplification that stutter has occurred. The only way to detect it seems to be in examining the results and seeing that a person has a value for a marker that does not agree with his known relatives. The only way to know the true value is to do a second test and hope that stutter does not occur again.

Because the Revised Correlation Coefficient relies on the actual values of the markers, this difference of 3 in one marker resulted in the misalignment of the kit in the phylogenetic tree. Removing DYS481 and generating a phylogenetic tree from the remaining 110 markers (and re-calibrating the years/RCC ratio) generated a tree with the kit in the alignment that all the other DNA and family tree evidence led to expect.

DYS481 does not always exhibit stutter. In the example above, all other kits had the value 22. It appears that stutter occurs in the laboratory, so that the actual value of the marker cannot be known without retesting it. Was this kit an

---

[7] http://www.bioforensics.com/dna-testing-issues

[8] https://www.phe-culturecollections.org.uk/services/celllineauthenticationservices/interpretation-of-cell-line-str-profiles-instrumental-artefacts.aspx

actual 22? Or was there some actual mutation present? It was impossible to know.

For whatever reason, the genetic genealogy literature seems unaware of the problem, most likely because it is difficult to spot. The only way we found it was when the generated phylogenetic tree placed the kit differently than all other evidence led us to expect it should be. Genetic genealogists have recently focused more on SNPs and neglected STRs, which is another reason the literature may be silent.

Forensic researchers have identified stutter in other markers. If your phylogenetic tree places a kit differently than where you expected, look at the markers and see if there is a multi-step difference (a difference of more than 1) in the value of one of that kit's markers, and then check the web to see if forensic DNA research has identified that as a DNA stutter marker.[9]

"Fast Mutating" Markers

Genetic genealogists have identified some STR markers as "fast mutating".[10] These tend to have more mutations in a period than do other markers, or they tend to mutate in multiple steps, such as a value of 24 mutating to 26. The impact of these markers on comparisons of pairs of kits is similar to cases of DNA stutter.

Genetic Distance: Stepwise and Simple

Two main ways exist to calculate genetic distance (GD) between two kits of Y-STRs. If a specific marker's value differs by 3 between the two kits, Stepwise GD counts the difference as 3. But Simple GD counts a distance of any size as 1.

Stepwise GD is thus impacted more by DNA stutter and fast mutations than is Simple GD. Simple GD has the effect of muting the impact of DNA stutter and fast mutations.

RCC1

Work with different families to generate phylogenetic trees revealed projects in which either DNA stutter or fast mutation led to phylogenetic trees that placed one or more kits on branches not supported by Big Y SNP results, autosomal DNA results and documented family trees. The misplacement in the tree led to examination of the markers in those kits.

---

[9] See Table 11A at http://www.pbso.org/qualtrax/QTDocuments/4253.PDF for other markers found by the Palm Beach County Sheriff's Office to have different levels of stutter.

[10] See https://isogg.org/wiki/Mutation_rates and https://lists.rootsweb.com/hyperkitty/list/genealogy-dna.rootsweb.com/thread/2492202/

Jumping ahead, in one family, the problem was DNA stutter in DYS481. In the other family, the problem was a 2-step mutation of CDYb (the higher value of the 2-valued marker CDY). In both cases, RCC1 received the same data as RCC but generated trees that more accurately reflected the placement expected from Big Y and autosomal DNA and documented family trees.

So, what is RCC1?

Essentially, RCC1 is to RCC what Simple GD is to Stepwise GD. RCC1 mutes the impact of multi-step mutations. The RCC1 program retains the majority of the code of the RCC version and differs only in how it computes the Revised Correlation Coefficient.

And, specifically, how does RCC1 differ from RCC?

The difference in the two methods is primarily in the routine that calculates the correlation coefficient and then the RCC or RCC1 value.

In RCC, the correlation coefficient is calculated directly from the results of the two kits' values in each marker.

```
# calculate a condensed distance matrix consisting of RCC values as per Bill Howards
paper
# the condensed distance matrix is a single dimension matrix containing
# the top triangle of a two dimensional distance matrix
# e.g. [1,2,3,4]
#      [2,1,2,3]
#      [3,2,1,2]
#      [4,3,2,1]
# becomes a condensed distance matrix Y of [2,3,4,2,3,2]
Y = []
X = []
for i1,v1 in enumerate(K):
    X1 = []
    for i2,v2 in enumerate(K):
        if i2 > i1:
            Y.append((1/pearsonr(v1,v2)[0]-1)*10000.0)
        X1.append((1/pearsonr(v1,v2)[0]-1)*10000.0)
    X.append(X1)
```

RCC1 fabricates new vectors z1 and z2, replacing the actual kits in the calculation of the correlation coefficient. To do this RCC1 uses the user-set variable **RCC1added** to populate the first vector. (I use 13 for RCC1added, since it is the average value of all 111 markers for most kits.) RCC1 also requires that the user specify in variable **RCC1markers** the number of markers in the input kits,

In the first part of the processing, RCC1 initializes vector z1 to have a value of 1 for the second marker (the one with an index value of 1 in the vector) and a value of RCC1added in all other markers.

RCC1 then compares the two actual kit (vectors v1 and v2) for each marker. If the kits have the same value for the marker, RCC1 sets that marker in vector z2 to the same value as vector z1 has for that marker (i.e. either 1 or 13, if RCC1added was set to 13). If the marker has different values in the two kits v1 and v2, then RCC1 sets the marker in vector z2 to one less than the value of the marker in vector z1 (i.e., either 0 or 12, if RCC1added was set to 13).

Thus vector z1 is always fixed, while vector v2 changes for those markers where the two kits differ in value. Here is the code for the first part of the RCC1 routine that corresponds to the routine in RCC.

```
RCC1added = 13
RCC1markers = 111
...
# calculate a condensed distance matrix consisting of RCC values as per Bill Howards
paper
# the condensed distance matrix is a single dimension matrix containing
# the top triangle of a two dimensional distance matrix
# e.g. [1,2,3,4]
#      [2,1,2,3]
#      [3,2,1,2]
#      [4,3,2,1]
# becomes a condensed distance matrix Y of [2,3,4,2,3,2]
Y = []
X = []
for i1,v1 in enumerate(K):
    X1 = []
# ---------------------------------------------------------
    z1 = np.zeros((RCC1markers,), dtype=int)
    for z1x in range(RCC1markers):
        if z1x == 1:
            z1[z1x] = 1
        else:
            z1[z1x] = RCC1added
    for i2,v2 in enumerate(K):
        z2 = np.zeros((RCC1markers,), dtype=int)
        for x in range(RCC1markers):
            if v2[x] == v1[x]:
                z2[x] = z1[x]
            else:
                z2[x] = z1[x]-1
```

RCC1 then calculates the correlation coefficient and RCC1 value of the two fabricated vectors z1 and z2 and then calculates the RCC1 value. This code is almost identical to the RCC code, except that z1 and z2 replace v1 and v2. However, tests were added to deal with the two arrays being identical, since that case has zero variance and fails in the calculation of the correlation coefficient.

```
        if i2 > i1:
            if np.ndarray.all(z1 == z2):
```

```
                Y.append(0)
            else:
                Y.append((1/pearsonr(z1,z2)[0]-1)*10000.0)
        if np.ndarray.all(v1 == v2):
            X1.append(0)
        else:
            X1.append((1/pearsonr(z1,z2)[0]-1)*10000.0)
    X.append(X1)
```

Appendix 1

RCC Python Code

```
# User Manual at http://www.wwjohnston.net/famhist/bill-howard-ydna.htm
from scipy.stats import pearsonr
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist
from matplotlib import pyplot as plt
import math

# ---------1 - SET CONTROL VARIABLES-----------------------------

I = 'yoemans-111Corrected'
#Specify the file name of the CSV file.
# This will also be the phylogenetic tree's title and file name.

Rscale = 34.65
Byear = 1950
# Rscale is the years/RCC ratio, calibrated to this family & number of markers.
# Byear is the Birth Year used to compute the split years.
# Set Byear about the birth year of the test takers.

# ---------2 - READ INPUT INTO MATRICES-----------------------------

# read the Input file
C = pd.read_csv("{input}.csv".format(input=I)).values

# extract the kits as rows of observations and columns of attributes (markers)
K = C[:,2:845]
# Note that the numbering starts with 0 and not 1 for both rows and columns.

# extract the kit numbers and names to produce a labels matrix
L = []
for ix,row in enumerate(C):
    L.append("#{ix} {kit} {name}".format(ix=ix+1,kit=row[0],name=row[1]))

# reverse the list and kits
#L = L[::-1]
#K = K[::-1]

# ---------3 - CALCULATE RCC MATRIX FOR ALL PAIRS OF KITS-------------

# calculate a condensed distance matrix consisting of RCC values as per Bill Howards
paper
# the condensed distance matrix is a single dimension matrix containing
# the top triangle of a two dimensional distance matrix
# e.g. [1,2,3,4]
#      [2,1,2,3]
#      [3,2,1,2]
```

```
#      [4,3,2,1]
# becomes a condensed distance matrix Y of [2,3,4,2,3,2]
Y = []
X = []
for i1,v1 in enumerate(K):
    X1 = []
    for i2,v2 in enumerate(K):
        if i2 > i1:
            Y.append((1/pearsonr(v1,v2)[0]-1)*10000.0)
        X1.append((1/pearsonr(v1,v2)[0]-1)*10000.0)
    X.append(X1)

# ---------4 - GENERATE THE PHYLOGENETIC TREE SPLITS-------------

# perform the agglomerative clustering using the average method as per Bill Howards
paper
Z = linkage(Y,method='weighted',optimal_ordering=False)

# perform linkage reordering such that the shorter branch is first, the longer branch
second
# the lower index is first, the higher index is second
for link in Z:
    leftDepth = link[0]
    if leftDepth >= len(L):
        leftDepth = Z[int(leftDepth)-len(L),3]
    else:
        leftDepth = 1
    rightDepth = link[1]
    if rightDepth >= len(L):
        rightDepth = Z[int(rightDepth)-len(L),3]
    else:
        rightDepth = 1
    if leftDepth < rightDepth:
        t = link[0]
        link[0] = link[1]
        link[1] = t
    elif link[0] < link[1] and link[0] < len(L) and link[1] < len(L):
            t = link[0]
            link[0] = link[1]
            link[1] = t

# ---------5 - PLOT THE PHYLOGENETIC TREE-------------

# plot the cluster hierarchy produced by linkage as a dendrogram
F = plt.figure(figsize=(16,20),dpi=72) # A1 paper
plt.title(I)
plt.xlabel("RCC")
plt.grid(True,which='major',axis='x',color='g',linestyle='dashed')
plt.minorticks_on()
plt.tick_params(axis='x',which='minor')
plt.tick_params(axis='y',which='minor',length=0)

#plt.xscale('symlog',basex=2)

plt.xticks(np.arange(0,800,12))
```

```
#plt.xticks((0,1,2,4,8,16,32,64,128,256),(0,1,2,4,8,16,32,64,128,256))
# - use this on recent kits within the last 50 RCC: plt.xticks(np.arange(50))
# - use this on kits that split way back BCE: plt.xticks(np.arange(0,800,25))

D                                                                                                    =
dendrogram(Z,labels=L,color_threshold=3.5,leaf_font_size=12,leaf_rotation=0,orientati
on='left')
#plt.gca().invert_yaxis()
for i, d, c in zip(D['icoord'], D['dcoord'], D['color_list']):
        y = 0.5 * sum(i[1:3])
        x = d[1]
        if x > 0:
            plt.plot(x, y, 'o', c=c)
            yr = math.floor((Byear - int(x* Rscale))/10)*10
            yr = int(yr)
            if yr >= 0:
                yr_txt = "{yr}".format(yr=yr)
            else:
                yr_txt = "{yr} BCE".format(yr=-yr)
            #rcc_txt = int(x*10)/10
            rcc_txt = "{:.2f}".format(x)
            plt.annotate("%s" % yr_txt, (x, y), xytext=(-6, -12),
                            textcoords='offset points', color='r',
                            va='center', ha='center', rotation=90)
            plt.annotate("%s" % rcc_txt, (x, y), xytext=(+7, 0),
                            textcoords='offset points', color='r',
                            va='center', ha='center', rotation=90)
plt.annotate("RCC = {rscale} years".format(rscale=Rscale),(0,0),xytext=(0,-5))
F.subplots_adjust(left=0.05, right=0.85, top=0.97, bottom=0.05)
plt.savefig("{input}.jupyter.png".format(input=I))

# cite: http://www.jogg.info/pages/72/files/Howard.htm
# Dating Y-DNA Haplotypes on a Phylogenetic Tree: Tying the Genealogy of Pedigrees and
Surname Clusters into Genetic Time Scales
# William E. Howard III and Frederic R. Schwab
# http://www.wwjohnston.net/famhist/bill-howard-ydna.htm

# This code originally written by David Johnston in Australia.
# Modified by George El Zakhem and Wesley Johnston in the USA.
```

Appendix 2

RCC1 Python Code

```
# User Manual at http://www.wwjohnston.net/famhist/bill-howard-ydna.htm
from scipy.stats import pearsonr
from scipy.cluster.hierarchy import dendrogram, linkage
#from pprint import pprint
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist
from matplotlib import pyplot as plt
import math

# ---------1 - SET CONTROL VARIABLES-----------------------------

I = 'yoemans-111Corrected'
#Specify the file name of the CSV file.
# This will also be the phylogenetic tree's title and file name.

Rscale=1.27
Byear = 1950
RCC1added = 13
RCC1markers = 111
# Rscale is the years/RCC ratio, calibrated to this family & number of markers.
# Byear is the Birth Year used to compute the split years.
# Set Byear about the birth year of the test takers.

RCC1added = 13
RCC1markers = 111
# RCC1 is the value to which fabricated vector z1 markers are set.
# RCC1markers is the number of Y-STR markers of each input kit.

# ---------2 - READ INPUT INTO MATRICES-----------------------------

# read the Input file
C = pd.read_csv("{input}.csv".format(input=I)).values

# extract the kits as rows of observations and columns of attributes (markers)
K = C[:,2:845]
# Note that the numbering starts with 0 and not 1 for both rows and columns.

# extract the kit numbers and names to produce a labels matrix
L = []
for ix,row in enumerate(C):
    L.append("#{ix} {kit} {name}".format(ix=ix+1,kit=row[0],name=row[1]))

# reverse the list and kits
#L = L[::-1]
#K = K[::-1]

# ---------3 - CALCULATE RCC1 MATRIX FOR ALL PAIRS OF KITS-------------
```

```
# calculate a condensed distance matrix consisting of RCC values as per Bill Howards
paper
# the condensed distance matrix is a single dimension matrix containing
# the top triangle of a two dimensional distance matrix
# e.g. [1,2,3,4]
#      [2,1,2,3]
#      [3,2,1,2]
#      [4,3,2,1]
# becomes a condensed distance matrix Y of [2,3,4,2,3,2]
Y = []
X = []
for i1,v1 in enumerate(K):
    X1 = []
    z1 = np.zeros((RCC1markers,), dtype=int)
# --------3A-Fabricate vectors z1 and z2------------------
    for z1x in range(RCC1markers):
        if z1x == 1:
            z1[z1x] = 1
        else:
            z1[z1x] = RCC1added
    for i2,v2 in enumerate(K):
        z2 = np.zeros((RCC1markers,), dtype=int)
        for x in range(RCC1markers):
            if v2[x] == v1[x]:
                z2[x] = z1[x]
            else:
                z2[x] = z1[x]-1
# --------3B-Calculate RCC1 in Matrix-------------------------
        if i2 > i1:
            if np.ndarray.all(z1 == z2):
                Y.append(0)
            else:
                Y.append((1/pearsonr(z1,z2)[0]-1)*10000.0)
        if np.ndarray.all(v1 == v2):
            X1.append(0)
        else:
            X1.append((1/pearsonr(z1,z2)[0]-1)*10000.0)
    X.append(X1)

# ---------4 - GENERATE THE PHYLOGENETIC TREE SPLITS-------------

# perform the agglomerative clustering using the average method as per Bill Howards
paper
Z = linkage(Y,method='weighted',optimal_ordering=False)

# perform linkage reordering such that the shorter branch is first, the longer branch
second
# the lower index is first, the higher index is second
for link in Z:
    leftDepth = link[0]
    if leftDepth >= len(L):
        leftDepth = Z[int(leftDepth)-len(L),3]
    else:
        leftDepth = 1
```

```
    rightDepth = link[1]
    if rightDepth >= len(L):
        rightDepth = Z[int(rightDepth)-len(L),3]
    else:
        rightDepth = 1
    if leftDepth < rightDepth:
        t = link[0]
        link[0] = link[1]
        link[1] = t
    elif link[0] < link[1] and link[0] < len(L) and link[1] < len(L):
            t = link[0]
            link[0] = link[1]
            link[1] = t

# ---------5 - PLOT THE PHYLOGENETIC TREE-------------

# plot the cluster hierarchy produced by linkage as a dendrogram
F = plt.figure(figsize=(16,20),dpi=72) # A1 paper
plt.title(I)
plt.xlabel("RCC1")
plt.grid(True,which='major',axis='x',color='g',linestyle='dashed')
plt.minorticks_on()
plt.tick_params(axis='x',which='minor')
plt.tick_params(axis='y',which='minor',length=0)

#plt.xscale('symlog',basex=2)

plt.xticks(np.arange(0,800,12))
#plt.xticks((0,1,2,4,8,16,32,64,128,256),(0,1,2,4,8,16,32,64,128,256))
# - use this on recent kits within the last 50 RCC: plt.xticks(np.arange(50))
# - use this on kits that split way back BCE: plt.xticks(np.arange(0,800,25))

D                                                                                =
dendrogram(Z,labels=L,color_threshold=3.5,leaf_font_size=12,leaf_rotation=0,orientati
on='left')
#plt.gca().invert_yaxis()
for i, d, c in zip(D['icoord'], D['dcoord'], D['color_list']):
        y = 0.5 * sum(i[1:3])
        x = d[1]
        if x > 0:
            plt.plot(x, y, 'o', c=c)
            yr = math.floor((Byear - int(x* Rscale))/10)*10
            yr = int(yr)
            if yr >= 0:
                yr_txt = "{yr}".format(yr=yr)
            else:
                yr_txt = "{yr} BCE".format(yr=-yr)
            rcc_txt = "{:.2f}".format(x)
            plt.annotate("%s" % yr_txt, (x, y), xytext=(-6, -12),
                                textcoords='offset points', color='r',
                                va='center', ha='center', rotation=90)
            plt.annotate("%s" % rcc_txt, (x, y), xytext=(+7, 0),
                                textcoords='offset points', color='r',
                                va='center', ha='center', rotation=90)
plt.annotate("RCC1 = {rscale} years".format(rscale=Rscale),(0,0),xytext=(0,-5))
```

```
F.subplots_adjust(left=0.05, right=0.85, top=0.97, bottom=0.05)
plt.savefig("{input}.jupyter.png".format(input=I))

# cite: http://www.jogg.info/pages/72/files/Howard.htm
# Dating Y-DNA Haplotypes on a Phylogenetic Tree: Tying the Genealogy of Pedigrees and
Surname Clusters into Genetic Time Scales
# William E. Howard III and Frederic R. Schwab

# This code originally written by David Johnston in Australia.
# Modified by George El Zakhem and Wesley Johnston in the USA.
```

Appendix 3

Testing Input Order of Kits

Context

Many clustering methods rely on a "seed" case to be identified, from which the algorithm then can measure distance to determine whether other cases belong in the same or a different cluster. When the seed is pre-determined or is somehow implicit in the order of presentation of the input data to the program, researchers need to know about this bias, so that they can take it into account when seeking conclusions.

The phylogenetic tree method does not rely on an explicitly pre-established seed case. The following tests sought to establish whether different input order of the kits resulted in different output of phylogenetic trees. The tests found no such difference. The program generated the same phylogenetic tree, regardless of the order in which the kits were presented to the program.

The tests sought to create extreme conditions, since the high number of all permutations of the input order prohibit exhaustive testing. Thus, there may be some situations in which the input order of the kits does generate different trees. But at this time, the tests found no such bias.

The tests were run only on the RCC version and not the RCC1 version, simply because at the time the tests were run the RCC1 version had not yet been created.

Orderings Tested

The input consisted of 19 111-marker kits in the same family project. Four permuations of the order were tested, in two pairs:

1. Sorted on Kit Number
    a. Ascending Order
    b. Descending Order
2. Sorted on stepwise genetic distance from the mode
    a. Ascending Order
    b. Descending Order

The second pair metric was calculated as the stepwise genetic distance of each kit from the mode of all of them. If the values of a specific marker for two kits being compared differs by 3, then stepwise genetic distance counts that as a

distance of 3 (as opposed to simple genetic distance which counts any size mutation as 1).

Appendix 4

Convergence, Reverse and Parallel Mutations

Context

Once Y-SNP testing became available, some researchers either advocated ignoring Y-STRs completely in favor of Y-SNPs or else cited some very real problems of Y-STRs as evidence that they can never effectively conform with Y-SNP results. Other researchers saw the continuing need, because Y-SNP tests cost much more that Y-STR tests, of predicting Y-SNP results from Y-STRs.

Paraphrasing Irwin project administrator James Irvine: Y-STR-based analytical methods are only predictors or estimators, unlike Y-SNP-based haplotrees which illustrate actual phylogenetic relationships. There is inherent uncertainty in Y-STRs and inherent certainty in Y-SNPs.

With more Y-SNP tests, the haplotree fills in more and more completely, bringing more and more certainty. But as long as the costs of the two types of tests differ so greatly, we will always have more Y-STR test results than Y-SNP results. And the Y-STRs really do hold information.

But Y-STRs also have possibilities of mutational events that confound analysis of results in living test takers. From the Y-STR results alone, we cannot know just how the test takers' haplotypes (their values for the Y-STR markers) came to be as they are today. This does not make Y-STRs useless but does require that these issues remain in our consideration of results.

Convergence: Detecting and Avoiding

Convergence is when two lines of descent that at some point had clearly distinct Y-STR haplotypes come, through reverse and/or parallel mutations, to have identical or near-identical values for the markers in two kits when the test takers do not in fact share a recent common ancestor.[11]

In the selection of input kits for RCC trees, you can detect convergence and avoid it. Two methods greatly eliminate its impact.

Select Kit of the Same Haplogroup or Signature SNP Sub-Clade

The surest way to counter convergence is to assure that the kits are in the same haplogroup or share the same signature SNP. Here again (see section 2 of the

[11] See the ISOGG page on Convergence at https://isogg.org/wiki/Convergence and also Maurice Gleeson's excellent examples on his page https://dnaandfamilytreeresearch.blogspot.com/2017/05/convergence-what-is-it.html

user manual) is a note that Bill Howard posted 30 Nov 2013 on the Rootsweb Genealogy-DNA e-mail list:[12]

"One last point, please try to make the haplotype set as pure as possible. By that I mean, if it is an M222 SNP, be sure all haplotypes have been SNP-tested. If it is a surname set, I would prefer that they all are in the same haplogroup (otherwise the tree gets unwieldy, long, and will lack time resolution in regions of genealogical interest). The larger the set, the better the optimization process will be when the tree is produced …"

In both the match list for a specific kit and in the results pages of Family Tree DNA surname projects, the terminal SNP of each kit tells whether two kits are in the same sub-clade in the Y-SNP haplotree. So, select kits that are in the same haplogroup or, even better, in the same sub-clade.

The primary criterion for evaluating RCC trees is how well they conform with the Family Tree DNA SNP-based Big Y Block Tree. Generally, RCC trees conform quite well with the Big Y Block Tree, when the input adheres to proper limits.

Select Kits with 67 or 111 Y-STR Markers

Most research published on Y-STR convergence considers 37 or fewer markers, with statements that it can occur in 67 or 111 marker kits as well. The reality is that such a statement is virtually meaningless in the absence of empirical evidence. You could just as truthfully say that convergence could occur in kits of 10,000 markers. Just because it theoretically can occur does not mean that it does occur to any significant extent.

Existing convergence research rests on empirical evidence of the same or similar haplotypes having different SNP results. There is no empirical evidence of the underlying mechanisms: how often they occur, over what time frames and in what other conditions. Because all the mutations happened before now, there is no way to gather such empirical past evidence. So, we are left with theoretical considerations.

When the total number of markers is small, the probability of all or nearly all of them aligning for convergence is higher than when the total number of markers is large. The more markers under consideration, the more mutations would be needed to have either reverse or parallel mutation result in two kits ending up with nearly identical STRs when in fact their SNP results differ.

Of course, even if only one out of 111 markers has a reverse or parallel mutation back in time somewhere, the kit can still exhibit convergence. But

[12] Rootsweb terminated support for e-mail lists in 2020. The Genealogy-DNA list successor for the broadest group of former Rootsweb members is now at https://groups.io/g/genealogy-dna

during that same period, it is likely that other markers will also mutate, so that the higher number of markers still reduces the likelihood of convergence.

The bottom line: the more Y-STR markers you use, the less likely – exponentially and not linearly – the possibility of convergence. So, use only 67 or, more preferably, 111 marker kits as input to your RCC tree runs.

Time Frame and Compactness

For convergence to become a problem, mutations must occur. In the case of reverse mutations, two non-concurrent mutations must occur. If the objective of the RCC tree is to show kits that are expected to be related within the era of surnames (roughly 20 generations ago) or more recently and the kits all share the same SNP, then the probability of convergence happening during the time period to the Most Recent Common Ancestor is low.

As a quick alternative to looking up SNPs for kits that have not Big Y tested, Wesley Johnston uses a rule of thumb that the kits going into the RCC algorithm that form a compact group: kits that have stepwise genetic distance of less than 15/111 from the mode of the group.

Another way to enforce a limited time span is to generate an RCC tree and then select one particular branch of that tree to recalibrate and run another RCC tree with just those kits.

The Bottom Line: Convergence is Detectable and Avoidable

While Y-STR convergence is real, it occurs over a long period of time in small sets of markers. The RCC tree (and all other forms of Y-STR analysis) can detect and avoid the impact of convergence with easy steps in the selection of input kits:

1. Select kits in the same haplogroup or sub-clade.
2. Select kits with 67 or, preferably, 111 markers.
3. Select kits that form a compact group.