The probability of a nucleotide sequence formed by random mutations

by Ulrich Utiger*

Abstract

I analyze the stochastic formation of a nucleotide sequence based on Dawkins' weasel algorithm, according to which each nucleotide can mutate with a certain probability and natural selection favors the individuals closest to the ideal genetic information. I provide an analytical solution to this algorithm, from which it is possible to obtain a probability distribution and the average number of generations necessary to build a functional nucleotide sequence. From these results, I estimate a lower limit of the time when the split between the genera *Pan* and *Homo* should at least have occurred and compare it with empirical data, according to which it happened some million years ago. I also show that this lower limit is also valid for more sophisticated algorithms than Dawkins'.

Contents

Introduction 2
1 One nucleotide and several kids 4
1.1 One kid mutating at every generation 4
1.2 One kid mutating occasionally 8
1.3 Several kids mutating occasionally 10
2 Several nucleotides and one kid 12
2.1 First and second generation 12
2.2 Beyond generation two 15
2.3 Reducing the dimension 17
2.4 A direct calculation 20
2.5 Mean and variance 26
3 Several nucleotides and several kids 28
3.1 An analytical solution to the weasel algorithm 28
3.2 A direct calculation 33
3.3 Initial variation 35
3.4 Properties of the weasel distribution 36
3.5 Genes 39
4 A model favoring natural selection 42
4.1 Genetic difference, alphabet length and mutation rate 42
4.2 Fertility, birth and mortality rates 43
4.3 Population size and number 44
4.4 A single parents model 48
4.5 Dating the Pan / Homo split 51
Conclusion 56
Notes 59
References 65

* Ulrich Utiger is an independent researcher who earned a master's degree in theoretical physics from the University of Fribourg, Switzerland. He was lecturer at the HFTM College in Switzerland and is retired since 2018.

Updates to this article can be downloaded from historycycles.org/nucleotides.pdf.

Contact: ulrich.utiger@gmail.com; DOI: 10.6084/m9.figshare.12103413

Last updated on April, 2020

Introduction

Richard Dawkins is a reputed British zoologist who promotes evolution and natural selection through various popular publications. In his book *The Blind Watchmaker*, he compares nucleotide sequences like genes with ordinary text written by humans (2006 pp. 46-49). He first shows that the sentence "METHINKS IT IS LIKE A WEASEL" taken from Shakespeare has a probability of $1/27^{28} \approx 10^{-40}$ to be produced by chance alone as there are 28 letters in the sentence and the English alphabet contains 27 letters the space character included. He admits that the probability of a working genetic code to be formed by pure chance is even more hopeless, as genes contain thousands of nucleotides.

However, Dawkins suggests that this hurdle can be overcome if natural selection is taken into account, which presumably favors individuals with genes producing phenotypes that give any advantage for survival. If such genes are transmitted to following generations, they might be accumulated and eventually yield new species. In order to sustain his hypothesis, he wrote a little computer program, in which he gradually transforms an arbitrarily chosen sequence of 28 letters into the above-mentioned sentence. He did not publish the program, but from his indications one can conclude that at each trial a certain number of "kids" is formed from the initial sequence – the "parent" – by randomly mutating some characters with a certain probability. Each kid is then compared with the target sentence. The kid closest to the target is selected and made the parent of the next trial.

Dawkins made three runs, reaching the sentence in generation 43, 64 and 41 respectively. This yields an average of about 50 generations. If an adequate mutation rate and number of kids are chosen, it is effectively possible to achieve the task in fewer than 100 trials or "generations" with such a simulation. Thereby, the probability that a functional nucleotide sequence is formed is increased by several orders of magnitude, which is a considerable boost.

This is why the weasel program is taken by many as a proof that natural selection can produce new species. It has gained sufficient popularity to spark criticism from all kinds of Christian denominations. William A. Dembski and Robert J. Marks (2009; see also www.evoinfo.org), two prominent members of the intelligent design movement, tried to show that random processes cannot create new information and thereby the weasel program is not representative of how nature works. Although Dawkins himself admits that life does not work like his program and evolution has no programmed long-term goal, he nevertheless suggests that it be efficient as a short-term selection process. This appears to be true, as his short sentence is rapidly achieved.

However, it is quite another issue to estimate how much time the weasel algorithm would need to achieve long nucleotide sequences corresponding to genetic divergences between species. The suggestion that it also works on such large scales is a linear extrapolation into unknown dark territory. Michael Behe (2006 p. 6) would call this a black box, which he defines as something people believe to understand, although they have no clear idea of its inner mechanics. Unfortunately, such linear extrapolations based on pure intuition are very widespread. While intuition is absolutely necessary to initiate any scientific research, it is pseudoscience if such research remains in its stage of intuition. Furthermore, intuition is often misleading in cases where a rigorous mathematical background is lacking. Complex realities may behave linearly on short local extends, but on larger scales they are most often unpredictable. During dark ages, such short-sighted thinking has deluded people into believing that the Earth is flat because sitting on a high mountain and looking around suggests this impression.

Dawkins did not even take care to perform a linear approximation. For this purpose, he could have done two simulations with realistic genetic parameters (alphabet length, mutation rate, number of kids and so on) instead of taking a sentence from Shakespeare: the first simulation with a little nucleotide sequence and the other with a larger one. With some trying out, he could have estimated the average number of generations necessary to reach the target (the mean for short) for each sequence. From this result, he could have drawn a straight line connecting both means and extrapolate it linearly to a very large sequence – to only realize that it still needs a

tremendous number of generations to reach a nucleotide sequence of the order of some millions of nucleotides corresponding, for instance, to the genetic difference between chimps and humans. But apparently, he only did content himself to stimulate the fantasy of his readers, who are eagerly willing to believe in his arguments and displayed authority, which he formidably succeeded, judging by all the websites devoted to his algorithm[1].

Was he the tireless fighter for the truth, as whom he likes to present himself, he could even have performed more than two simulations with different sequence lengths, to be able to announce then resoundingly that the corresponding mean values are not linear as function of the number of nucleotides but curving down like earth's surface beyond the horizon in the dark area inaccessible to the eye, thus shortening the mean considerably. However, is it bending down enough? This is the answer I am looking for in this article: instead of proving that the weasel algorithm is unrealistic like Dembski and Marks have tried to show, I will take quite a different approach and presuppose as null hypothesis that it can produce new species and test it against empirical data. In other words, contrarily to Dawkins, I take care to perform accurate calculations with error estimates and so on in order to shed light into this black box and see what really happens when Dawkins' program is applied to large nucleotide sequences.

A major problem to overcome in this effort is that such numerical Monte Carlo simulations would last thousands of years on a desktop computer. Even supercomputers would probably take some considerable time. Therefore, what is needed is an analytical solution to the weasel algorithm that can quickly be applied to different parameters like the number of letters, the mutation rate and so forth. I also use extrapolations, but they are based on an accurate mathematical background. If there is any uncertainty regarding parameters or if simplifications are made, I will always favor evolutionary concepts.

The mathematical formalism is made on a level such that it should hopefully also be accessible for non-mathematicians like biologists, anthropologists, paleontologists and – zoologists. Computer algorithms are performed with Mathematica, which is a very intuitive language. No programming prerequisites are necessary, even though it is an advantage. On the other hand, basic concepts in probability theory, calculus, linear algebra and so on are required. No exotic theories and methods are used, which is why no references are made to them as they can easily be found in corresponding standard textbooks. In order to not burden the main text with too much calculations, a number of them are relegated to the Mathematica notebook file *codes.nb*[*] and also the *Notes* section.

With these tools established, I show that the expected number of generations provided by the weasel algorithm is a lower limit for more realistic algorithms. I will apply it to the split between the human and chimpanzee lineages, which is believed to have happened some million years ago. Because natural selection is favored, the expected number of generations necessary to produce the human species since this split should yield a time span that is far below millions of years or at least come close to such figures if natural selection was true. On the other hand, if the expected number yields a time far above millions of years, then natural selection is proven a null hypothesis that should be dismissed by the scientific community.

[*] Available from historycycles.org/codes.zip.

1 One nucleotide and several kids

Let us start with simple conditions, assuming only one nucleotide, that is, a sentence of length one. The length of the alphabet is labeled *a* and the number of trials or generations *v*. For different models regarding mutation, persons involved and so on, we will then calculate distributions $P(v)$, yielding the probability that a "gene" of length one is built after *v* generations. From this distribution, we will be able to calculate the mean value $\mu = \langle v \rangle$ representing the average number of trials necessary to achieve the target nucleotide, in other words, for a kid to obtain a functional gene after *v* generations.

1.1 One kid mutating at every generation

As an introductory calculation, let us start with the simplest possible model: the throwing of a die, in which case $a = 6$ and the probability of mutation is equal to one. In other words, it takes place at every generation. This scenario amounts to take only one single kid, or equivalently, a parent. One method to obtain the mean is to count how many trials *v* are necessary to get the target nucleotide, which can be represented by any number between 1 and 6, it does not matter. If one repeats this over and over again, one is able to get an idea of the average number of trials necessary to achieve the goal. But this is a very time-consuming method. A better approach is to first calculate the distribution and then to calculate the mean analytically. Numerically, the distribution can be calculated with the following Mathematica algorithm:

Code 1

```
1   a=6; z=10^6; V=40; target=a; w[v_]:=0;
2   Do[
3    v=1; parent=RandomInteger[{1,a}];
4    While[parent!=target,
5     v++;
6     If[v==V,Break[]];
7     parent=RandomInteger[{1,a}]];
8    w[v]++,
9    {z}];
10  ListPlot[Table[w[v]/z,{v,1,V-1}]]
```

A series of parameters is initialized on line 1. The *Do* loop between lines 2 and 9 just repeats *z* times the same code, which produces a winner *w*[*v*] for every *z*. In this case, *w*[*v*] is incremented on line 8 after $v-1$ unsuccessful trials inside the *While* loop between lines 4 and 7, which runs as long as $parent \neq target$ or *v* exceeds *V* (line 6). This parameter initialized to 40 on the first line may accelerate the calculation considerably as it hinders the *While* loop to make endless iterations in a region where $P(v)$ decreases to zero. For a large number of *z* (a million times in our case), the relative frequency *w*[*v*]/z tends to $P(v)$. On line 3 at the beginning of the *Do* loop, a die roll is simulated with the attribution of a random integer between 1 and *a* to the variable *parent* and then again at the end of the *While* loop on line 7. The result can be displayed with *ListPlot*, inside which a list of the outcome is generated with *Table* (line 10). The parameter *v* only runs from 1 to $V-1$ because *w*[*V*] is an erroneous result that must be dismissed, for it is not incremented because of a winner but because of the breaking inside *While*. Some plots of this numerical simulation for different *a* can be seen in figure 1.

Next, the laws of probability are used in order to get not only the same result even faster but also in a form that is applicable for different values of *a*: for each trial, there is only one favorable outcome, for instance when the die shows 6 (the target). The possible outcomes are 1, 2, 3, 4, 5 or 6, so their number is 6 (or *a* in general). Therefore, the probability of the event G = "the parent got the good nucleotide" is

$$p = P(\mathrm{G}) = \frac{1}{a} \tag{1}$$

on every single trial. Before this event happens, the event $\overline{\mathrm{G}}$ = "the parent got a bad nucleotide" occurs $v-1$ times, which has $a-1$ favorable outcomes. In the case of a die, these are 1, 2, 3, 4 and 5. So the probability is

$$q = P\left(\overline{\mathrm{G}}\right) = \frac{a-1}{a} = 1 - \frac{1}{a} = 1 - p \tag{2}$$

on every single trial, which shows that q is the complementary probability of p. From this, one gets the following probability distribution

$$P(v) = P\left(v-1 \text{ times } \overline{\mathrm{G}} \text{ and G on } v^{\text{th}} \text{ trial}\right)$$

where "and" means a logical AND equivalent to the intersection $\cap$, whereas "times" means a series of independent events related by AND. Hence, the corresponding probabilities are multiplied with each other because the outcomes of die rolls are mutually independent. This yields

$$P(v) = P\left(v-1 \text{ times } \overline{\mathrm{G}}\right) \cdot P\left(\mathrm{G} \text{ on } v^{\text{th}} \text{ trial}\right) = q^{v-1} p = \left(1 - \frac{1}{a}\right)^{v-1} \frac{1}{a} \tag{3}$$

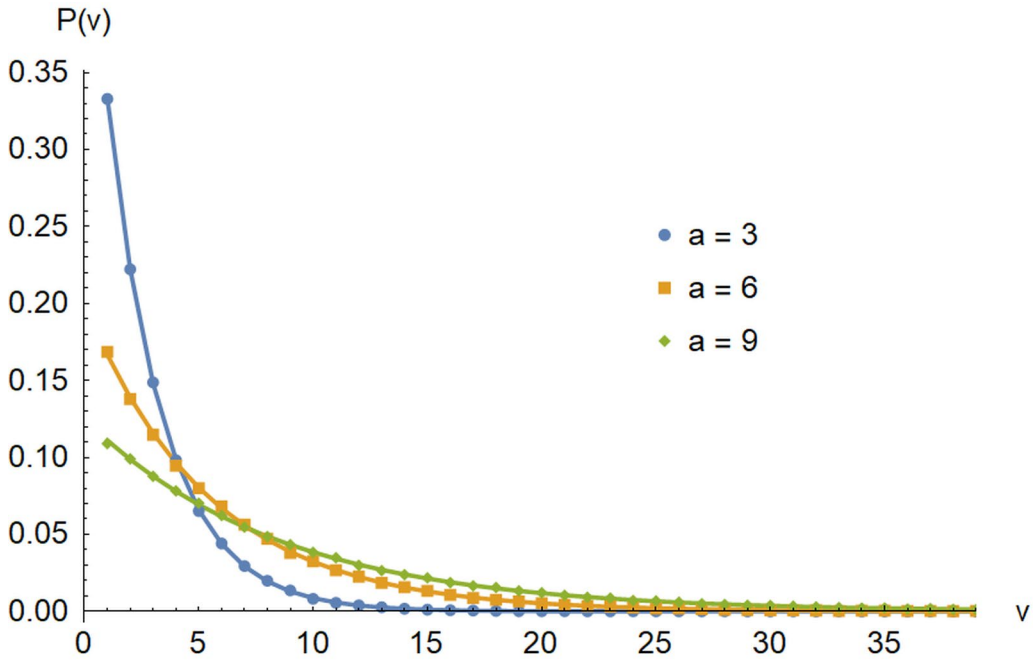

Figure 1: The probability distribution of throwing a die until it shows the correct number after v trials, a being the number of regular faces of the die. This models a single nucleotide mutating at every generation.

Distributions of this form are called *geometric*. In figure 1, an overlay of some numerical points and analytical plots (colored joined curves) shows that they perfectly coincide (see codes.nb). In order to prove that the above law is really a probability distribution, one needs to show that the sum of all its values yields unity. For this purpose

$$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q} \to \frac{1}{1-q} \text{ for } n \to \infty \tag{4}^2$$

is used, from which follows

$$\sum_{v=1}^{\infty} P(v) = \sum_{v=1}^{\infty} q^{v-1} p = p \sum_{v=1}^{\infty} q^{v-1} = p \sum_{k=0}^{\infty} q^k = p \frac{1}{1-q} = p \frac{1}{1-(1-p)} = 1$$

What is of particular interest is the mean (or expected) value of the number of throws v necessary to get the correct nucleotide, which is defined as

$$\mu = \langle v \rangle = \sum_{v=1}^{\infty} v\,P(v) \tag{5}$$[3]

For this calculation, the following intermediate result is needed:

$$\sum_{v=1}^{\infty} v\,q^{v-1} = \frac{1}{(1-q)^2} \tag{6}$$[4]

Together with equations 1 and 2 this yields

$$\mu = \langle v \rangle = \sum_{v=1}^{\infty} v\,P(v) = p\sum_{v=1}^{\infty} v\,q^{v-1} = p\frac{1}{(1-q)^2} = p\frac{1}{p^2} = \frac{1}{p} = a \tag{7}$$

In other words, on the average there are as many throws needed as sides on the die in order to get the correct nucleotide. This is valid for any die that has a regular shape like cubes or polyhedrons that can have an arbitrary number a of sides of equal area. The center of gravity and geometric center must coincide, of course.

If Dawkins had done this simple calculation, he could have told his audience that on the average 10^{40} generations are necessary to produce the weasel sentence instead of telling them that its probability is 10^{-40}, as it is not immediately clear why an average is the reciprocal value of a probability (according to equation 7). This would have better fitted to his claim that on the average about 50 generations are necessary to achieve the weasel sentence with his algorithm. In other words, comparing probability and mean is like comparing apples and pears, which reveals his profound lack of understanding mathematical concepts. Trying to randomly produce such a sentence is in fact like throwing a very big die with 10^{40} faces: when throwing two dice each having six regular faces, the possible outcomes are $6^2 = 36$. Consequently, the probability that both dice show the same number is $p = 1/36$ on a single roll. Thereby, this is like throwing one single die with 36 faces. Extend this to an arbitrary number of dice.

Another important statistical quantity is the standard deviation σ, which has a similar form as the mean in that it tells us the mean deviation from the mean value of throws. This can be important because a number of throws outside the standard deviation may be very improbable. Mathematically, the standard deviation is the square root of the variance, which is defined as follows:

$$\sigma^2 = \sum_{v=1}^{\infty} \left(v - \langle v \rangle\right)^2 P(v)$$

With this definition, one gets

$$\sigma^2 = \langle v^2 \rangle - \langle v \rangle^2 \tag{8}$$[5]

In order to show what the variance of the geometric distribution is, the intermediate result

$$\sum_{v=1}^{\infty} v^2 q^{v-1} = \frac{1+q}{(1-q)^3} \tag{9}$$[6]

is used. From this result and equation 2 follows

$$\langle v^2 \rangle = \sum_{v=1}^{\infty} v^2 P(v) = p\sum_{v=1}^{\infty} v^2 q^{v-1} = p\frac{1+q}{(1-q)^3} = p\frac{2-p}{p^3} = \frac{2-p}{p^2}$$

Finally, with equations 7 and 8 the variance is

$$\sigma^2 = \frac{2-p}{p^2} - \frac{1}{p^2} = \frac{1-p}{p^2} = a(a-1) \tag{10}$$

Quite another situation arises if one requires that an entire population consisting of b persons shall get correct numbers. In other words, if there is only one person with a bad nucleotide, the event G = "all persons got good nucleotides" is not taking place. The probability of this event is as follows:

$$\begin{aligned} &P\left(\text{one person got a bad nucleotide and the others good ones}\right) \\ &= P\begin{pmatrix} \text{the } 1^{\text{st}} \text{ person got a bad nucleotide and the others good ones or ...} \\ \text{... or the } b^{th} \text{ person got a bad nucleotide and the others good ones} \end{pmatrix} \\ &= q\,p^{b-1} + \ldots + q\,p^{b-1} = b\,q\,p^{b-1} \end{aligned}$$

where "or" means a logical OR equivalent to the union $\cup$. If k persons got bad nucleotides and $b-k$ persons good ones, the number of combinations how these can be chosen out of b persons is the binomial coefficient

$$C_k^b = \binom{b}{k} = \frac{b!}{k!\,(b-k)!}$$

The corresponding probability is

$$P\left(k \text{ persons got a bad nucleotide and the others good ones}\right) = C_k^b\,q^k\,p^{b-k}$$

Hence, for every $k = 1,\ldots,b$ all these combinations need to be summed to get

$$P\left(\overline{\mathrm{G}}\right) = \sum_{k=1}^{b} C_k^b\,q^k\,p^{b-k}$$

Using the binomial theorem, $q + p = 1$ and $C_0^b = 1$, this can be simplified to

$$\sum_{k=0}^{b} C_k^b\,q^k\,p^{b-k} = \left(q+p\right)^b \;\Rightarrow\; \sum_{k=1}^{b} C_k^b\,q^k\,p^{b-k} = \left(q+p\right)^b - p^b = 1 - p^b \tag{11}$$

which finally yields

$$P\left(v\right) = P\left(v-1 \text{ times } \overline{\mathrm{G}} \text{ and G on the } v^{\text{th}} \text{ trial}\right) = P\left(\overline{\mathrm{G}}\right)^{v-1} P\left(\mathrm{G}\right) = \left(\sum_{k=1}^{b} C_k^b\,q^k\,p^{b-k}\right)^{v-1} p^c = \left(1-p^b\right)^{v-1} p^b \tag{12}$$

This is also a geometric distribution since p can be replaced with p^b in equation 3, which yields the mean

$$\mu = \langle v \rangle = \frac{1}{p^b} = a^b \tag{13}$$

from equation 7. Here therefore, the expected number of generations that a population of b persons gets the right nucleotides increases exponentially with b. The situation is quite different if one asks what is the probability that at least one person hits the right number (or gets the right nucleotide). For c persons, one would intuitively expect that it goes c times faster and that thereby the mean is reduced from a to a/c according to equation 7.

This comes close to the solution, but it's not entirely exact: let E be an arbitrary independent event with $p = P\left(\mathrm{E}\right)$ and $q = P\left(\overline{\mathrm{E}}\right) = 1 - p$. What then is the probability that E is realized k times in the course of c trials? E being independent for each trial, different orders in which E is realized will lead to the same result for fixed k and c. This is why the number of all possible combinations of the composed event *"k* times E and $c-k$ times $\overline{\mathrm{E}}$" is again the binomial coefficient C_k^c. Therefore, $P\left(\mathrm{E} \text{ realized } k \text{ times}\right) = C_k^c\,p^k\,q^{c-k}$, which is correspondingly called a *binomial distribution*. Now, the complementary event of "E realized at least once" is "E never realized", which yields

$$P\left(\mathrm{E} \text{ realized at least once}\right) = 1 - P\left(\mathrm{E} \text{ never realised}\right) = 1 - C_0^c p^0 q^{c-0} = 1 - q^c \tag{14}$$

In particular, if G = "at least one person got the good nucleotide" instead of E, its probability is

$$P(\mathrm{G}) = 1 - q^c$$

Before this happens on the v^{th} trial, the complementary event $\overline{\mathrm{G}}$ = "nobody got good nucleotides" occurs $v-1$ times with a probability that is always the same on a single independent trial:

$$\begin{aligned} P(\overline{G}) &= P\left(\text{the } 1^{\text{st}} \text{ person got a bad nucleotide and ... and the } c^{\text{th}} \text{ person got a bad nucleotide}\right) \\ &= P\left(\text{the } 1^{\text{st}} \text{ person got a bad nucleotide}\right) \cdot \ldots \cdot P\left(\text{the } c^{\text{th}} \text{ person got a bad nucleotide}\right) = q^c \end{aligned}$$

Consequently, the probability distribution is

$$P(v) = P(\overline{G})^{(v-1)} P(G) = q^{c(v-1)}\left(1 - q^c\right) \tag{15}$$

Comparing this result with equation 3, one realizes that this is also a geometric distribution since q can be replaced with q^c and thereby $p = 1 - q$ with $1 - q^c$ to get the mean

$$\mu = \langle v \rangle = \frac{1}{1-q^c} = \frac{1}{1-(1-p)^c}$$

from equation 7. Hence, this is not quite the same as $a/c = 1/(c\,p)$. But if p is very little, a first order Taylor expansion around zero yields:

$$\langle v \rangle \approx \frac{1}{c\,p} + \frac{c-1}{2c} + \frac{\left(c^2-1\right)p}{12c} \approx \frac{1}{c\,p} = \frac{a}{c}$$

The second approximation can be made because the first term is by far the dominant one (see codes.nb), so the above intuition is nevertheless correct. We will come back to these results in section 4.4.

1.2 One kid mutating occasionally

Let us now introduce mutation, assuming that the die is only thrown on some occasions with probability m. If the die is not thrown, one may just touch it with the fingertip and consider this a throw, which is a rule that would never be accepted in a casino... However, when the die is thrown or just touched is completely arbitrary. For such a stochastic process, code 1 must be modified as follows:

Code 2

```
1   Clear["Global`*"]
2   a=6; mn=3; md=10; z=10^6; V=40; target=a; w[v_]:=0;
3   Do[
4    v=1; parent=RandomInteger[{1,a}];
5    While[parent!=target,
6     v++;
7     If[v==V,Break[]];
8     If[1 <= RandomInteger[{1,md}] <= mn, parent=RandomInteger[{1,a}]]];
9    w[v]++,
10   {z}];
11  ListPlot[Table[w[v]/z,{v,1,V-1}]]
```

In order to avoid interferences with previous calculations, all variables are reset on line 1. The mutation rate is programmed to a probability of $m = mn/md$ on line 8, where *mn* is the numerator and *md* the denominator of *m* with $mn < md$. In fact, when randomly choosing a number between 1 and *md*, the probability that this number is between 1 and *mn* is precisely $m = mn/md$ because the possible outcomes are *md* and the favorable outcomes *mn*. If this happens, a new value is attributed to the parent, otherwise it remains the same as on the previous trial. This simulation takes a bit more time than code 1. Some results for different *m* can be seen in figure 2.

For an analytical solution of this situation, new parameters are needed: let M be the event "the parent mutates" and m its probability, therefore $P(\mathrm{M}) = m$ and $P(\overline{\mathrm{M}}) = 1 - m$, where $\overline{\mathrm{M}}$ means the complementary event of M. With equation 1, the probability that the parent mutates to the good nucleotide is therefore

$$p = P(\mathrm{M \text{ and } G}) = \frac{m}{a} \tag{16}$$

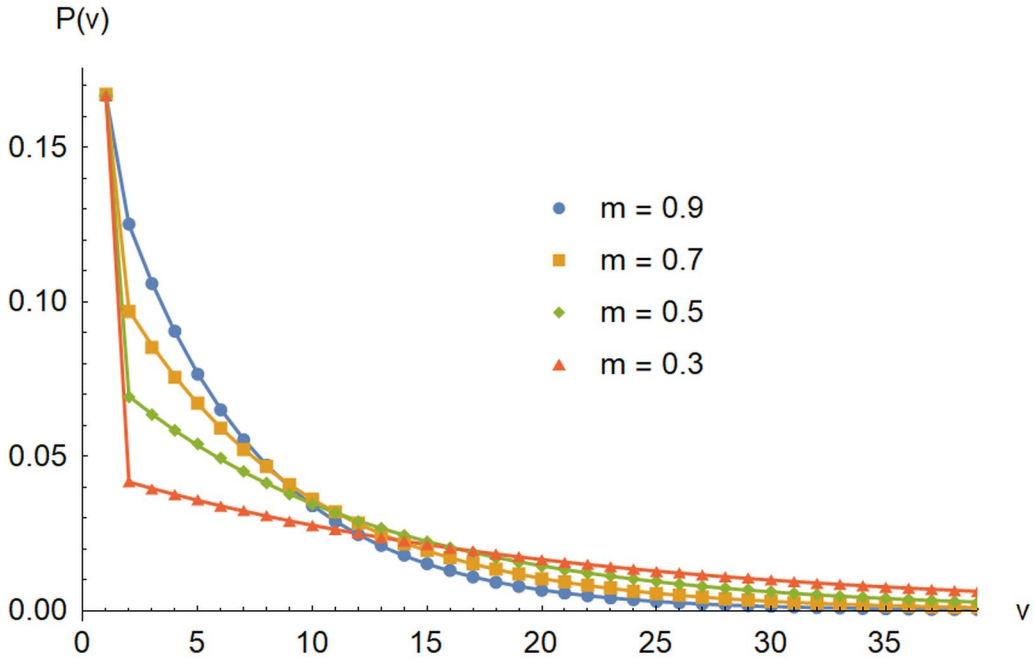

Figure 2: The probability distribution of a die occasionally thrown with probability m and $a = 6$, simulating the mutation of a nucleotide.

If the parent does not mutate, it remains the same as in the previous trial. Therefore, it has a bad nucleotide. With equation 2, the probability that the parent is wrong is then

$$q = P\left(\overline{\mathrm{M}} \text{ or } \left(\mathrm{M \text{ and } } \overline{\mathrm{G}}\right)\right) = 1 - m + m\left(1 - \frac{1}{a}\right) = 1 - \frac{m}{a} = 1 - p \tag{17}$$

On the first trial, there is no mutation, so $P(1) = P(\mathrm{G \text{ on first trial}}) = 1/a$. This yields the following distribution for $v > 1$:

$$\begin{aligned} P(v) &= P\left(\overline{\mathrm{G}} \text{ on } 1^{\mathrm{st}} \text{ trial and } v-2 \text{ times } \overline{\mathrm{G}} \text{ and G on } v^{\mathrm{th}} \text{ trial}\right) \\ &= \left(1 - \frac{1}{a}\right) q^{v-2} p = \left(1 - \frac{1}{a}\right)\left(1 - \frac{m}{a}\right)^{v-2} \frac{m}{a} \end{aligned} \tag{18}$$

As can be seen in figure 2, this distribution – almost geometric – coincides very well with the numerical result. And it is still a probability[7], whereas the mean is

$$\mu = \langle v \rangle = \frac{a + m - 1}{m} \tag{19}^{8}$$

It comes as no surprise that this yields a if $m = 1$, that is, the same result as in equation 7. The variance is

$$\sigma^2 = \frac{(a-1)(a-m+1)}{m^2} \tag{20}^{9}$$

As can be expected, this yields equation 10 for $m = 1$. Because m^2 is the denominator, the variance tends to infinity for m tending to zero, which is also suggested by figure 2 because the distribution flattens increasingly

for *m* decreasing. This reflects the fact that the target can only be reached on the first trial if the parent never mutates.

1.3 Several kids mutating occasionally

Let us now introduce a number *d* of kids: the target is achieved in the case where at least one of them has the correct nucleotide. For that purpose, the parameter *score* is introduced and set to 1, otherwise to 0. Consequently, code 2 must be modified as follows:

Code 3

```
1   Clear["Global`*"]
2   a=14; mn=3; md=10; target=a; d=5; z=10^5; V=100; w[v_]:=0;
3   Do[
4    v=1; score=0; parent=RandomInteger[{1,a}];
5    If[parent==target, score=1];
6    While[score==0,
7     v++;
8     If[v==V,Break[]];
9     Do[
10     If[1 <= RandomInteger[{1,md}] <= mn, kid=RandomInteger[{1,a}], kid=parent];
11     If[kid==target, score=1],
12     {d}]];
13    w[v]++,
14    {z}];
15  ListPlot[Table[w[v]/z,{v,1,V-1}]]
```

If on the first trial the condition *parent = target* is met on line 5, the *While* loop starting on line 6 is not solicited and w[1] is incremented on line 13. Subsequently, the condition for entering the *While* loop does no longer depend on whether the parent is different from the target but on whether at least one kid has scored or not, which is determined by the *Do* loop between lines 9 and 12, where a random integer is attributed to a kid if a mutation occurs, otherwise its value is copied from the parent.

For the analytic solution, $P(1)=1/a$ as in the previous section because there is only one parent in the beginning. For $v>1$, one can use equation 14 (by replacing *c* with *d*) in order to quantify the probability of the event $\mathrm{G}=$ "at least one kid got the good nucleotide", for the *d* kids all individually perform *d* trials, which can happen in all possible orders. Therefore, $P(\mathrm{G})=1-q^d=1-(1-m/a)^d$, using the same *q* than in the previous section (eq. 17). Before this event happens, there are $v-1$ realizations of the event "no kid got the good nucleotide", which is the complementary event of G. This can also be seen directly:

$$P\left(\overline{\mathrm{G}}\right)=P\left(\text{the } 1^{\text{st}} \text{ kid is wrong and the } 2^{\text{nd}} \text{ and ... and the } d^{\text{th}}\right)=q^d=\left(1-\frac{m}{a}\right)^d$$

This yields the following probability distribution for $v>1$:

$$\begin{aligned} P(v) &= P\left(\overline{\mathrm{G}} \text{ on } 1^{\text{st}} \text{ trial and } v-2 \text{ times } \overline{\mathrm{G}} \text{ and G on } v^{\text{th}} \text{ trial}\right) \\ &= \left(1-\frac{1}{a}\right)q^{d(v-2)}\left(1-q^d\right)=\left(1-\frac{1}{a}\right)\left(1-\frac{m}{a}\right)^{d(v-2)}\left(1-\left(1-\frac{m}{a}\right)^d\right) \end{aligned} \tag{21}$$

If $d=1$, one gets the same distribution as for one kid (eq. 18). One only has to replace *q* with q^d and $p=1-q$ with $1-q^d$. Figure 3 shows an overlay of the numerical and analytical results in the case of $a=14$ and $d=5$ for some values of *m*. This distribution is a probability[10] and its mean is

$$\mu=\langle v\rangle=\frac{a\left(q^d-2\right)+1}{a\left(q^d-1\right)}=\frac{a\left(\left(1-\frac{m}{a}\right)^d-2\right)+1}{a\left(\left(1-\frac{m}{a}\right)^d-1\right)} \tag{22}[11]$$

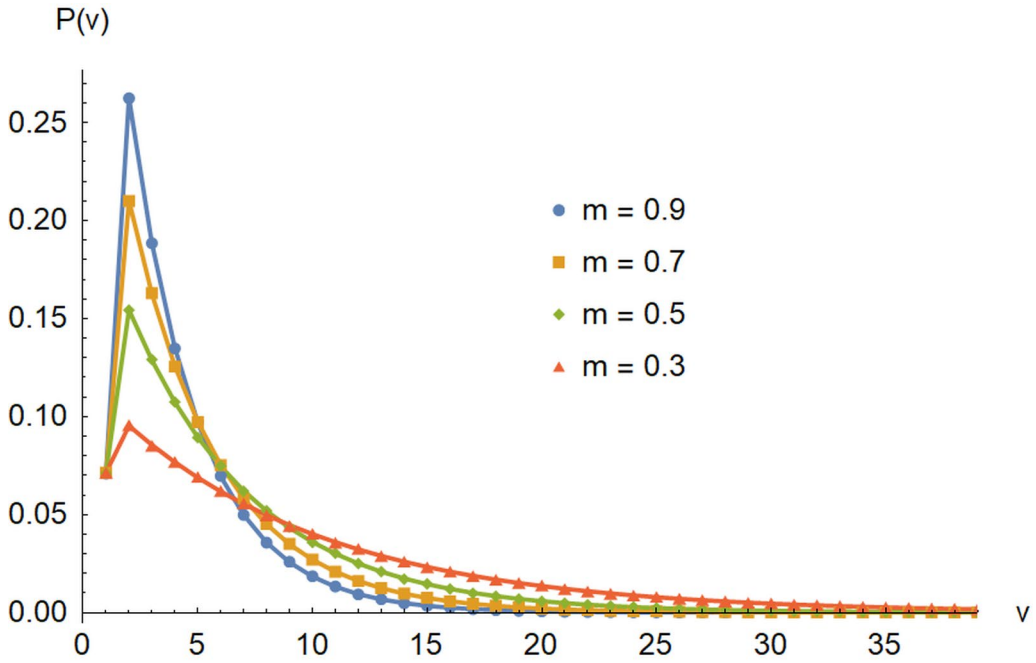

Figure 3: The probability distribution in the case where $a = 14$ and a number $d = 5$ of kids are throwing dice on occasions taking place with probability *m*, which simulates natural selection on a basic level.

whereas the variance is

$$\sigma^2 = \frac{(a-1)\left(1+a\,q^d\right)}{a^2\left(q^d-1\right)^2} = \frac{(a-1)\left(1+a\left(1-\frac{m}{a}\right)^d\right)}{a^2\left(\left(1-\frac{m}{a}\right)^d-1\right)^2} \qquad (23)^{12}$$

We will use these results in section 3.5 when it comes to integrate the concept of genes into Dawkins' weasel algorithm.

2 Several nucleotides and one kid

The complexity increases considerably when several nucleotides are allowed, which is why this situation is first analyzed with one kid, that is, the parent being the only kid. On the other hand, the algorithm itself does not change a lot compared to code 2:

Code 4

```
1   Clear["Global`*"]
2   a=5; t=2; mn=3; md=10; z=10^5; V=400; target=Table[a,{t}]; w[v_]:=0;
3   Do[
4    v=1; parent=RandomInteger[{1,a},t];
5    While[parent!=target,
6     v++;
7     If[v==V, Break[]];
8     Do[
9      If[1<=RandomInteger[{1,md}]<=mn, parent[[i]]=RandomInteger[{1,a}]],
10     {i,1,t}]];
11   w[v]++,
12   {z}];
13  ListPlot[Table[w[v]/z,{v,1,V-1}]]
```

There is a new parameter introduced: the number t of nucleotides. The parameter *target* is now not only one single number but a list of a's of length t, whereas the parent is a list of random numbers between 1 and a of length t, each corresponding to a nucleotide. The *Do* loop between lines 8 and 10 mutates every nucleotide of the parent individually.

2.1 First and second generation

On the first trial, there is no mutation yet and every nucleotide can independently be a number between 1 and a. Hence this yields

$$P(1) = P(\mathrm{G}) = \frac{1}{a^t} \tag{24}$$

For $v > 1$, let us introduce new parameters and modify some others: let N be the event "no mutation occurs" related to any single nucleotide of the parent, so $n = P(\mathrm{N}) = 1 - m$. Then let M be the event "a nucleotide of the parent mutates to the right nucleotide". This definition is somewhat different from the previous one as it is composed of the events "a nucleotide of the parent mutates" and "the parent got a correct nucleotide knowing that he mutates", but $p = P(\mathrm{M}) = m/a$ remains the same as equation 16. Then the event $\mathrm{Q} =$ "a nucleotide of the parent mutates to the bad nucleotide" is defined, which is composed of the events "a nucleotide of the parent mutates" and "the parent got a wrong nucleotide knowing that he mutates". Consequently $q = P(\mathrm{Q}) = m(1 - 1/a)$. Notice that Q is not the complementary event of M and thereby q is different from the previous one (eq. 17). In addition, the events $\mathrm{R} =$ "a nucleotide of the parent is right from the previous trial" and $\overline{\mathrm{R}} = \mathrm{S} =$ "a nucleotide of the parent is wrong from the previous trial" are defined with

$$r = P(\mathrm{R}) = \frac{1}{a} \text{ and } s = P(\mathrm{S}) = 1 - \frac{1}{a} = 1 - r \tag{25}$$

These events must be implied in the calculation because one needs to know whether a nucleotide was correct or not in the previous trial if it does not mutate. Finally, let A_k be all composed events that prevented the parent to get the right nucleotide in the previous trial. Since a nucleotide can either be right or wrong, these events are all t-tuples of the 2-set {R, S} except the tuple that contains only R because this would give good nucleotides to the parent. The events inside a tuple are mutually related by "and", while the relation of one tuple to another is "or". For $t = 2$ for instance, all the possible A_k are

$$\mathrm{A}_1 = \mathrm{R} \text{ and } \mathrm{S} \quad \text{or} \quad \mathrm{A}_2 = \mathrm{S} \text{ and } \mathrm{R} \quad \text{or} \quad \mathrm{A}_3 = \mathrm{S} \text{ and } \mathrm{S} \tag{26}$$

where the first event refers to the first nucleotide of the parent and so on. The event "R and R" does not make part of this list because it implies that the parent got all nucleotides right in the previous trial. The probabilities of the A_k are

$$P(\mathrm{A}_1) = P(\mathrm{A}_2) = r\,s$$
$$P(\mathrm{A}_3) = s^2$$

and the number of all A_k is $2^t - 1$ because the number of all t-tuples of a 2-set is 2^t and there is one less. Rather than using the Mathematica function *Tuples*, a sorted list A of all the probabilities of A_k can be generated with the following code:

Code 5

```
A=Flatten[Table[ConstantArray[r^(t-k) s^k,Binomial[t,k]],{k,1,t}],1]
```

For a fixed k, *ConstantArray* creates a list of the same components $r^{t-k}s^k$. The length of this list is equal to the binomial coefficient

$$C_k^t = \binom{t}{k} = \frac{t!}{k!(t-k)!}$$

because there are C_k^t possible combinations to arrange a number $t-k$ of r and a number k of s in a list of t elements. If $t=3$ and $k=2$ for instance, there are $C_2^3 = 3$ possible arrangements: $\{r,s,s\}$, $\{s,r,s\}$ and $\{s,s,r\}$. Using the binomial theorem

$$(x+y)^t = \sum_{k=0}^{t} C_k^t\, x^{t-k} y^k$$

with $x = y = 1$ one gets

$$2^t = \sum_{k=0}^{t} C_k^t$$

if all these possible combinations are summed from $k=0$ to $k=t$. In code 5 however, k is only running from 1 to t, which is why the case $C_0^t = 1$ must be removed to get the number $2^t - 1$ of all possible t-tuples less one as mentioned above. *Table* creates a list of these products and *Flatten* removes superfluous curly brackets inside this list. Since lists can be considered vectors, A is indeed a vector. In the case $t=2$, there are two $r\,s$ and one s^2 as components of this vector:

$$A = \begin{pmatrix} r\,s \\ r\,s \\ s^2 \end{pmatrix} \tag{27}$$

The A_k together with A_0 = "R and ... and R" form an exhaustive system of mutually disjoint events, of which the union is the entire probability space of the previous trial. In fact, since the sum of all these probabilities can be resumed as $(r+s)^t$ using the binomial theorem, the result of this sum is unity because $r+s = r+1-r = 1$. In other words:

$$\sum_{k=0}^{2^t-1} P(\mathrm{A}_k) = \sum_{k=0}^{2^t-1} A_k = \sum_{i=0}^{t} C_i^t\, r^{t-i} s^i = (r+s)^t = 1 \tag{28}$$

From this can be concluded that

$$\sum_{k=0}^{2^t-1} P(\mathrm{A}_k) = P(\mathrm{A}_0) + \sum_{k=1}^{2^t-1} P(\mathrm{A}_k) = \frac{1}{a^t} + |A| = 1 \;\Rightarrow\; |A| = 1 - \frac{1}{a^t} \tag{29}$$

where $P(\mathrm{A}_0) = P(\mathrm{R}\text{ and ... and R}) = 1/a^t$ according to equation 25 is used. $|X|$ is the 1-norm, which simply sums the absolute values of the components of an arbitrary vector X. Since in our case the components are

probabilities, which always lie between 0 and 1, one does not have to care about absolute values. This is why the Mathematica function *Total* can be used to achieve this sum. The same result can be obtained from a direct calculation if s is replaced by $1-r=1-1/a$.

The event $\mathrm{G}=$ "the parent got good nucleotides on the second trial" can only take place if one of the A_k except A_0 has occurred on the first trial. This condition and equation 28 are necessary in order to apply the law of total probability regarding G:

$$P(2)=P(\mathrm{G})=\sum_{k=0}^{2^t-1}P(\mathrm{A}_k)P(\mathrm{G}\,|\,\mathrm{A}_k)=\sum_{k=1}^{2^t-1}P(\mathrm{A}_k)P(\mathrm{G}\,|\,\mathrm{A}_k) \tag{30}$$

where $P(\mathrm{G}\,|\,\mathrm{A}_0)=0$ is used since G cannot happen if previously $\mathrm{A}_0=$ "the parent got good nucleotides in the previous trial" occurred. This sum can be written in a compact form with the following definitions:

$$G=\begin{pmatrix} P(\mathrm{G}\,|\,\mathrm{A}_1) & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & P(\mathrm{G}\,|\,\mathrm{A}_{2^t-1}) \end{pmatrix} \text{ and } A=\begin{pmatrix} P(\mathrm{A}_1) \\ \vdots \\ P(\mathrm{A}_{2^t-1}) \end{pmatrix}$$

For the sake of simplicity, no difference is made between the event G and its matrix G, except that the last is put in italic. Equation 30 can now be rewritten as follows:

$$|G\cdot A|=\left|\begin{pmatrix} P(\mathrm{G}\,|\,\mathrm{A}_1)P(\mathrm{A}_1) \\ \vdots \\ P(\mathrm{G}\,|\,\mathrm{A}_{2^t-1})P(\mathrm{A}_{2^t-1}) \end{pmatrix}\right|=\sum_{k=1}^{2^t-1}P(\mathrm{A}_k)P(\mathrm{G}\,|\,\mathrm{A}_k)=P(2) \tag{31}$$

G could also be defined as a vector. $P(2)$ could then simply be obtained by the scalar product between G and A, bypassing the 1-norm. In some demonstrations, however, G must be treated as a matrix, which is why it is preferable to define G as diagonal matrix. In order to determine the conditional probabilities, one must consider that for every R occurred in the previous trial either the event N or M must occur in order to get a good nucleotide in the actual trial. On the other hand, for every S occurred in the previous trial the event M must occur in the actual trial. So in the case $t=2$ according to equation 26, one gets the following conditional probabilities:

$$P(\mathrm{G}\,|\,\mathrm{A}_1)=P\big((\mathrm{N}\text{ and }\mathrm{M})\text{ or }(\mathrm{M}\text{ and }\mathrm{M})\big)=np+p^2=p(n+p)$$

$$P(\mathrm{G}\,|\,\mathrm{A}_2)=P\big((\mathrm{M}\text{ and }\mathrm{N})\text{ or }(\mathrm{M}\text{ and }\mathrm{M})\big)=pn+p^2=p(n+p)$$

$$P(\mathrm{G}\,|\,\mathrm{A}_3)=P(\mathrm{M}\text{ and }\mathrm{M})=p^2$$

and thereby

$$G=\begin{pmatrix} p(n+p) & 0 & 0 \\ 0 & p(n+p) & 0 \\ 0 & 0 & p^2 \end{pmatrix}$$

Comparing this result with vector A (eq. 27), one realizes that each r must simply be replaced with $n+p$ and each s with p. This replacement rule is valid for an arbitrary t. So G can be obtained with a code similar to code 5 generating vector A:

Code 6

```
G=DiagonalMatrix[Flatten[Table[ConstantArray[(n+p)^(t-k) p^k,Binomial[t,k]],{k,1,t}],1]]
```

2.2 Beyond generation two

For $v > 2$, a matrix must be constructed that contains the probabilities of all the events that prevent the parent from obtaining the right nucleotides: let B_i = "the outcome of the present trial is equal to A_i for the next trial". From this one gets

$$P\left(\text{the parent got bad nucleotides on the } 2^{\text{nd}} \text{ trial}\right) = P\left(\mathrm{B}_1 \text{ or ... or } \mathrm{B}_{2^t-1}\right) = \sum_{i=1}^{2^t-1} P\left(\mathrm{B}_i\right)$$

Since the events B_i differ from A_i only in the succession of the trials, they also form an exhaustive system of mutually disjoint events which is why the law of total probabilities regarding the event B_i can again be applied to get

$$P\left(\mathrm{B}_i\right) = \sum_{j=1}^{2^t-1} P\left(\mathrm{A}_j\right) P\left(\mathrm{B}_i \mid \mathrm{A}_j\right) = \sum_{j=1}^{2^t-1} b_{ij}\, a_j$$

where $P\left(\mathrm{A}_j\right) = a_j$ and the matrix element $b_{ij} = P\left(\mathrm{B}_i \mid \mathrm{A}_j\right)$ is defined. As in the case of the A_k, the event G = "the parent got good nucleotides on the third trial" can only take place if one of the B_i has occurred on the second trial. This is why one gets again

$$P(3) = P\left(\mathrm{G}\right) = \sum_{i=1}^{2^t-1} P\left(\mathrm{B}_i\right) P\left(\mathrm{G} \mid \mathrm{B}_i\right) = \sum_{i=1}^{2^t-1} \left(\sum_{j=1}^{2^t-1} b_{ij}\, a_j \right) P\left(\mathrm{G} \mid \mathrm{B}_i\right) = \sum_{i,j=1}^{2^t-1} g_i b_{ij}\, a_j = \left| G \cdot B \cdot A \right|$$

according to the law of total probabilities, where $P\left(\mathrm{G} \mid \mathrm{B}_i\right) = g_i$ and B as the square matrix formed by the b_{ij} are defined. Since the outcome of an event B_i is an event A_i for the next trial, one gets $P\left(\mathrm{G} \mid \mathrm{A}_i\right) = P\left(\mathrm{G} \mid \mathrm{B}_i\right) = g_i$. Furthermore, the union of G and all B_i, knowing that previously A_j happened, is also an exhaustive system of mutually disjoint events. This is why this yields

$$g_j + \sum_{i=1}^{2^t-1} b_{ij} = 1 \;\; \forall \;\; j = 1, \ldots, 2^t - 1$$

analogously to equation 29. From this follows

$$\left| \left(G + B\right) \cdot X \right| = \left| X \right| \text{ where } X = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^t-1} \end{pmatrix} \text{ is an arbitrary vector.} \qquad (32)^{13}$$

We will use this result further on in order to simplify the mean and variance. The idea is then to define the b_{ij} in such a way to get

$$P\left(v\right) = \left| G \cdot B^{v-2} \cdot A \right| \text{ for } v > 2 \qquad (33)$$

This distribution is very similar to those we already met (eqs. 3, 18 and 21). It only works if the matrix multiplication $B \cdot A$ yields a probability vector of the same events and in the same order as the A_k. To find B according to this definition is the most difficult part of the whole calculation because it needs to be automated, the number of all the different probabilities increasing exponentially when t increases. Let us analyze the situation first for the case $t = 2$. All involved probabilities of the previous trial are summarized in A and all probabilities of the present trial in F:

$$A=\begin{pmatrix} r & s \\ s & r \\ s & s \end{pmatrix} \text{ and } F=\begin{pmatrix} p & q \\ p & n \\ q & p \\ q & q \\ q & n \\ n & p \\ n & q \\ n & n \end{pmatrix} \tag{34}$$

A unites all possible tuples of the 2-set $\{r, s\}$ less the tuple ($r\,r$) because the related event would have yielded a good nucleotide sequence to the parent in the previous trial. Analogously, F unites all possible tuples of the 3-set $\{p, q, n\}$ less the tuple ($p\,p$) for the same reason but applicable to the actual trial. So each row is a tuple, inside which the probabilities must finally be multiplied with each other because their corresponding events are related by "and". On the other hand, the tuples must be summed with each other because their events are related by "or". Notice that this A is not quite the same as above (eq. 27), where the components of each row are already factorized. Consequently, A above is a vector, whereas here it is a matrix. Notice also that the first column refers to the first nucleotide and so on.

Since the actual trial depends on the previous one, the probabilities of F can be considered functions of r and s. For instance, if previously there was R for some nucleotide of the parent and actually M happens, there will be R in the next trial. This can be expressed with $p(r)=r$. So the argument r of the actual probability p is the probability of the previous event, while the function value is the probability of the actual trial. Let us define all these functions:

$$p(r)=r,\ p(s)=r,\ q(r)=s,\ q(s)=s,\ n(r)=r,\ n(s)=s$$

Now, for each nucleotide, that is for every column, all possible combined outcomes of the previous and actual trial must be evaluated. For that purpose every probability of a column of A is taken as argument for every probability of the corresponding column of F, which yields a tensor T of order 3, that is, a list of 3×2 matrices:

$$T=\left\{\begin{pmatrix} p(r) & q(s) \\ p(s) & q(r) \\ p(s) & q(s) \end{pmatrix} \begin{pmatrix} p(r) & n(s) \\ p(s) & n(r) \\ p(s) & n(s) \end{pmatrix} \begin{pmatrix} q(r) & p(s) \\ q(s) & p(r) \\ q(s) & p(s) \end{pmatrix} \begin{pmatrix} q(r) & q(s) \\ q(s) & q(r) \\ q(s) & q(s) \end{pmatrix} \begin{pmatrix} q(r) & n(s) \\ q(s) & n(r) \\ q(s) & n(s) \end{pmatrix} \begin{pmatrix} n(r) & p(s) \\ n(s) & p(r) \\ n(s) & p(s) \end{pmatrix} \begin{pmatrix} n(r) & q(s) \\ n(s) & q(r) \\ n(s) & q(s) \end{pmatrix} \begin{pmatrix} n(r) & n(s) \\ n(s) & n(r) \\ n(s) & n(s) \end{pmatrix}\right\}$$
$$=\left\{\begin{pmatrix} r & s \\ r & s \\ r & s \end{pmatrix} \begin{pmatrix} r & s \\ r & r \\ r & s \end{pmatrix} \begin{pmatrix} s & r \\ s & r \\ s & r \end{pmatrix} \begin{pmatrix} s & s \\ s & s \\ s & s \end{pmatrix} \begin{pmatrix} s & s \\ s & r \\ s & s \end{pmatrix} \begin{pmatrix} r & r \\ s & r \\ s & r \end{pmatrix} \begin{pmatrix} r & s \\ s & s \\ s & s \end{pmatrix} \begin{pmatrix} r & s \\ s & r \\ s & s \end{pmatrix}\right\}$$

Taking a look at the first matrix of T all functions evaluated, one realizes that all rows are ($r\,s$) and thereby represent an event A_1 for the next trial. But only the first row also represents an event A_1 in the previous trial because the arguments taken to evaluate it are ($r\,s$) as well. So this factorized row, that is $p\,q$, must be added to b_{11} since the index i in b_{ij} stands for the actual and j for the previous trial. The other combinations that had A_1 on the first as well as A_1 on the next trial are the 1st row of the 2nd matrix, the 1st row of the 7th matrix and the 1st row of the 8th matrix. Thereby $b_{11}=p\,q+p\,n+n\,q+n\,n=(n+p)(n+q)$. This must be done likewise for all b_{ij}. On the other hand, the 2nd row of the 2nd matrix yields ($r\,r$). This result must be sorted out because it represents a combination of events that yield good nucleotides to the new parent. The same is true for the 1st row of the 6th matrix.

Thus, here one sees that this is completely irregular. No pattern is recognizable that could be exploited, even if the matrices are rearranged. So for higher t it is impossible to do this calculation manually. In the general case, T indeed contains $t\left(2^t-1\right)\left(3^t-1\right)$ possible events. In the following, a solution is presented how this task can be automated with a code that is understandable but runs slowly. Care for rapidity is taken later. The first thing to do is to program the code for A and F. This could be done with the Mathematica function *Tuples*, but this does not sort A in a preferable way. We will understand further on why the order in A is important. For F however, it can be used, as the order does not matter:

Code 7

```
A={};
Do[
  ls=Join[ConstantArray[0,t-k],ConstantArray[1,k]];
  A=Join[A,Permutations[ls]],
  {k,1,t}];
F=Drop[Tuples[{p,q,n},t],1];
```

ConstantArray creates a list of 0 of length $t-k$ in the first case and a list of 1 of length k in the second case. The 0 and 1 replace r and s respectively because taking numbers in their place accelerates slightly the computation. From this list all possible permutations are generated, creating a submatrix with the same number of $r=0$ and $s=1$ on its rows. This submatrix is then successively joined to A inside the *Do* loop. F is simply created with *Tuples* in form of a matrix, from which the first row is dropped. Next, T is constructed with the following code:

```
T=Table[Transpose[Table[Map[F[[i,j]],Transpose[A][[j]]],{j,1,t}]],{i,1,3^t-1}];
```

The function *Map* is responsible for transferring the columns of arguments of A to every single p, q and n of the corresponding columns of F. For this purpose, A has to be transposed. The outer and inner *Table* iterate i and j through their corresponding domains, which generates all possible combinations. First iterating j and then i directly produces the transpose. The final step is to verify whether a certain row j of a matrix k of T is equal to a certain A_i in which case it is added to the corresponding b_{ij}:

Code 8

```
1   P=Table[Apply[Times,F[[k]]],{k,1,3^t-1}];
2   p[0]=0;p[1]=0;q[0]=1;q[1]=1;n[0]=0;n[1]=1;
3   b[i_,j_]:=0;
4   Do[
5    If[T[[k,j]]==A[[i]],b[i,j]=b[i,j]+P[[k]]],
6    {k,1,3^t-1},{i,1,2^t-1},{j,1,2^t-1}]
7   B=Table[b[i,j],{i,1,2^t-1},{j,1,2^t-1}]//Simplify
```

On line 1, the vector P is created, containing the products of the components of a row of F. These products are added to the b_{ij} on line 5 if the above-mentioned test is fulfilled. On line 2, p, q and n are defined as functions of $r=0$ and $s=1$. The matrix elements of B are initialized to zero on line 3. This needs to be done because the elements of P are added to them on line 5 inside the *Do* loop between lines 4 to 6. Finally, B is created and simplified on line 7. This code yields the following B for $t=2$:

$$B=\begin{pmatrix} (n+p)(n+q) & pq & p(n+q) \\ pq & (n+p)(n+q) & p(n+q) \\ q(n+q) & q(n+q) & (n+q)^2 \end{pmatrix}$$

For $t=3$, B is a square matrix of dimension 8, so already quite big. For $t=7$, its dimension is 127 and it would take several days to calculate it on a PC because the time increases exponentially with t increasing, as its dimension increases in the same way. Fortunately, it is not necessary to calculate all elements of B, some elements can be calculated directly. Nevertheless, rather than optimizing the code, the dimension of A, B and G will be reduced to t. This does not make the previous calculation useless because these matrices are needed in order to be able to reduce their dimension and to calculate them directly in a last step.

2.3 Reducing the dimension

As A is generated with *ConstantArray*, it has several same components (code 5). Since all probabilities are summed in the final calculation, these identical components can be summed from the start. In the case $t=2$ for instance, $A=\{r\,s, r\,s, s^2\}$ becomes $A=\{2\,r\,s, s^2\}$. This reduces the dimension from 2^t-1 to t. The code for generating A this way becomes then

Code 9

```
A=Table[Binomial[t,k] r^(t-k) s^k,{k,1,t}]
```

The same method can be applied for reducing *G*, except that the binomial coefficient must not be used again because it is already used in *A*. As an example, let us reduce the dimension from 3 to 2 of a diagonal matrix and a vector, both of which have each identical first two components. Their product is as follows:

$$\left|\begin{pmatrix} u & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & v \end{pmatrix} \cdot \begin{pmatrix} x \\ x \\ y \end{pmatrix}\right| = \left|\begin{pmatrix} u\,x \\ u\,x \\ v\,y \end{pmatrix}\right| = u\,x + u\,x + v\,y = 2u\,x + v\,y$$

This is the same as

$$\left|\begin{pmatrix} u & 0 \\ 0 & v \end{pmatrix} \cdot \begin{pmatrix} 2x \\ y \end{pmatrix}\right| = \left|\begin{pmatrix} 2u\,x \\ v\,y \end{pmatrix}\right| = 2u\,x + v\,y$$

and shows that the binomial coefficient $C_1^2 = 2$ must only be used once, either on *A* or *G*. Hence code 6 becomes

Code 10

```
G=DiagonalMatrix[Table[p^k (n+p)^(t-k),{k,1,t}]]
```

The reduction of *G* becomes then

$$G = \begin{pmatrix} p(n+p) & 0 & 0 \\ 0 & p(n+p) & 0 \\ 0 & 0 & p^2 \end{pmatrix} \text{ to } \begin{pmatrix} p(n+p) & 0 \\ 0 & p^2 \end{pmatrix}$$

in the case $t = 2$. This does not change the calculation of $P(2)$ a lot as the event G still can only take place if one of the A_k has occurred on the first trial. It is only the number of components that changes from $2^t - 1$ to *t*. Thus with

$$A = \begin{pmatrix} P(\mathrm{A}_1) \\ \vdots \\ P(\mathrm{A}_t) \end{pmatrix} \text{ and } G = \begin{pmatrix} P(\mathrm{G}\,|\,\mathrm{A}_1) & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & P(\mathrm{G}\,|\,\mathrm{A}_t) \end{pmatrix}$$

equation 31 becomes

$$|G \cdot A| = \left|\begin{pmatrix} P(\mathrm{G}\,|\,\mathrm{A}_1)P(\mathrm{A}_1) \\ \vdots \\ P(\mathrm{G}\,|\,\mathrm{A}_t)P(\mathrm{A}_t) \end{pmatrix}\right| = \sum_{k=1}^{t} P(\mathrm{A}_k)P(\mathrm{G}\,|\,\mathrm{A}_k) = P(2) \tag{35}$$

The reduction of *B* is more complicated. Let us see again what happens if we reduce the dimension from 3 to 2 of a general matrix:

$$\left|\begin{pmatrix} u & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & v \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ x \\ y \end{pmatrix}\right| = \left|\begin{pmatrix} u & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & v \end{pmatrix} \cdot \begin{pmatrix} (b_{11} + b_{12})x + b_{13}\,y \\ (b_{21} + b_{22})x + b_{23}\,y \\ (b_{31} + b_{32})x + b_{33}\,y \end{pmatrix}\right|$$

$$= u\left((b_{11} + b_{12} + b_{21} + b_{22})x + (b_{13} + b_{23})y\right) + v\left((b_{31} + b_{32})x + b_{33}\,y\right)$$

This is the same as

$$\left|\begin{pmatrix} u & 0 \\ 0 & v \end{pmatrix}\cdot\begin{pmatrix} b_{11}+b_{12}+b_{21}+b_{22} & b_{13}+b_{23} \\ b_{31}+b_{32} & b_{33} \end{pmatrix}\cdot\begin{pmatrix} x \\ y \end{pmatrix}\right| = \left|\begin{pmatrix} u & 0 \\ 0 & v \end{pmatrix}\cdot\begin{pmatrix} (b_{11}+b_{12}+b_{21}+b_{22})x+(b_{13}+b_{23})y \\ (b_{31}+b_{32})x+b_{33}\,y \end{pmatrix}\right|$$
$$= u\left((b_{11}+b_{12}+b_{21}+b_{22})x+(b_{13}+b_{23})y\right)+v\left((b_{31}+b_{32})x+b_{33}\,y\right)$$

Consequently, B is decomposed into submatrices and their elements summed. These summed submatrices become elements of the reduced matrix

$$B=\left(\begin{array}{cc:c} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ \hdashline b_{31} & b_{32} & b_{33} \end{array}\right) \rightarrow \begin{pmatrix} b_{11}+b_{12}+b_{21}+b_{22} & b_{13}+b_{23} \\ b_{31}+b_{32} & b_{33} \end{pmatrix}$$

Notice that $\{x,x,y\}$ is not reduced to $\{2x,y\}$ but to $\{x,y\}$. This means that in order to get the same result between the totals of the multiplication of unreduced B with $\{x,x,y\}$ and reduced B with $\{2x,y\}$ the first column of reduced B needs to be divided by 2. In the general case, each j^{th} column of the reduced matrix needs to be divided by the binomial coefficient C_j^t.

This decomposition into submatrices must be made within intervals delimited by the binomial factors C_k^t with k going from 1 to t, as A is reduced in the same manner (code 9). In the case $t=2$, these factors are $C_1^2=2$ and $C_2^2=1$. This means that one submatrix of the elements of unreduced B is formed with indices $1\le i\le 2$ and $1\le j\le 2$, a second with $1\le i\le 2$ and $j=3$, a third with $i=3$ and $1\le j\le 2$. The fourth submatrix has only the element b_{33}. For a general t, the last submatrix has always only the element $b_{2^t-1,2^t-1}$ because $C_t^t=1$. In the case $t=3$, the binomial coefficients are $C_1^3=3$, $C_2^3=3$ and $C_3^3=1$, thus B is decomposed and reduced in the following manner:

$$B=\left(\begin{array}{ccc:ccc:c} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & b_{27} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} & b_{37} \\ \hdashline b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} & b_{47} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} & b_{57} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} & b_{67} \\ \hdashline b_{71} & b_{72} & b_{73} & b_{74} & b_{75} & b_{76} & b_{77} \end{array}\right) \rightarrow \begin{pmatrix} \sum_{i,j=1}^{3} b_{ij} & \sum_{i=1}^{3}\sum_{j=4}^{6} b_{ij} & \sum_{i=1}^{3} b_{i7} \\ \sum_{i=4}^{6}\sum_{j=1}^{3} b_{ij} & \sum_{i,j=4}^{6} b_{ij} & \sum_{i=4}^{6} b_{i7} \\ \sum_{j=1}^{3} b_{7j} & \sum_{j=4}^{6} b_{7j} & b_{77} \end{pmatrix}$$

For an arbitrary t, the borders of the indices can be generated with the code

```
c[x_]:=Sum[Binomial[t,k],{k,1,x}]
{1+c[x-1],c[x]}
```

where x stands for both the index i for the rows and j for the columns of the reduced matrix going each from 1 to t. In order to take out of B a corresponding submatrix, one can use

```
Take[B,{1+c[i-1],c[i]},{1+c[j-1],c[j]}]
```

Finally, the elements of these submatrices need to be summed with *Total* and the reduced B generated with *Table*:

```
Table[Total[Take[B,{1+c[i-1],c[i]},{1+c[j-1],c[j]}],2],{i,1,t},{j,1,t}]
```

For $t=2$ this yields

$$B=\begin{pmatrix} 2\left(pq+(n+p)(n+q)\right) & 2p(n+q) \\ 2q(n+q) & (n+q)^2 \end{pmatrix}$$

As mentioned above, the first column of this matrix needs still to be divided with 2 for the final calculation, which yields

$$B=\begin{pmatrix} pq+(n+p)(n+q) & 2p(n+q) \\ q(n+q) & (n+q)^2 \end{pmatrix}$$

2.4 A direct calculation

Before each j^{th} column of the reduced B is divided by C_j^t, we will look for a pattern inside the different B in order to reproduce them directly without reducing them from an unreduced B, for in any case we are not able to go beyond $t=7$ with this method as the time needed for the calculation lies outside reasonable limits. Such a pattern detection is certainly not a very elegant method, but it is possibly the only one. This task is easier if B is let in the intermediate state. So let us consider some of these matrices as functions of t and see if there is a pattern:

$$B(2)=\begin{pmatrix} 2(pq+(n+p)(n+q)) & 2p(n+q) \\ 2q(n+q) & (n+q)^2 \end{pmatrix}$$

$$B(3)=\begin{pmatrix} 3(n+p)(2pq+(n+p)(n+q)) & 3p(pq+2(n+p)(n+q)) & 3p^2(n+q) \\ 3q(pq+2(n+p)(n+q)) & 3(n+q)(2pq+(n+p)(n+q)) & 3p(n+q)^2 \\ 3q^2(n+q) & 3q(n+q)^2 & (n+q)^3 \end{pmatrix}$$

$$B(4)=\begin{pmatrix} 4(n+p)^2(3pq+(n+p)(n+q)) & 4p(n+p)(3pq+3(n+p)(n+q)) & 4p^2(pq+3(n+p)(n+q)) & 4p^3(n+q) \\ 4(n+p)q(3pq+3(n+p)(n+q)) & 6\left(p^2q^2+4p(n+p)q(n+q)+(n+p)^2(n+q)^2\right) & 6p(n+q)(2pq+2(n+p)(n+q)) & 6p^2(n+q)^2 \\ 4q^2(pq+3(n+p)(n+q)) & 6q(n+q)(2pq+2(n+p)(n+q)) & 4(n+q)^2(3pq+(n+p)(n+q)) & 4p(n+q)^3 \\ 4q^3(n+q) & 6q^2(n+q)^2 & 4q(n+q)^3 & (n+q)^4 \end{pmatrix}$$

At first, one discovers that all matrix elements are products of two factors (red on the left and black on the right). The red factors can easily be reproduced with the following code:

Code 11

```
f[i_,j_]:=If[(t-i-j)>=0,(n+p)^(t-i-j),(n+q)^-(t-i-j)];
b[i_,j_]:=If[i<=j, p^(j-i) f[i,j] Binomial[t,i], q^(i-j) f[i,j] Binomial[t,j]];
Table[b[i,j],{i,1,t},{j,1,t}]
```

For $t=3$ for instance, this yields the following matrix of the corresponding red factors:

$$\begin{pmatrix} 3(n+p) & 3p & 3p^2(n+q) \\ 3q & 3(n+q) & 3p(n+q)^2 \\ 3q^2(n+q) & 3q(n+q)^2 & (n+q)^3 \end{pmatrix}$$

This code is confirmed for all matrices up to $t=7$. It is thereby unlikely that for higher t some other property will manifest itself and invalidate the code, although one never knows. This is in fact not a rigorous mathematical proof. However, I renounce to deliver such a proof because of its complexity. In any case, the analytic probability distribution will be compared with the numerical simulation and one will find that there is a neat agreement between both.

For the sake of rendering the matrices more readable, the red factors are removed from all matrices. Since doing so will leave only ones on the last column and row, they are removed too. Another property is that the matrices are symmetric with regard to the black factors. This is why, only the upper tridiagonal part is considered. All this can be performed with the inverse functions of code 11:

Code 12

```
invf[i_,j_]:=If[(t-i-j)>=0,(n+p)^-(t-i-j),(n+q)^(t-i-j)];
invb[i_,j_]:=If[i<=j,p^(i-j) invf[i,j] B[[i,j]]/Binomial[t,i],0];
Table[invb[i,j],{i,1,t-1},{j,1,t-1}]//Simplify//MatrixForm
```

The result can be seen on page 21 where these simplified matrices are labeled $\widetilde{B}$. In order to better recognize the pattern, the matrix elements have been rearranged and grouped into colored zones: in the red zones of all

Matrices from which the factor generated by code 11 has been removed:

$$\widetilde{B}(2)=\begin{pmatrix}(n+p)(n+q)+pq\end{pmatrix}$$

$$\widetilde{B}(3)=\begin{pmatrix}(n+p)(n+q)+2pq & 2(n+p)(n+q)+pq\\ 0 & (n+p)(n+q)+2pq\end{pmatrix}$$

$$\widetilde{B}(4)=\begin{pmatrix}(n+p)(n+q)+3pq & 3(n+p)(n+q)+3pq & 3(n+p)(n+q)+pq\\ 0 & (n+p)^2(n+q)^2+4pq(n+p)(n+q)+p^2q^2 & 2(n+p)(n+q)+2pq\\ 0 & 0 & (n+p)(n+q)+3pq\end{pmatrix}$$

$$\widetilde{B}(5)=\begin{pmatrix}(n+p)(n+q)+4pq & 4(n+p)(n+q)+6pq & 6(n+p)(n+q)+4pq & 4(n+p)(n+q)+pq\\ 0 & (n+p)^2(n+q)^2+6pq(n+p)(n+q)+3p^2q^2 & 3(n+p)^2(n+q)^2+6pq(n+p)(n+q)+p^2q^2 & 3(n+p)(n+q)+2pq\\ 0 & 0 & (n+p)^2(n+q)^2+6pq(n+p)(n+q)+3p^2q^2 & 2(n+p)(n+q)+3pq\\ 0 & 0 & 0 & (n+p)(n+q)+4pq\end{pmatrix}$$

$$\widetilde{B}(6)=\begin{pmatrix}(n+p)(n+q)+5pq & 5(n+p)(n+q)+10pq & 10(n+p)(n+q)+10pq & 10(n+p)(n+q)+5pq & 5(n+p)(n+q)+pq\\ 0 & (n+p)^2(n+q)^2+8pq(n+p)(n+q)+6p^2q^2 & 4(n+p)^2(n+q)^2+12pq(n+p)(n+q)+4p^2q^2 & 6(n+p)^2(n+q)^2+8pq(n+p)(n+q)+p^2q^2 & 4(n+p)(n+q)+2pq\\ 0 & 0 & (n+p)^3(n+q)^3+9pq(n+p)^2(n+q)^2+9p^2q^2(n+p)(n+q)+p^3q^3 & 3(n+p)^2(n+q)^2+9pq(n+p)(n+q)+3p^2q^2 & 3(n+p)(n+q)+3pq\\ 0 & 0 & 0 & (n+p)^2(n+q)^2+8pq(n+p)(n+q)+6p^2q^2 & 2(n+p)(n+q)+4pq\\ 0 & 0 & 0 & 0 & (n+p)(n+q)+5pq\end{pmatrix}$$

$$\widetilde{B}(7)=\begin{pmatrix}(n+p)(n+q)+6pq & 6(n+p)(n+q)+15pq & 15(n+p)(n+q)+20pq & 20(n+p)(n+q)+15pq & 15(n+p)(n+q)+6pq & 6(n+p)(n+q)+pq\\ 0 & (n+p)^2(n+q)^2+10pq(n+p)(n+q)+10p^2q^2 & 5(n+p)^2(n+q)^2+20pq(n+p)(n+q)+10p^2q^2 & 10(n+p)^2(n+q)^2+20pq(n+p)(n+q)+5p^2q^2 & 10(n+p)^2(n+q)^2+10pq(n+p)(n+q)+p^2q^2 & 5(n+p)(n+q)+2pq\\ 0 & 0 & (n+p)^3(n+q)^3+12pq(n+p)^2(n+q)^2+18p^2q^2(n+p)(n+q)+4p^3q^3 & 4(n+p)^3(n+q)^3+18pq(n+p)^2(n+q)^2+12p^2q^2(n+p)(n+q)+p^3q^3 & 6(n+p)^2(n+q)^2+12pq(n+p)(n+q)+3p^2q^2 & 4(n+p)(n+q)+3pq\\ 0 & 0 & 0 & (n+p)^3(n+q)^3+12pq(n+p)^2(n+q)^2+18p^2q^2(n+p)(n+q)+4p^3q^3 & 3(n+p)^2(n+q)^2+12pq(n+p)(n+q)+6p^2q^2 & 3(n+p)(n+q)+4pq\\ 0 & 0 & 0 & 0 & (n+p)^2(n+q)^2+10pq(n+p)(n+q)+10p^2q^2 & 2(n+p)(n+q)+5pq\\ 0 & 0 & 0 & 0 & 0 & (n+p)(n+q)+6pq\end{pmatrix}$$

Continuation to page 20

matrices, $(n+p)(n+q)$ and pq have the same power and are the only summands of all the elements. In the blue zones, there are three summands in each element, in which the exponent of $(n+p)(n+q)$ decreases from 2 to 0, whereas for pq it increases from 0 to 2. A similar statement can be made for the green zones, where there are four summands in each element, the exponent of $(n+p)(n+q)$ decreasing from 3 to 0 and of pq increasing from 0 to 3.

This can easily be reproduced. To begin, let us only calculate the number of the summands in each zone. For this purpose, the matrices must be divided into a part where the elements of each zone are on rows and another where they are aligned on columns. The code for the rows is as follows:

Code 13

```
b[i_,j_]:=0;
Do[b[i,j]=i+1,{i,1,Floor[t/2]},{j,i,t-i}];
Table[b[i,j],{i,1,t-1},{j,1,t-1}]
```

As usual, the index *i* stands for the number of the row and *j* for the number of the column. On the first line, all matrix elements are initialized to zero. This will allow us to distinguish the elements that are processed further from those that are only initialized. In order to create the first part, *i* is restricted between 1 and the middle of the matrix. Since *t* can be odd, $t/2$ has to be rounded down with *Floor*. On the other hand, *j* depends on *i* because it is restricted between *i* and $t-i$. This creates the following matrix for $t=7$:

$$\begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 3 & 3 & 3 & 3 & 0 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The code for the columns is similar:

Code 14

```
Clear[b]
b[i_,j_]:=0;
Do[b[i,j]=t-j+1,{j,Ceiling[t/2],t-1},{i,t-j+1,j}];
Table[b[i,j],{i,1,t-1},{j,1,t-1}]
```

Here, *j* is restricted between the middle of the matrix and $t-1$. For odd *t* the middle is obtained by rounding up $t/2$ with *Ceiling*. In this code, it is *i* that depends on *j*, being restricted between $t-j+1$ and *j*. This creates the following matrix for $t=7$:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

When finally both matrices are added to each other, one gets the desired result:

$$\begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 3 & 3 & 3 & 3 & 2 \\ 0 & 0 & 4 & 4 & 3 & 2 \\ 0 & 0 & 0 & 4 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

The next step is to define the code for the summands still without the coefficient. Now that how to iterate *i* and *j* is known, this is easy:

Code 15

```
1   s[i_,j_,K_]:=Sum[(p q)^k((n+p)(n+q))^(K-k),{k,0,K}];
2   b[i_,j_]:=0;
3   Do[b[i,j]=s[i,j,i],{i,1,Floor[t/2]},{j,i,t-i}];
4   Do[b[i,j]=s[i,j,t-j],{j,Ceiling[t/2],t-1},{i,t-j+1,j}];
5   Table[b[i,j],{i,1,t-1},{j,1,t-1}]
```

On the first line, the function $s(i,j,K)$ is defined with the usual parameters and the parameter *K*, which sets an upper limit for the iterator *k*. This function creates a sum of $K+1$ summands with the exponent of pq increasing from 0 to *K* and the exponent of $(n+p)(n+q)$ decreasing from *K* to 0 as discussed above. The *Do* loops do exactly the same as above (codes 13 and 14), except that for the first part of the matrix $s(i,j,i)$ is iterated instead of $i+1$ (line 3). On the other hand, $s(i,j,t-j)$ is iterated instead of $t-j+1$ in order to create the second part (line 4).

The greater difficulty is the reproduction of the coefficients. For the sake of clarity, let us rewrite the matrices with only these coefficients in the same order as above:

$$C(3)=\begin{pmatrix} 1,\ 1\cdot 2 & 2,\ 1\cdot 1 \\ & 1,\ 2\cdot 1 \end{pmatrix}$$

$$C(4)=\begin{pmatrix} 1,\ 1\cdot 3 & 3,\ 1\cdot 3, & 3,\ 1\cdot 1 \\ & 1,\ 2\cdot 2,\ 1\cdot 1 & 2,\ 2\cdot 1 \\ & & 1,\ 3\cdot 1 \end{pmatrix}$$

$$C(5)=\begin{pmatrix} 1,\ 1\cdot 4 & 4,\ 1\cdot 6 & 6,\ 1\cdot 4 & 4,\ 1\cdot 1 \\ & 1,\ 2\cdot 3,\ 1\cdot 3 & 3,\ 2\cdot 3,\ 1\cdot 1 & 3,\ 2\cdot 1 \\ & & 1,\ 3\cdot 2,\ 3\cdot 1 & 2,\ 3\cdot 1 \\ & & & 1,\ 4\cdot 1 \end{pmatrix}$$

$$C(6)=\begin{pmatrix} 1,\ 1\cdot 5 & 5,\ 1\cdot 10 & 10,\ 1\cdot 10 & 10,\ 1\cdot 5 & 5,\ 1\cdot 1 \\ & 1,\ 2\cdot 4,\ 1\cdot 6 & 4,\ 2\cdot 6,\ 1\cdot 4 & 6,\ 2\cdot 4,\ 1\cdot 1 & 4,\ 2\cdot 1 \\ & & 1,\ 3\cdot 3,\ 3\cdot 3,\ 1\cdot 1 & 3,\ 3\cdot 3,\ 3\cdot 1 & 3,\ 3\cdot 1 \\ & & & 1,\ 4\cdot 2,\ 6\cdot 1 & 2,\ 4\cdot 1 \\ & & & & 1,\ 5\cdot 1 \end{pmatrix}$$

$$C(7)=\begin{pmatrix} 1,\ 1\cdot 6 & 6,\ 1\cdot 15 & 15,\ 1\cdot 20 & 20,\ 1\cdot 15 & 15,\ 1\cdot 6 & 6,\ 1\cdot 1 \\ & 1,\ 2\cdot 5,\ 1\cdot 10 & 5,\ 2\cdot 10,\ 1\cdot 10 & 10,\ 2\cdot 10,\ 1\cdot 5 & 10,\ 2\cdot 5,\ 1\cdot 1 & 5,\ 2\cdot 1 \\ & & 1,\ 3\cdot 4,\ 3\cdot 6,\ 1\cdot 4 & 4,\ 3\cdot 6,\ 3\cdot 4,\ 1\cdot 1 & 6,\ 3\cdot 4,\ 3\cdot 1 & 4,\ 3\cdot 1 \\ & & & 1,\ 4\cdot 3,\ 6\cdot 3,\ 4\cdot 1 & 3,\ 4\cdot 3,\ 6\cdot 1 & 3,\ 4\cdot 1 \\ & & & & 1,\ 5\cdot 2,\ 10\cdot 1 & 2,\ 5\cdot 1 \\ & & & & & 1,\ 6\cdot 1 \end{pmatrix}$$

Each coefficient of a certain summand is separated from the others by a coma. Furthermore, they have been split into two factors, except for the red numbers because the other factor is always 1. This decomposition into factors helps to find the law behind these numbers. The colors have a different meaning here: the red numbers refer to the coefficients of the summands that contain the factor $p^0q^0=1$, so only the factor $(n+p)(n+q)$ is left over when generated with $s(i,j,K)$. Accordingly, the blue numbers refer to the summands that contain the factor $p^1q^1=pq$ and the green numbers to the summands containing the factor p^2q^2 and so on.

In reality, they are all binomial numbers: on the first row of all matrices, there are $t-1$ red numbers that can be obtained by iterating C_{j-1}^{t-1} for $1\le j\le t-1$. In the case $t=3$ for instance this yields $C_0^2=1$ and $C_1^2=2$, in the case $t=4$ one has $C_0^3=1$, $C_1^3=3$ and $C_2^3=3$, and so on. On the second row, they can be obtained by iterating

C_{j-2}^{t-2} for $2 \le j \le t-1$. In the general case, they are obtained by iterating C_{j-i}^{t-i} for $i \le j \le t-1$. In order to get the other colored numbers (so this does not concern the black numbers), only a slight modification is needed by introducing the parameter *k*: instead of C_{j-i}^{t-i} we use C_{j-i+k}^{t-i}, while the domain of iteration remains the same. Hence, $k=0$ for the red numbers, $k=1$ for the blue numbers, $k=2$ for the green numbers, and so on. For their reproduction at the right places inside the matrices, code 15 can be used by replacing the function $s(i,j,K)$ with

Code 16

```
s[i_,j_,K_]:=Table[Binomial[t-i,j-i+k],{k,0,K}]
```

For $t=5$ for instance, the result is the same as for $C(5)$, just the representation differs slightly:

$$\begin{pmatrix} \{1,4\} & \{4,6\} & \{6,4\} & \{4,1\} \\ 0 & \{1,3,3\} & \{3,3,1\} & \{3,1\} \\ 0 & 0 & \{1,2,1\} & \{2,1\} \\ 0 & 0 & 0 & \{1,1\} \end{pmatrix}$$

The black numbers are also binomial coefficients, but they do not appear as a series in a row nor a column, which is why they must be reproduced by iterating C_{i-k}^{i}. Replacing the function $s(i,j,K)$ with

Code 17

```
s[i_,j_,K_]:=Table[Binomial[i,i-k],{k,0,K}]
```

the result for $t=7$ is:

$$\begin{pmatrix} \{1,1\} & \{1,1\} & \{1,1\} & \{1,1\} & \{1,1\} & \{1,1\} \\ 0 & \{1,2,1\} & \{1,2,1\} & \{1,2,1\} & \{1,2,1\} & \{1,2\} \\ 0 & 0 & \{1,3,3,1\} & \{1,3,3,1\} & \{1,3,3\} & \{1,3\} \\ 0 & 0 & 0 & \{1,4,6,4\} & \{1,4,6\} & \{1,4\} \\ 0 & 0 & 0 & 0 & \{1,5,10\} & \{1,5\} \\ 0 & 0 & 0 & 0 & 0 & \{1,6\} \end{pmatrix}$$

In the final form, $s(i,j,K)$ is replaced with

```
s[i_,j_,K_]:=Sum[Binomial[t-i,j-i+k] Binomial[i,i-k] (p q)^k((n+p)(n+q))^(K-k),{k,0,K}]
```

In this code, the generation of the colored and black numbers (codes 16 and 17) are integrated in the sum of code 15. The next step is to restore the factors that have been removed with code 12 and to divide all columns with C_j^t as discussed on page 19. For this purpose, a slightly modified version of code 11 is used, of which the function *f* remains the same. On the other hand, as the function *b* makes a distinction between the lower and upper tridiagonal part (where $i \le j$ is valid), it can directly be integrated into the *Do* loops of code 15, as these only create the upper part. Therefore, these both loops become

```
Do[b[i,j]=p^(j-i) f[i,j] Binomial[t,i]/Binomial[t,j] s[i,j,i],{i,1,Floor[t/2]},{j,i,t-i}];
Do[b[i,j]=p^(j-i) f[i,j] Binomial[t,i]/Binomial[t,j] s[i,j,t-j],{j,Ceiling[t/2],t-1},{i,t-j+1,j}];
```

The binomial coefficients could be simplified, but for the sake of clarity they are let as they are. This will be done in a compilable version (see codes.nb). Next, the last column is calculated, which does not need to be divided by C_j^t because $j=t$ and $C_t^t=1$:

```
Do[b[i,t]=p^(t-i) f[i,t] Binomial[t,i],{i,1,t}];
```

Finally, the lower tridiagonal part (without the diagonal) from the matrix elements in the upper part is calculated. For this purpose, some algebra needs to be done: let b_{ij} be the elements of the final matrix we are looking for and c_{ij} the elements of the intermediate matrix, of which the j^{th} column is still not divided by C_j^t as discussed on page 20. As a consequence, one gets

$$b_{ij} = c_{ij} \Big/ \binom{t}{j} \Rightarrow c_{ij} = b_{ij} \binom{t}{j} \tag{36}$$

From these c_{ij} factors were removed using code 12, which generated a symmetric matrix. Let s_{ij} be the elements of this matrix. According to code 11, these removed factors can be restored in the upper tridiagonal part (where $i \le j$ is valid) with the equation $c_{ij} = s_{ij}\, p^{j-i} f(i,j) C_i^t$, thus

$$s_{ij} = c_{ij}\, p^{i-j}\, invf(i,j) \Big/ \binom{t}{i} \tag{37}$$

where $invf(i,j)$ is defined in code 12. On the other hand, $s_{ji} = s_{ij}$ in a symmetric matrix. And as the factors in the lower part (where $i > j$ is valid) must be restored by multiplication of each element with $q^{i-j} f(i,j) C_j^t$ according to code 11, one gets

$$c_{ij} = s_{ji}\, q^{i-j} f(i,j) \binom{t}{j} \tag{38}$$

Substituting equation 37 into 38 yields

$$c_{ij} = c_{ji}\, p^{j-i}\, invf(j,i) \Big/ \binom{t}{j} q^{i-j} f(i,j) \binom{t}{j} = c_{ji} \left(\frac{q}{p}\right)^{i-j} = b_{ji} \binom{t}{i} \left(\frac{q}{p}\right)^{i-j}$$

where equation 36 and $invf(j,i) f(i,j) = 1$ are used, which is also valid when the arguments are inverted. Finally, the elements of the lower part can be calculated from the elements of the upper part by using

$$b_{ij} = c_{ij} \Big/ \binom{t}{j} = \left(\frac{q}{p}\right)^{i-j} \binom{t}{i} \Big/ \binom{t}{j} b_{ji}$$

Consequently, the code for this operation is as follows:

```
Do[b[i,j]=(q/p)^(i-j) Binomial[t,i]/Binomial[t,j] b[j,i],{i,2,t},{j,1,i-1}];
```

We are now in a position to define the function that yields the matrix we are looking for:

Code 18

```
1   Bmatrix[a_,t_,m_]:=Module[{n,p,q,s,f,b},
2    n=1-m; p=m/a; q=m(1-1/a);
3    s[i_,j_,K_]:=Sum[Binomial[i,i-k] Binomial[t-i,j-i+k] (p q)^k ((n+p)(n+q))^(K-k),{k,0,K}];
4    f[i_,j_]:=If[(t-i-j)>=0,(n+p)^(t-i-j),(n+q)^-(t-i-j)];
5    Do[b[i,j]=p^(j-i) f[i,j] Binomial[t,i]/Binomial[t,j] s[i,j,i],{i,1,Floor[t/2]},{j,i,t-i}];
6    Do[b[i,j]=p^(j-i) f[i,j] Binomial[t,i]/Binomial[t,j] s[i,j,t-j],{j,Ceiling[t/2],t-1},{i,t-j+1,j}];
7    Do[b[i,t]=p^(t-i) f[i,t] Binomial[t,i],{i,1,t}];
8    Do[b[i,j]=(q/p)^(i-j) Binomial[t,i]/Binomial[t,j] b[j,i],{i,2,t},{j,1,i-1}];
9   Return[Table[b[i,j],{i,1,t},{j,1,t}]]];
```

On the first line, the function $Bmatrix(a,t,m)$ is defined, taking as arguments the length *a* of the alphabet, the number *t* of nucleotides and the mutation rate *m*. The function *Module* handles all parameters of the curly brackets locally. On line 2, some of these parameters are initialized as defined in section 2.1. The functions *s* and *f* as well as the *Do* loops between lines 5 and 8 have already been discussed. Compared to the previous code, this one is very fast. With some light modifications, it can be optimized, C-compiled (see codes.nb) and allows creating matrices for high values of *t* within reasonable times.

Figure 4 shows some plots tested against numerical simulations using the parameters $a = 5$ and $m = 10^{-4}$ for some *t*. As can be seen, there is a neat accordance between numerical black points calculated with code 4 and colored analytical curves, despite the fact that the numerical data are very disparate. This comes from the slow change of the curves over a large range of about 400'000 generations. Consequently, there are not enough results that can be averaged for a specific *v* in order to make tend w[v]/z accurately towards $P(v)$, even if code 4 runs for hours. However, the data can be smoothed with a Gaussian filter

```
data=Table[w[v]/z,{v,1,V-1}];
ListPlot[GaussianFilter[data,400]]
```

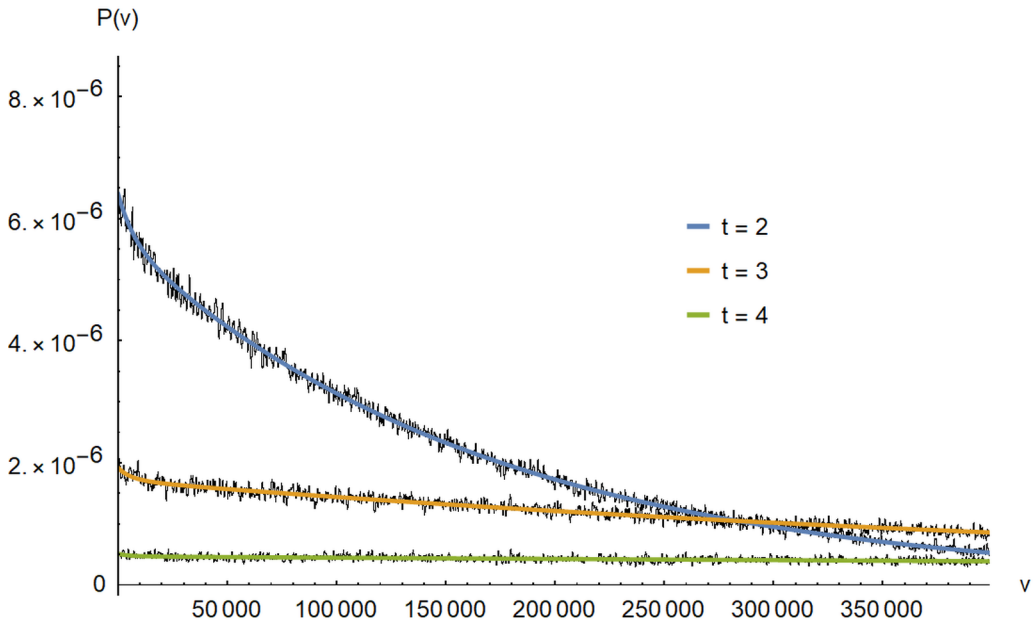

Figure 4: The probability distribution in the case where a sequence of nucleotides of length t mutates in order to achieve a target sequence ($a = 5$, $m = 10^{-4}$).

which takes several points (400 here) and averages them for every generation v. This makes the curves considerably more continuous.

2.5 Mean and variance

In this section, the mean and variance of the number of generations necessary to build functional DNA are calculated based on the distribution defined by equations 24, 31 and 33. Assuming that B^0 yields the identity matrix, one gets

$$P(v) = \begin{cases} \dfrac{1}{a^t} & \text{if } v = 1 \\ \left| G \cdot B^{v-2} \cdot A \right| & \text{if } v > 1 \end{cases} \tag{39}$$

from these equations. We will first show that this is a probability, in other words, that $\sum P(v) = 1$. For this purpose, equation 4 is needed like for the other distributions. In this equation the real number q can be replaced with a squared matrix B, for the operations addition, subtraction and multiplication used in the demonstration of equation 4 are also valid for matrices when care is taken not to commute them when multiplied. On the other hand, division needs to be replaced with the matrix inverse. Therefore, equation 4 yields:

$$\sum_{k=0}^{n} B^k = (I - B)^{-1} \cdot \left(I - B^{n+1}\right) \tag{40}$$

A matrix can only be inverted if its determinant is different from zero, which is only the case if all its columns and rows are linearly independent. This can be shown with a matrix plot, which reveals that B is almost a diagonal matrix. So they are always invertible and consequently also $I - B$. What must also be shown numerically is that $\lim_{n \to \infty} B^{n+1} = 0$ for $n \to \infty$, where 0 means a matrix composed of zeros everywhere (see codes.nb). Therefore, one gets

$$\sum_{k=0}^{\infty} B^k = (I - B)^{-1} \tag{41}$$

Another intermediate result that is needed is:

$$\left|G\cdot\left(I-B\right)^{-1}\cdot X\right|=\left|X\right| \tag{42}$$[14]

From these results as well as equations 29, 39, 41 and 42, one finally can conclude that

$$\sum_{v=1}^{\infty}P(v)=\frac{1}{a^t}+\sum_{v=2}^{\infty}\left|G\cdot B^{v-2}\cdot A\right|=\frac{1}{a^t}+\left|\sum_{v=2}^{\infty}G\cdot B^{v-2}\cdot A\right|=\frac{1}{a^t}+\left|G\cdot\left(\sum_{k=0}^{\infty}B^k\right)\cdot A\right|=\frac{1}{a^t}+\left|G\cdot\left(I-B\right)^{-1}\cdot A\right|$$

$$\left|G\cdot\left(I-B\right)^{-1}\cdot A\right|=\left|A\right|=1-\frac{1}{a^t}\;\Rightarrow\;\sum_{v=1}^{\infty}P(v)=1 \tag{43}$$

For the calculation of the mean, equation 6 is needed like for the other distributions. However, one must first show that this equation can be used for matrices because its demonstration involves derivatives, which in principle are also possible for matrices. However, a more convenient way is to use a proof by induction, which actually yields

$$\sum_{v=1}^{\infty}v\,B^{v-1}=\left(I-B\right)^{-2} \tag{44}$$[15]

With this result, the mean is

$$\mu=\left|\left(I-B\right)^{-1}\cdot A\right|+1 \tag{45}$$[16]

For the variance, equation 9 is adapted for the matrix B without giving a demonstration because the same method by induction can be used, which is left as an exercise. This yields

$$\sum_{v=1}^{\infty}v^2B^{v-1}=\left(I-B\right)^{-3}\cdot\left(I+B\right) \tag{46}$$

and

$$\sigma^2=\left|\left(I-B\right)^{-2}\cdot\left(I+B\right)\cdot A\right|-\left|\left(I-B\right)^{-1}\cdot A\right|^2 \tag{47}$$[17]

We will use these results later in section 4.5 when it comes to the main subject of this article: the dating of the *Pan* / *Homo* split.

3 Several nucleotides and several kids

Let us now go a step further and introduce several kids whose nucleotides can mutate occasionally and independently at each site. As usual, the situation with a numerical simulation is first examined, which in fact reproduces Dawkins' weasel program with a target sentence composed of numbers instead of letters. For this purpose, codes 3 and 4 must be merged together:

Code 19

```
1   ClearAll["Global`*"]
2   a=5; t=2; mn=1; md=100; d=6; z=10^4; V=1000; w=Table[0,{V}]; target=Table[a,{t}];
3   Do[
4    v=1;test=0;parent=RandomInteger[{1,a},t];
5    If[parent==target,test=t];
6    While[test<t,
7     v++;
8     If[v==V,Break[]];
9     kid=Table[0,{d},{t}];score=Table[0,{d}];
10    Do[
11     If[1 <= RandomInteger[{1, md}] <= mn,kid[[i,j]]=RandomInteger[{1,a}],kid[[i,j]]=parent[[j]]];
12     If[kid[[i,j]]==a,score[[i]]++],
13     {i,1,d},{j,1,t}];
14    best=Ordering[score,-1][[1]];
15    parent=kid[[best]];
16    test=score[[best]]];
17   w[[v]]++,
18   {z}]//Timing
19  ls=Table[w[[v]]/z,{v,1,V-1}];
20  ls=GaussianFilter[ls,10];
21  ListPlot[ls]
```

On line 2, *w* and *target* are initialized somewhat differently than in the other codes. The reason for this is that the algorithm must be compatible with the programming language *C++* in order to be able to compile it (see codes.nb). The outer *Do* loop is quite similar to the other simulations: it just repeats the same code *z* times. At the end of every run, a winner *w* is achieved, in which case *w*[[*v*]] is incremented on line 17 after $v-1$ unsuccessful trials inside the *While* loop between lines 6 and 16. This loop is executed as long as $test < t$ is true, where *test* is set to *t* if all of the nucleotides of the parent are correct on the first trial (line 5) or the score of the best kid (line 16). The score of each kid is evaluated on line 12 and is equal to the number of its correct nucleotides. Indeed, *kid* is a $d \times t$ matrix initialized on line 9 with zeros. Each row *i* of this matrix corresponds to kid number *i*, so kid[[i, j]] means nucleotide *j* of kid *i*. This nucleotide is either mutated or copied from the parent inside the *Do* loop between lines 10 and 13. The best kid is evaluated on line 14, where *Ordering* yields the position of the maximal element inside *score*, in other words, the number of the kid with the highest score. This kid then becomes the new parent on line 15 and the *While* loop runs again until one kid gets all nucleotides correct. Finally, the distribution of all winners w[[v]]/z tending to $P(v)$ for big *z* is created on line 19, smoothed on line 20 and the result plotted on line 21.

3.1 An analytical solution to the weasel algorithm

With code 19, we will be able to calculate distributions for $t < 50$, which is still rather slow. But it is rapid enough to test the analytical result, which will run faster. On the first trial, the probability remains the same as before as kids are only generated afterwards, thus $P(1) = 1/a^t$. For $v = 2$, the proceeding is analogous to one nucleotide and several kids: there is the similar event G = "at least one kid got good nucleotides". The only difference is that there one has only one nucleotide, whereas here one has several (see p. 10). And one needs to take into account that G depends on the previous trial, that is, on the events A_k, which are the same as in the case $d = 1$. Therefore, equation 35 can be retaken

$$P(2) = P(\mathrm{G}) = \sum_{k=1}^{t} P(\mathrm{A}_k) P(\mathrm{G} \mid \mathrm{A}_k)$$

even though here the $P(\mathrm{G}\,|\,\mathrm{A}_k)$ are different. In order to find them, one can also use equation 14 to get a similar result to $P(\mathrm{G}) = 1 - q^d$, but here this is not the same q of course. If one defines the event K = "a single kid got correct nucleotides" and its complementary event Q = "a single kid got wrong nucleotides", one gets

$$P(\mathrm{G}\,|\,\mathrm{A}_k) = 1 - P(\mathrm{Q}\,|\,\mathrm{A}_k)^d = 1 - \left(1 - P(\mathrm{K}\,|\,\mathrm{A}_k)\right)^d$$

analogously to the derivation of equation 14 with $P(\mathrm{K}\,|\,\mathrm{A}_k) = p^k (n+p)^{t-k}$ like in code 10. In order to calculate $P(2)$ with matrices such that one gets $P(2) = |H \cdot A|$ in a compact form similar to equation 35, one defines

$$G = \begin{pmatrix} P(\mathrm{K}\,|\,\mathrm{A}_1) & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & P(\mathrm{K}\,|\,\mathrm{A}_t) \end{pmatrix}$$

and the diagonal matrix

$$H = I - (I - G)^d = \begin{pmatrix} 1 - \left(1 - P(\mathrm{K}\,|\,\mathrm{A}_1)\right)^d & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 - \left(1 - P(\mathrm{K}\,|\,\mathrm{A}_t)\right)^d \end{pmatrix} \tag{48}$$

For $v > 2$ things get more complicated as usual. We will proceed in a similar manner than for the calculation of *B*, that is, we will first program a slow but understandable code that just generates the correct result and then determine a direct calculation for the sake of speed. Let us start with $d = 2$ and $v = 3$: the event "no kid got all nucleotides correct" is composed of all possible combinations "B_i and B_j" with B_k = "the outcome of the present trial is equal to A_k for the next trial" as in section 2.2, while *i* is the index of the first and *j* the index of the second kid, both going from 1 to *t*. In this case of failure, the maximum score a kid can achieve must be smaller than *t*. But these events depend on how many nucleotides of its parent are correct. In other words, they depend on all possible A_k the parent achieved in the previous generation. This is why, assuming that all kids have the same parent, all possible events that prevent them from getting the correct nucleotides are "A_k and $(\mathrm{B}_i$ and $\mathrm{B}_j)$" with $i, j, k = 1,...,t$. Hence, there are t^3 events, of which the probabilities are as follows:

$$P(\text{best score} < t) = \sum_{i,j,k=1}^{t} P\left(\mathrm{A}_k \text{ and } (\mathrm{B}_i \text{ and } \mathrm{B}_j)\right) = \sum_{i,j,k=1}^{t} P(\mathrm{A}_k) P(\mathrm{B}_i\,|\,\mathrm{A}_k) P(\mathrm{B}_j\,|\,\mathrm{A}_k) = \sum_{i,j,k=1}^{t} b_{ik}\, b_{jk}\, a_k$$

In order to be able to calculate $P(v)$ easily for $v > 2$, a matrix *F* is constructed that yields

$$P(\text{best score} < t) = |F \cdot A| = \sum_{i,j,k=1}^{t} b_{ik}\, b_{jk}\, a_k = \sum_{k=1}^{t} \left(\sum_{i,j=1}^{t} b_{ik}\, b_{jk} \right) a_k \tag{49}$$

in order to get $P(3) = |H \cdot F \cdot A|$ and finally $P(v) = |H \cdot F^{v-2} \cdot A|$ for any $v > 2$. As in the case $d = 1$, for this being true the multiplication $F \cdot A$ must yield a probability vector of the same events and in the same order as the A_k. To get a first idea what *F* must look like to fulfill this condition, let us rearrange above equation to

$$|F \cdot A| = \left| \begin{pmatrix} \sum_{k=1}^{t} f_{1k}\, a_k \\ \vdots \\ \sum_{k=1}^{t} f_{tk}\, a_k \end{pmatrix} \right| = \left| \sum_{k=1}^{t} \begin{pmatrix} f_{1k} \\ \vdots \\ f_{tk} \end{pmatrix} a_k \right| = \left| \sum_{k=1}^{t} f^k\, a_k \right| = \sum_{k=1}^{t} |f^k|\, a_k \tag{50}$$

where the k^{th} column f^k of *F* is defined and $|X + Y| = |X| + |Y|$ of the 1-norm is used in the case of vectors *X* and *Y* with positive components. Comparing equation 49 with equation 50, one gets

$$\left|f^k\right| = \sum_{i,j=1}^{t} b_{ik}\, b_{jk} \tag{51}$$

This still does not define the k^{th} column of F entirely but only implies that the second index of the $b_{ik}b_{jk}$ must correspond to the k^{th} column, while the first index indicates the score of the actual trial because of $b_{ij} = P\left(\mathrm{B}_i \mid \mathrm{A}_j\right)$. So over the first index, there is still leeway to sort the vector $F \cdot A$ like A, which is sorted according to the score the parent achieved in the previous generation. In other words, the greatest score is achieved when A_1 happens because this event is composed of $t-1$ times the event R according to code 9. Score 0 happens with A_t, which contains no event R. This is why the score decreases with $t-k$ when $k = 1, ..., t$ increases.

Therefore, all summands $b_{ik}b_{jk}$ in equation 51 that represent $score = t - x$ of the actual trial must be summed to form the x^{th} component of f^k. The event $score = t - x$ happens when at least one kid has such a score. The other kids can either have the same or an inferior score. Hence, all summands $b_{ik}\, b_{jk}$ must be summed over all i and j that fulfill $x = i \le j \le t$ and $t \ge i > j = x$ for a fixed x^{th} component of f^k ($i > j$ and not $i \ge j$ in the second inequality because the case $i = j$ is already included in the first inequality). In the case $t = 2$, this yields $f_{1k} = b_{1k}^2 + b_{1k}b_{2k} + b_{2k}b_{1k} = b_{1k}^2 + 2b_{1k}b_{2k}$ for the first component of f^k and $f_{2k} = b_{2k}^2$ for the second. With this rule, the matrix F can be constructed as follows:

$$F = \begin{pmatrix} b_{11}^2 + 2b_{11}b_{21} & b_{12}^2 + 2b_{12}b_{22} \\ b_{21}^2 & b_{22}^2 \end{pmatrix} \tag{52}$$

With a similar calculation in the case $t = 3$ one gets

$$F = \begin{pmatrix} b_{11}^2 + 2b_{11}b_{21} + 2b_{11}b_{31} & b_{12}^2 + 2b_{12}b_{22} + 2b_{12}b_{32} & b_{13}^2 + 2b_{13}b_{23} + 2b_{13}b_{33} \\ b_{21}^2 + 2b_{21}b_{31} & b_{22}^2 + 2b_{22}b_{32} & b_{23}^2 + 2b_{23}b_{33} \\ b_{31}^2 & b_{32}^2 & b_{33}^2 \end{pmatrix} \tag{53}$$

In the following, the second index is temporarily removed to get $b_{ij} \to b_i$ because it is neuter inside a certain column (eq. 51). We first restrict ourselves to the case $t = 2$ but consider a general number d of kids, in which case the building of F needs to be automated: under certain conditions, the sum of products can be factorized. For instance, the well known first binomial formula $b_1^2 + 2b_1b_2 + b_2^2 = \left(b_1 + b_2\right)^2$ can be written as

$$\sum_{i,j=1}^{2} b_i b_j = \left(\sum_{k=1}^{2} b_k\right)^2$$

This is the same as equation 51 in the case $t = 2$ and can also be obtained if the components of a column in matrix 52 are summed. For a general d, this yields

$$\sum_{\alpha_1,\ldots,\alpha_d=1}^{2} \prod_{\beta=1}^{d} b_{\alpha_\beta} = \left(\sum_{k=1}^{2} b_k\right)^d \tag{54}$$

involving a number d of variables α_β instead of i and j. The right side of equation 54 can be expanded using the binomial theorem (eq. 11):

$$\left(\sum_{k=1}^{2} b_k\right)^d = \sum_{i=1}^{d} C_i^d\, b_1^i\, b_2^{d-i} \tag{55}$$

Therefore

$$\sum_{\alpha_1,\ldots,\alpha_d=1}^{2} \prod_{\beta=1}^{d} b_{\alpha_\beta} = \sum_{i=1}^{d} C_i^d\, b_1^i\, b_2^{d-i} \tag{56}$$

This shows that $\prod b_{\alpha\beta}$ corresponds to $b_1^i b_2^{d-i}$ even though there are *d* factors in the first product, whereas there are only 2 in the second. However, the second product is simplified because same factors are written as powers. Also, the number of summands in the first is 2^d, whereas it is only *d* in the second. Likewise, this is because in the second sum they are simplified, that is, same summands are grouped together, for each of which there are C_i^d.

Now, as discussed above, the x^{th} component of an arbitrary column of *F* is extracted by summing $b_i b_j$ over all $x = i \le j \le t$ and $t \ge i > j = x$ in the case $d = 2$. When only over $x = i \le j \le t$ is summed, every $b_i b_j$ is unique because $b_i \ne b_j$ if $i \ne j$. Summing also over $t \ge i > j = x$ creates doubles expect for the case $i = j$, as can be seen in matrix 52. For an arbitrary *d*, this becomes more complicated because the index of every b_{α_β} must be fixed likewise and all others made the same or greater, which creates doubles, triples and so on. Fortunately, this is not necessary to do because of equation 56, which tells us that there are exactly C_i^d of every $b_1^i b_2^{d-i}$. Furthermore, it doesn't matter if $\prod b_{\alpha\beta}$ or $b_1^i b_2^{d-i}$ is taken, while it is preferable to take the first product for reasons that will be understood further on.

Before giving a general formula for the calculation of an arbitrary f_{ij}, we extend *t* to an arbitrary integer. This does not become much more complicated. Furthermore, the second index is restored and label *j*. Thus, equation 54 becomes

$$\sum_{\alpha_1,\ldots,\alpha_d=1}^{t} \prod_{\beta=1}^{d} b_{\alpha_\beta,j} = \left(\sum_{i=1}^{t} b_{ij} \right)^d$$

and the binomial theorem (eq. 55) becomes the multinomial theorem

$$\left(\sum_{i=1}^{t} b_{ij} \right)^d = \sum_{\delta_1+\ldots+\delta_t=d} \binom{d}{\delta_1,\ldots,\delta_t} \prod_{i=1}^{t} b_{ij}^{\delta_i} \tag{57}$$

where

$$\binom{d}{\delta_1,\ldots,\delta_t} = \frac{d!}{\delta_1!\ldots\delta_t!}$$

is the multinomial coefficient with $d = \sum \delta_i$. This replaces equation 56 with

$$\left| f^j \right| = \sum_{\alpha_1,\ldots,\alpha_d=1}^{t} \prod_{\beta=1}^{d} b_{\alpha_\beta,j} = \sum_{\delta_1+\ldots+\delta_t=d} \binom{d}{\delta_1,\ldots,\delta_t} \prod_{i=1}^{t} b_{ij}^{\delta_i} \tag{58}$$

which calculates the 1-norm of the j^{th} column of *F* because the sum goes over all possible α_β like in equation 51. Here again, the sums and products do the same on both sides like in equation 56. On the right side, the terms are just simplified. And it tells us that the multinomial coefficient can be exploited in order to sum only once over all α_β fulfilling $x = \alpha_1 \le \cdots \le \alpha_d \le t$ and to calculate the doubles, triples, and so on with the multinomial coefficient. This is rather a suggestion than a proof. To prove it formally seems to be difficult. Maybe it could be done by induction, but this would certainly be very long. I content myself to show that it works for small *d* and *t*, assuming then that it works for all possible values. Finally, a neat correspondence with the numerical calculations is proof enough for me.

Summing over all α_β fulfilling $x = \alpha_1 \le \cdots \le \alpha_d \le t$ means that the first index is held constant and all others vary correspondingly. Since we want to calculate f_{ij}, $x = \alpha_1$ can be replaced by *i*. The remaining $d-1$ indices are then labeled from α_1 to α_{d-1}. And since *i* is held constant, b_{ij} occurs in every product $\prod b_{\alpha_\beta,j}$ and can therefore be factored out of the sum. Accordingly, the partly guessed solution is

$$f_{ij} = b_{ij} \sum_{i \le \alpha_1 \le \cdots \le \alpha_{d-1} \le t} \binom{d}{\delta_1,\ldots,\delta_t} \prod_{\beta=1}^{d-1} b_{\alpha_\beta,j} \tag{59}$$

Summing over all α_β fulfilling a variable length of inequalities $i \le \alpha_1 \le \cdots \le \alpha_{d-1} \le t$ is not a standard function of Mathematica. Hence, equation 59 needs to be programmed. Translating it into code, one basically gets

Code 20

```
f[i_,j_]:=b[i,j]Sum[c Product[b[α[β],j],{β,1,d-1}],{α[1],α[0],t},...,{α[d-1],α[d-2],t}]
```

where *c* stands for the multinomial coefficient and {α[β], α[β-1], t} means: sum over α[β] from α[β-1] to *t*, which represents the sequence of inequalities. This code is not working yet, of course, because *c* and the sequence of inequalities of variable length are still not defined. This last can almost be done with

```
Table[{α[β],α[β-1],t},{β,1,d-1}]
```

However, the result is not a sequence but a list enclosed inside curly brackets, which need to be removed automatically. This can be done with the function *Apply*, which transfers the elements of a list as arguments to any function. For instance, Apply[f, {a, b, c}] yields f[a, b, c]. In the present case, the list of inequalities is transferred to the function *Sequence*:

```
Apply[Sequence,Table[{α[β],α[β-1],t},{β,1,d-1}]]
```

In order to make believe to *Sum* that this code is a normal sequence, it must be forced with *Evaluate*. Furthermore, the first index α[0] must be initialized to some value otherwise the sum cannot start. In our case this value is *i*.

The calculation of the coefficients is more difficult: when a sum of *t* summands is raised to the power of *d* and expanded, the sum of the exponents for each summand is *d* as well. In the case $t = 3$ and $d = 2$ for instance, expanding $(a+b+c)^2$ yields $a^2+b^2+c^2+2ab+2ac+2bc$, in which the sum of the exponents of the summand $2ab = 2a^1 b^1 c^0$ is $1+1+0=2$. This is also the case for all other summands because each summand in this example is of the form $a^x b^y c^z$ without the coefficients. In the above example, $x=1$, $y=1$ and $z=0$ for $2ab = 2a^1 b^1 c^0$, whereas $x=2$, $y=0$ and $z=0$ for $a^2 = a^2 b^0 c^0$. So there are always $t=3$ different exponents for every summand. In general, the coefficients can be calculated with this sequence of *t* exponents. For instance, Multinomial[1, 1, 0] corresponds to $2ab$ and thereby yields 2, whereas Multinomial[2, 0, 0] corresponds to a^2 and yields 1.

Therefore, the main task consists in calculating the correct sequence of exponents for every summand $\prod b_{\alpha_\beta,j}$ in equation 59. This information cannot be extracted directly from the summand itself like in equation 56 but must be deduced from the indices α_β: when the sum is calculated in code 20, the indices α_β have always certain values for each summand $\prod b_{\alpha_\beta,j}$. If the value of one index α_β is different from each value of the other indices, the exponent of the corresponding $b_{\alpha_\beta,j}$ will be 1. In order to store this information, a unit vector of length *t* is created, corresponding to the input length of the function *Multinomial*: 1 is stored in its α_β^{th} component and 0 elsewhere. This can be done with the following code:

```
UnitVector[t,α[β]]
```

If the values of two indices α_β and α_γ are the same for $\beta \neq \gamma$, then $b_{\alpha_\beta,j} = b_{\alpha_\gamma,j}$ and the exponent of this element will be 2. In order to store this information, a unit vector for every α_β and α_γ is created. If both vectors are added together, one gets a vector of which the α_β^{th} (or $\alpha_\gamma^{\text{th}}$) component is 2 and 0 elsewhere. If this is done for every index of a summand and finally all unit vectors are summed up, one gets a vector of all the exponents of a summand. Since vectors are lists in Mathematica, they are enclosed in curly brackets, which need to be removed as the input of *Multinomial* is a sequence of exponents. This is made with the same method explained above with *Apply*. Consequently, the final code is as follows:

Code 21

```
1   Clear["Global`*"]
2   d=2; t=3;
3   f[i_,j_]:=Module[{α,c,s},
4    α[0]:=i;
5    c:=Multinomial[Apply[Sequence,Sum[UnitVector[t,α[β]],{β,0,d-1}]]];
6    s:=Apply[Sequence,Table[{α[β],α[β-1],t},{β,1,d-1}]];
```

```
7    b[i,j]Sum[c Product[b[α[β],j],{β,1,d-1}],Evaluate[s]]];
8   F=Table[f[i,j],{i,t},{j,t}];
9   F//Expand//MatrixForm
```

On line 2, a certain d and t are initialized, otherwise the code will not work. These can be changed arbitrarily, of course. On line 3, the matrix element f_{ij} is defined with *Module*, which treats the parameters in the curly brackets locally. On line 4, α_0 is set as a function of i because the sum goes over all α_β fulfilling $i \le \alpha_1 \le \cdots \le \alpha_{d-1} \le t$. The multinomial coefficient c is defined on line 5 as explained above and executed on line 7 inside the sum for every summand. The function s creates the sequence of inequalities for the sum, which is the working version of code 20. Finally, the matrix F is generated on line 8. Its elements are expanded and displayed as a matrix (instead of a nested list) on line 9 in order to compare the result with the matrices calculated manually (eqs. 52 and 53).

3.2 A direct calculation

As mentioned, this code is slow, but it allows us to find a pattern that can be exploited in view of finding a direct rapid code, which will be much easier than finding the pattern for the matrix B. For this purpose, one only needs to look at a single column of F because they are all the same except for the second index, which is removed for the sake of clarity. Rather than expanding the components, they are simplified with Mathematica. Let us look at one of these columns in the case $d = 4$ and $t = 6$:

$$\text{column of } F = \begin{pmatrix} b_1\left(b_1^3 + 4b_1^2\left(b_2+b_3+b_4+b_5+b_6\right) + 6b_1\left(b_2+b_3+b_4+b_5+b_6\right)^2 + 4\left(b_2+b_3+b_4+b_5+b_6\right)^3\right) \\ b_2\left(b_2^3 + 4b_2^2\left(b_3+b_4+b_5+b_6\right) + 6b_2\left(b_3+b_4+b_5+b_6\right)^2 + 4\left(b_3+b_4+b_5+b_6\right)^3\right) \\ b_3\left(b_3^3 + 4b_3^2\left(b_4+b_5+b_6\right) + 6b_3\left(b_4+b_5+b_6\right)^2 + 4\left(b_4+b_5+b_6\right)^3\right) \\ b_4\left(b_4^3 + 4b_4^2\left(b_5+b_6\right) + 6b_4\left(b_5+b_6\right)^2 + 4\left(b_5+b_6\right)^3\right) \\ b_5\left(b_5^3 + 4b_5^2 b_6 + 6b_5 b_6^2 + 4b_6^3\right) \\ b_6^4 \end{pmatrix}$$

Regarding at the first component, one sees that it is a product between b_1 and a sum of another product between the binomial coefficients 1, 4, 6, 4 and powers of b_1 as well as of the sum $b_2+b_3+b_4+b_5+b_6$. For the other components, this sum has the b_i with the lowest index successively removed. The binomial coefficients are the same for every component, suggesting that in the general case they are C_h^d for h increasing from 0 to $d-1$. On the left, the first factors of the outer product run from b_1 to b_6, which is why it is straightforward to assume that they run up to b_t for a general t. The same is true for the second factor of the inner product. Furthermore, its exponent decreases from 3 to 0 on each components, in other words, from $d-1$ to 0 for a general d. On the other hand, the exponent of the inner sum increases everywhere from 0 to 3, that is, up to $d-1$ generally. Hence, the code for the matrix element of the first $t-1$ components is easy to program:

Code 22

```
b[i,j] Sum[Binomial[d,h] b[i,j]^(d-h-1) Sum[b[g,j],{g,i+1,t}]^h,{h,0,d-1}]
```

In principle, the last component follows the same pattern, but since it is reduced to b_6^4, it can be generated directly with b_t^d. This is why this element will be calculated separately rather than generated with the same code because this will increase the speed a little bit. Therefore, the final code for the matrix F as a function of t, d and B is defined as follows:

Code 23

```
1   Clear["Global`*"]
2   Fmatrix[t ,d ,B ]:=Module[{f},
3    f[i_,j_]:=If[i<t,
4     B[[i,j]](B[[i,j]]^(d-1) +
5      Sum[Binomial[d,h] B[[i,j]]^(d-h-1) Sum[B[[g,j]],{g,i+1,t}]^h,{h,1,d-2}] +
6      d Sum[B[[g,j]],{g,i+1,t}]^(d-1)),
```

```
7     B[[t,j]]^d];
8   Return[Table[If[f[i,j]<10^-100,0,f[i,j]]//N,{i,1,t},{j,1,t}]]];
```

On lines 2 to 7, the calculation of the matrix element f_{ij} according to code 22 is split into two parts: one for $i<t$ and the other for $i=t$, in which case it is b_t^d (line 7). On the other hand, the outer sum of code 22 has been split into three parts corresponding to the three cases where $h=0$, $0<h<d-1$ and $h=d-1$. This is necessary, otherwise summands of the indefinite form 0^0 are sometimes generated because some elements b_{ij} are set to zero (code 18). On line 8, this is also made for some f_{ij}, which tend to get even littler than the components of the matrix *B*. With the matrices *F* and *H* (eq. 48) thus defined, the single kid distribution 39 becomes

$$P(v)=\begin{cases}\dfrac{1}{a^t} & \text{if } v=1\\ \left|H\cdot F^{v-2}\cdot A\right| & \text{if } v>1\end{cases} \tag{60}$$

The norm of this distribution yields unity, using the same method as for the single kid distribution. The mean and variance are also the same as for the single kid distribution (eqs. 45 and 47). One only has to replace *B* with *F*, even though one should still show that the equations used to derive them are also valid for *H* and *F*, which is left as an exercise.

Henceforth, I will call this probability *weasel distribution*, not only because Dawkins used a sentence in which this animal is mentioned but also because weasels are nice little beasts known for their extreme agility. In German there is the saying "*flink wie ein Wiesel*". Analogously, the weasel distribution is nimble in the sense that it runs much faster than natural selection and thereby its mean can be used as a lower limit for more realistic simulations, as we are going to see. One may tacitly include $P(0)=0$, which mathematically does not change anything to the mean, variance and other quantities. The zeroth generation can be understood as the departing ancestors who are definitely different from future generations, wherefore they have zero probability to detain a genome that is identical to the target. In other words, the probability that a single nucleotide of the target sequence is part of their genome is zero, whereas the probability that the first generation owns such nucleotides is different from zero. But see next section, for a more mathematical approach to the meaning of the first generation.

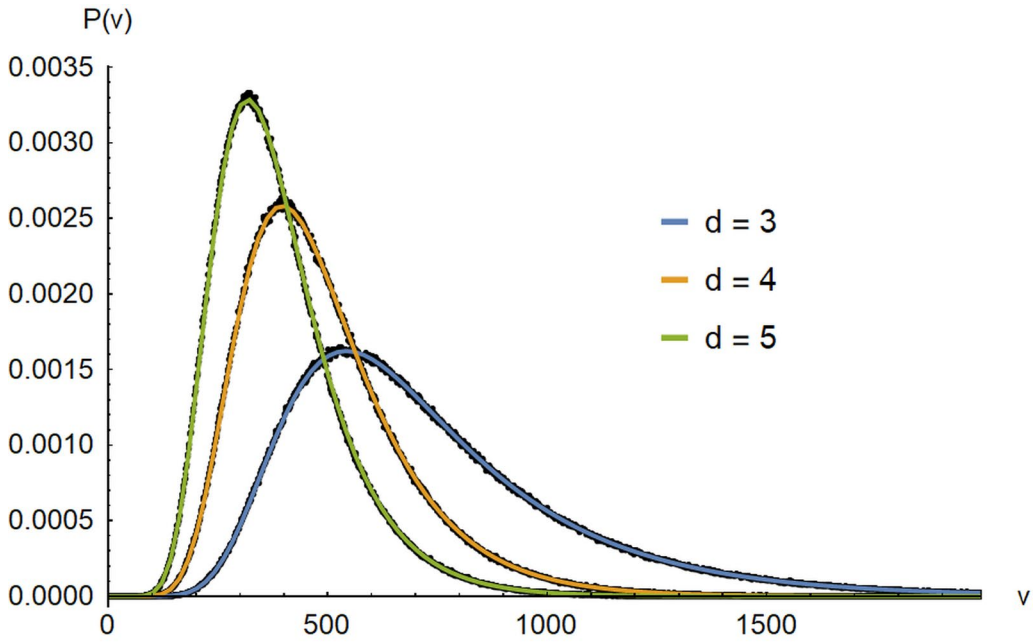

Figure 5: The distribution of the probability to achieve a target nucleotide sequence in the case of several kids. The kid that is closest to the target is favored by natural selection ($a = 5$, $m = 10^{-2}$, $t = 20$).

Let us compare whether the weasel distribution agrees with its numerical simulation (code 19), using the parameters $a = 5$, $m = 10^{-2}$ and $t = 20$ for some values of d. As can be seen in figure 5, there is close agreement between the numerical (black points) and analytical results (colored curves). The numerical data are less hashed here than in figure 4 because the maximal number V of trials only ranges from 0 to about 2000, such that there is no need for smoothing the data. What is also interesting is that peaks slightly skewed to the right appear contrary to figure 4, where the curves immediately decrease from the maximal value (the mode) at $v = 1$. Consequently, if there is more than one kid, the mode is not near the first trials but far from there. What is also obvious is that the mean located slightly on the right of the mode decreases for d increasing. This means that the probability that one of the kids gets the right nucleotides increases as well, which comes as no surprise. On the other hand, the variance decreases for d increasing, which makes it less probable that this event happens outside an increasingly restricted area around the mean.

3.3 Initial variation

One can simulate the assemblage of a nucleotide sequence in two manners: (1) One takes the whole DNA sequence of an ancestor species as initial sequence and then looks after how many generations on average the new sequence of the new species is reached. This implies that a certain number of nucleotides of the initial sequence diverge from the new one at some positions. Given a certain percentage of difference, the number of diverging nucleotides is fixed, their positions in the sequence can vary, though. This divergence is usually very low from one ancestor species to the new one. For instance, the lowest value of genetic difference between chimpanzees and humans mentioned in the literature is about 1% as discussed in section 4.1. So this implies that 99% of all initial nucleotides are already correct. (2) As a compromise, one can also just calculate how many generations on average are necessary to build just the difference, which implies 0% of correct initial nucleotides. This takes less time to calculate but has the inconvenience that one ignores the fact that the already correct nucleotides can also mutate. So this will yield a somewhat biased result.

In the weasel algorithm, the genome of the first parents is generated as a sequence of length t out of $a = 4$ possible nucleotides. As explained on page 7, this yields the binomial distribution $P(k) = C_k^t \, p^k \, q^{t-k}$ where k is

the number of correct nucleotides with $k = 0, ..., t$, $p = 1/a$ and $q = 1 - p$ because the event here is E = "a nucleotide is correct" realized *k* times in the course of building a nucleotide sequence of length *t*. Thereby, *t* trials are carried out. If all nucleotides are correct, one gets $P(t) = C_t^t\, p^t\, q^0 = p^t = 1/a^t$, which corresponds to $P(1)$ of the weasel distribution (eq. 24). As can be shown, the expected number of the binomial distribution is pt, which corresponds to the average number of correct nucleotides[18]. If this quantity is expressed as a percentage of the total of *t* nucleotides, one gets $100\,pt/t\,\% = 25\%$ of correct nucleotides. So for method (1) this value is too low. For method (2) it is too high. Since we want a lower limit of the mean, natural selection is favored and therefore method (2) used because the genetic target will be reached faster if already 25% of nucleotides are initially correct. In addition, ignoring 99% of correct nucleotides, which can also mutate, is accelerating the pace considerably.

However, one could use the weasel distribution in order to start with a given number of correct nucleotides other than 25%. In fact, it is the vector *A* that controls this number: according to code 9, the k^{th} component of *A* is $C_k^t\, r^{t-k}\, s^k$, which stands for the event that $c = t - k$ nucleotides are initially correct. Thus, all possible events are united in the vector *A*. Therefore, if *c* is fixed and the component $k = t - c$ taken from *A*, setting all other components to 0 as follows

$$A_k = A_{t-c} = \begin{pmatrix} 0 \\ \vdots \\ C_{t-c}^t\, r^c\, s^{t-c} \\ \vdots \\ 0 \end{pmatrix} \Rightarrow \left|A_{t-c}\right| = C_{t-c}^t\, r^c\, s^{t-c}$$

one gets a probability distribution that has a fixed $100\,c/t\,\%$ of correct nucleotides on the first generation. The position $k = t - c$ of the non-zero component runs from the bottom to the top for increasing $c = 0, ..., t-1$. Accordingly, code 9 must be replaced with

```
A = Binomial[t,t-c] r^c s^(t-c) UnitVector[t,t-c]
```

The number *c* of correct nucleotides being fixed and below *t*, $P(1) = 0$ because in this case the possibility that the parent gets all nucleotides correct on the first generation is an impossible event. For the numerical result, *parent* must be replaced in code 4 with

```
parent = RandomSample[Join[ConstantArray[a,c], RandomInteger[{1,a-1},t-c]]]
```

This distribution is not normalized, though. In order to normalize it, one can proceed like in the demonstration of equation 43:

$$\sum_{v=1}^{\infty} P(v) = \sum_{v=2}^{\infty} \left|G \cdot B^{v-2} \cdot A_k\right| = \left|\sum_{v=2}^{\infty} G \cdot B^{v-2} \cdot A_k\right| = \left|G \cdot \left(\sum_{k=0}^{\infty} B^k\right) \cdot A_k\right| = \left|G \cdot (I - B)^{-1} \cdot A_k\right| = \left|A_k\right| = C_k^t\, r^{t-k}\, s^k$$

Therefore, the usual distribution must be divided by $C_k^t\, r^{t-k}\, s^k$ in order to get a normalized distribution (see codes.nb for examples). It would be interesting to calculate a lower limit of the average number of generations to build human DNA from scratch, in which case *t* is the entire sequence length, $c = 0$ and $a = 5$, base pair 0 standing for no nucleotide at all. But since this would cross a huge number of different species each with different fertility and mutation rates as well as generation times, this is no easy task.

3.4 Properties of the weasel distribution

We are now able to calculate the mean of the weasel distribution for different parameters in a relatively short time in order to visualize how it behaves when the parameters change. This is best done if only one parameter changes, holding the others constant. First, only the mutation rate *m* will be changed but for different numbers

of kids *d*, from which the difference between a single to a multiple kid situation becomes evident. This can be done with the following code:

Code 24

```
ClearAll["Global`*"]
SetDirectory["d:/nucleotides/functions"];
Get["BmatrixUncp.m"]; Get["FmatrixUncp.m"]; Get["mean.m"];
a = 5; t = 40; d = 2; ls = {};
Do[
 B = BmatrixUncp[a, t, (12/10)^-e];
 F = FmatrixUncp[t, d, B];
 AppendTo[ls, {1.2^-e, mean[a, t, F]}],
 {e, 0, 40}];
SetDirectory["d:/nucleotides/properties"];
name = StringJoin["mean a=", ToString[a], " t=", ToString[t]," m=1.2^-e", " d=", ToString[d], ".txt"];
file = OpenWrite[name];
Write[file, ls];
Close[file]
```

The default directory is set to the location where the functions are stored. Directories must be separated by a slash, so not with the usual backslash. The uncompiled functions are loaded on the next line for this purpose, otherwise the function *mean* produces errors when it calculates the inverse $(I-F)^{-1}$, possibly because some *m* are too close to 1. Inside the *Do* loop, the mutation rate is incremented as $m = base^{-e}$ with an exact base rather than a decimal fraction, otherwise all matrix elements will become approximate decimal fractions and the calculation of the inverse will generate errors. The incrementation is exponential because a log-log plot of the results will be made, the data being very disparate. This will ensure that the distance in the plot from one *m* to the next remains constant. Finally, the result is stored on the hard drive and can be visualized. This code can be adapted easily for the other plots in this section (see codes.nb).

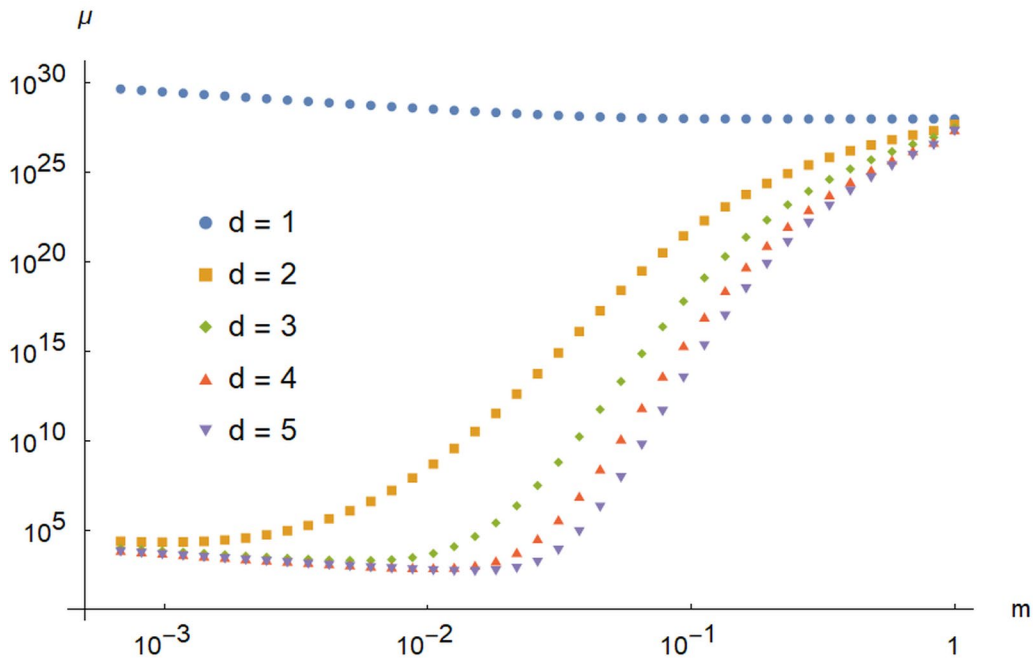

Figure 6: The mean μ decreases dramatically if the number *d* of kids is greater than one and the mutation rate *m* is sufficiently low ($a = 5$, $t = 40$, *m* decreases in steps of 1.2^{-e}).

What is interesting from figure 6 is that for $d > 1$ and sufficiently small *m* there is a dramatic decrease of the mean μ of several orders of magnitude, which confirms that the probability of achieving a good nucleotide sequence increases with *d*. But it is nevertheless surprising that the difference is so important. What can also be seen is that minima appear, one for each $d > 1$, so there is a certain *m* for which the probability to achieve good nucleotides is optimal. This completely contrasts with the curve for $d = 1$, which decreases everywhere for increasing *m*.

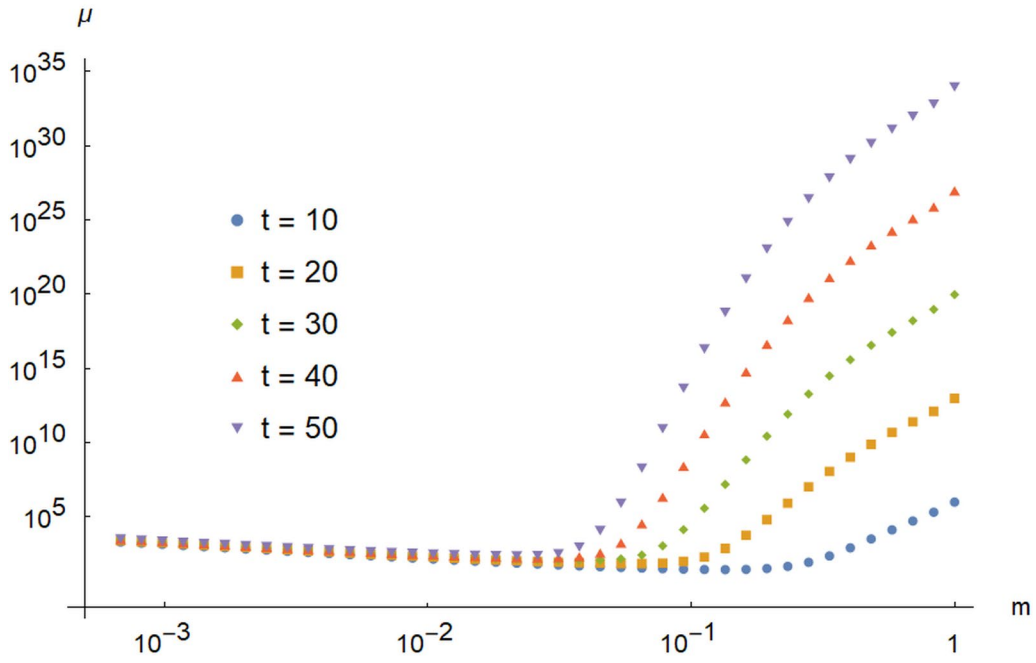

Figure 7: The values of the mean μ come close to each other for sufficiently small m and different t on the left of the minima ($a = 5$, $d = 10$, m increases in steps of 1.2^{-e} for $e = 0,1,...,40$).

Let us now see what happens if we fix $d = 10$ and let vary m for different t. As can be seen in figure 7, this yields a similar plot than in figure 6. The curves seem to approach asymptotically a straight line with negative slope and almost indistinctly lay one over another on the left of the minima, although one would see a difference in a normal plot, of course. It appears that there is only one minimum for each t even though one does not exactly know what happens when m converges to zero.

But it is not likely that the curves would bend down beyond $m = 1.2^{-40}$ to form other minima because the mean tends to infinity when m tends to zero. So one can expect that the straight lines on the left of the minima continue their path to infinity. If a function of x is approaching a straight line in a log-log graph, it is indeed of the form $r\,x^s$ in a normal graph, where s is the slope and $\log_{10} r$ the intercept of the line where $x = 1$ [19]. Therefore, if the slope is negative, the function tends to infinity for $m = x$ going to zero, which is what is expected because a kid will never achieve good nucleotides if these never mutate. In this case, the corresponding probability reduces to zero and the mean is infinite.

One can determine from this whether there is a minimum between two different mutation rates: if $\mu_1(m_1) > \mu_2(m_2)$ for $m_1 < m_2$ and if there is a third mean with $\mu_2(m_2) > \mu_3(m_3)$ and $m_2 < m_3$, one can conclude that the minimum is on the right of the first two means and not between them, even though it cannot be excluded that it may be between the second and third mean.

As the main purpose of this article is to calculate the average number of generations since the *Pan* / *Homo* split, it is paramount to know the behavior of the mean as function of the number of nucleotides t in order to be able to extrapolate it to real values. It is appropriate to do this for m in the region before the minima where the curves almost touch each other (fig. 7) because the mutation rate used is in this region, as we will see. In a normal plot, the mean $\mu(t)$ as a function of t can be approximated by a logarithmic law for sufficiently small m. This can easily be verified if $\mu(t)$ is used as exponent of some base $b(m)$, which yields straight lines. But for different m that are increased exponentially, this yields very disparate data difficult to display. Therefore, what is needed is again a log plot, even though the mean is already a logarithmic law.

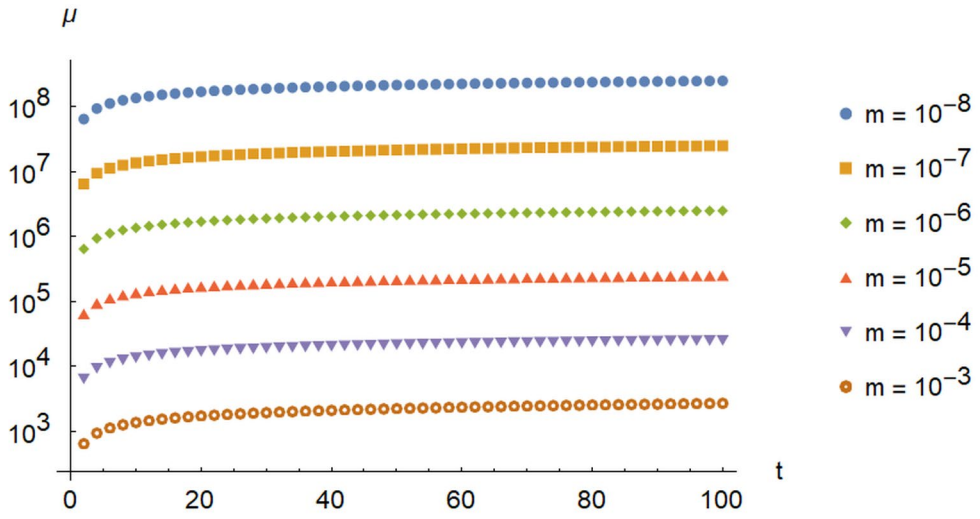

Figure 8: The mean μ increases in the same orders of magnitude as m decreases and the difference of neighboring curves yields approximately the same constant ($a = 5$, $d = 10$).

As can be seen in figure 8, which shows log plots for different m, the distance of each graph to one another is approximately of one order of magnitude corresponding to the exponential decrease of m. It also seems that the difference from one curve to another is constant, which is a consequence of the logarithmic nature of the curves: let $\mu_1(t) = \log_{b_1} t$ and $\mu_2(t) = \log_{b_2} t$ be two neighboring curves with $\mu_1 < \mu_2$. Then their difference in a log plot with arbitrary base is

$$\log \mu_2 - \log \mu_1 = \log\left(\frac{\mu_2}{\mu_1}\right) = \log\left(\frac{\log_{b_2} t}{\log_{b_1} t}\right) = \log\left(\frac{\frac{\log t}{\log b_2}}{\frac{\log t}{\log b_1}}\right) = \log\left(\frac{\log b_1}{\log b_2}\right) = \text{constant}$$

where the logarithm change of base rule is used. It can be shown numerically that every difference of neighboring curves yields approximately the same constant, especially for very small m (see codes.nb), as also suggested by figure 8. This is quite an extraordinary property of the weasel distribution and by the way is a nice example of how order can get out of chaos. This property will be used when it comes to calculating the mean for t corresponding to big genetic differences.

3.5 Genes

Genes can acquire mutations in their sequence, leading to different variants known as alleles and causing different phenotypic traits like eye color, the number of limbs, blood type and so on. Alleles are always restricted to a specific category of traits despite their variation. Genes contain hundreds up to thousands of nucleotides. Human DNA is composed of about 25’000 genes. The largest gene seems to be the TTN gene, containing over 80’000 nucleotides and producing the protein titin, whereas the smallest working code may be considered tRNA molecules that are only 76 base pairs long.

Consequently, several hundreds or thousands of nucleotides must be rearranged in order to produce a functional gene responsible for a new phenotype of another category. This is why a substantial advantage in survival is only granted from the moment when the entire gene is generated. Intermediate steps with partial sequences of nucleotides are not functional and thereby do not give any advantages to an individual. Furthermore, they are often neutral generating no phenotypic functionality (Kimura 1984). The minimal length of a working code must be three because this is the nucleotide number of codons, which are responsible for the encoding of amino acids, the building blocks of proteins.

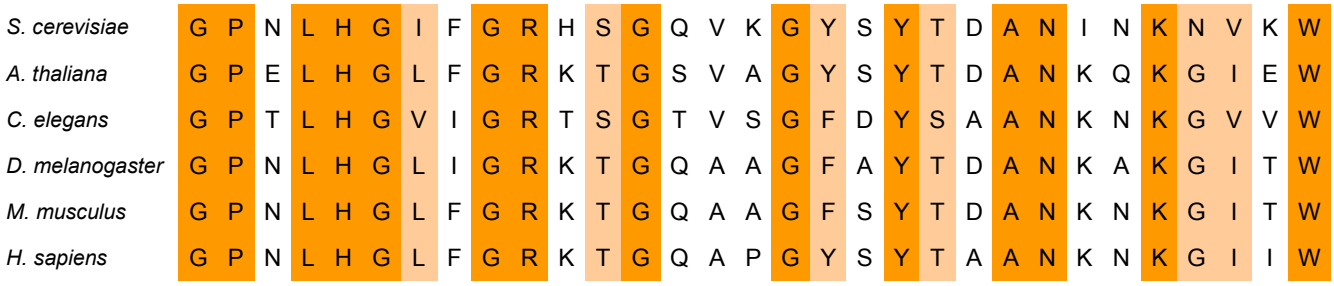

Figure 9: The amino acid sequence for equivalent portions of the cytochrome C protein in six species: *Saccharomyces cerevisiae* (yeast), *Arabidopsis thaliana* (a weed like flowering plants), *Caenorhabditis elegans* (a nematode), *Drosophila melanogaster* (the fruit fly), *Mus musculus* (the house mouse) and *Homo sapiens* (redrawn from Hartwell p. 5).

Figure 9 shows the amino acid sequence for equivalent portions of the cytochrome C protein in six different species (Hartwell 2015 p. 5). Each letter refers to a specific codon responsible for the encoding of an amino acid. As can be seen, there are differences from one species to another. It is assumed that the target sequence only consists of the nucleotides that are different from one species to another. If there are identical nucleotide sequences between differences, they are simply ignored. This is a concept in favor of natural selection: for instance, the probability to achieve the target sequence GPN from an initial sequence GPE is smaller than the probability to simply achieve N from E because the codons G and P are also exposed to mutation (see also section 3.3).

Genes can be simulated if a sequence of t nucleotides is decomposed into blocks of equal length g. The number of genes of a kid is then $t_g = t/g$, which must yield an integer. Accordingly, code 19 must be modified as follows:

Code 25

```
1   ClearAll["Global`*"]
2   g=2; a=4; t=4; mn=1; md=20; d=6; z=10^4; V=1000;
3   w=Table[0,{V}]; tg=Round[t/g]; gene=Table[a,{g}]; target=Table[a,{tg},{g}];
4   Do[
5    v=1;test=0;parent=RandomInteger[{1,a},{tg,g}];
6    If[parent==target,test=tg];
7    While[test<tg,
8     v++;
9     If[v==V,Break[]];
10    kid=Table[0,{d},{tg},{g}];score=Table[0,{d}];
11    Do[
12     If[1 <= RandomInteger[{1, md}] <= mn,kid[[i,j,k]]=RandomInteger[{1,a}],kid[[i,j,k]]=parent[[j,k]]],
13     {i,1,d},{j,1,tg},{k,1,g}];
14    Do[
15     If[kid[[i,j]]==target,score[[i]]++],
16     {i,1,d},{j,1,tg}];
17    best=Ordering[score,-1][[1]];
18    parent=kid[[best]];
19    test=score[[best]]];
20   w[[v]]++,
21   {z}]//Timing
22  ls=GaussianFilter[Table[w[[v]]/z,{v,1,V-1}]//N,10];
23  ListPlot[ls, PlotRange->All]
```

On line 3, the number t_g is introduced as well as the parameter *gene* as a list of *a*'s of length *g*. Only when an evolving gene of a kid corresponds to *gene*, it is advantaged, that is, its score incremented on line 15. This requires that *kid* be set as a nested list of order 3 on line 10. Surprisingly, this has to be done every time again inside the *While* loop, otherwise a wrong result is generated… The mutations, however, still happen on the level of every nucleotide on line 12 (see codes.nb for a compilable and parallel executable version).

It is not necessary to analyze this situation exactly because it can be approximated accurately enough for our purpose with the weasel distribution taking special parameters a_g, t_g and m_g. In fact, a gene can be considered a single nucleotide. If this approximation was exact, one would have $P_g(a,tm) = P(a_g,t_g,m_g)$, where P_g is the

exact distribution of algorithm 25 and P the weasel distribution. In this case, one would have $P_g(1)=1/a^t=P(1)=1/a_g^{t_g}$ and thereby $a_g=a^g$. One obtains the same result with the following reasoning: there are a combinations to form a nucleotide out of an alphabet of length a. So here the possible combinations are equal to a, which is at the same time the length of the alphabet. Now, if one nucleotide is assumed to be a gene of length g, there are a^g possible combinations to form a gene. By analogy, the alphabet length becomes $a_g=a^g$. However, this only true for the first generation because the weasel distribution is not an exact solution of code 25, which is why another way is needed to find a_g and m_g in.

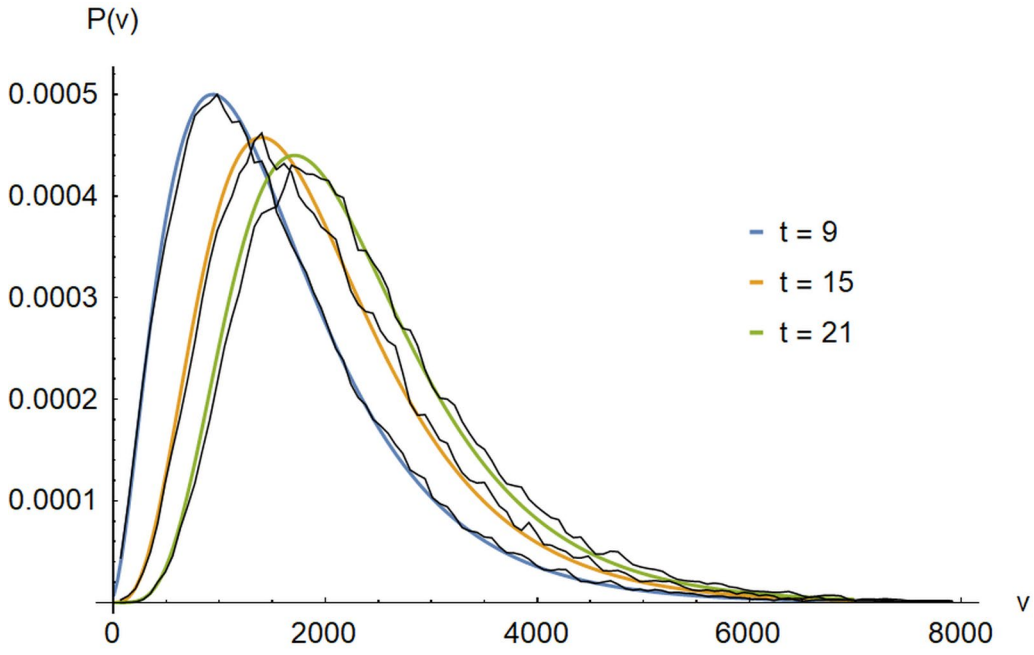

Figure 10: The probability distribution in the case of achieving a target nucleotide sequence under the condition that only phenotypic advantages for survival are produced if genes of length g are generated instead of single beneficial nucleotides. The black curves are calculated numerically with code 25 ($g=3$, $a=4$, $m=0.01$, $d=6$) and the colored curves with the weasel distribution taking $a_g=22.5202932694025$ and $m_g=0.0042457512170479775$.

For a nucleotide sequence of arbitrary length t, the alphabet length a remains always the same for any t. In the same manner, if a gene is considered a single nucleotide, it is assumed that a_g is fixed for an arbitrary $t_g=t/g$ but has a fixed gene length g. The same is assumed for the mutation rate m_g. If this is true for $g=1$, it is not entirely true for $g>1$ but can be assumed as an approximation as shown by numerical results. Thereby, this rule can also be extended to $t=g$, implying that the entire nucleotide sequence is taken as a single gene, in which case $t_g=t/g=1$, that is, there is one nucleotide and several kids as in section 1.3. Therefore, the single nucleotide distribution 21 can be used to find a_g and m_g solved via the equation system of the analytical solutions of the mean 22 and variance 23 equated with numerical values of the mean and variance calculated with code 25. Feeding distribution 60 with values found with this method yields quite good results, as can be seen in figure 10. The relative error is low but seems to increase for increasing t, whereas the approximate analytical result is always lower than the exact numerical result, which is why the first must be considered a lower limit of the second (see codes.nb).

Unfortunately, code 25 even compiled and parallelly executed is too slow parameterized with very low mutation rates of the order of 10^{-8}. Also the use of memory is important. In other words, my PC has not the required performance for this kind of calculation, which is why I am obliged to renounce to it and assume that every single nucleotide change may be beneficial.

4 A model favoring natural selection

Let us now take parameters related to the split between the human and chimpanzee lineages. There is no consensus among scholars when exactly the chimpanzee–human last common ancestor lived. Arnason et al. (1998) place the divergence time between the genera *Pan* and *Homo* at 10 to 13 million years ago (Ma) based on the molecular clock. However, White et al. (2009) think that it happened between 7 and 10 Ma based on the fossil evidence, whereas Wilkinson et al. (2011) combine both methods to arrive at 5.7 to 10 Ma. We will see if these proposed times are enough to produce the genetic difference between humans and chimpanzees.

4.1 Genetic difference, alphabet length and mutation rate

The Chimpanzee Sequencing and Analysis Consortium (Mikkelsen et al. 2005) estimates that this difference includes 35 million single-nucleotide substitutions and 5 million insertions/deletions (indels). In this calculation, what was considered to be so-called junk DNA, that is, sequences that do not encode proteins, has been excluded. However, there is growing awareness that such sequences have other important functions (Makalowski, 2016). This is why there is no consensus either about these numbers. In fact, more recent studies arrive at results with much higher differences (Cohen 2007, Preuss 2012, Tomkins 2013). Nevertheless, conservative parameters favoring natural selection and thereby the lowest mentioned difference are used, that is, 35 million single-nucleotide substitutions by neglecting the other differences because indels are much less frequent than substitutions. In other words, $t = 3.5 \times 10^7$ is taken.

As substitutions occur during meiosis when the two DNA strands are disconnected from each other, this kind of mutations occur on the level of the nucleotides cytosine (C), guanine (G), adenine (A) and thymine (T). According to base pairing rules, however, A is always paired with T, and C with G, which also yields four possible pairs after recombination: AT or TA and CG or GC (Hartwell p. 208). Accordingly, $a = 4$ as the alphabet length is taken, although it could be extended to more if indels were taken into consideration.

The mutation rate is a parameter on which there is no clear consensus either (Callaway 2015). Based on mathematics developed by Kimura (1984) and the genetic difference between humans and chimpanzees, Nachman et al. (2000) estimated the mutation rate to be between 1.3×10^{-8} and 3.4×10^{-8} depending on different parameters like population size, generation time, divergence time and so on. However, as this genetic difference is controversial, other methods are necessary to obtain results that are more accurate.

According to Moran (2016), the error rate of DNA replication is close to 10^{-8} per base pair for cells that are capable of proofreading, by which about 99% of these errors are repaired by enzymes. In other words, only 1% survive. Thus, the overall error rate is about 10^{-10} per base pair. Since the human genome contains about 3.2×10^9 base pairs, there are on average $3.2 \times 10^9 \times 10^{-10} = 0.32$ new substitutions every time the complete genome is replicated.

According to Hartwell et al. (p. 210), women are born with essentially all of the primary oocytes they will ever produce in their entire life. The germ-line cells of a woman need to undergo only 24 mitotic cell divisions to produce all of these oocytes (30 according to Moran). This is why the haploid egg genome produces about $0.32 \times 24 \approx 8$ new mutations. On the other hand, male germ-line cells undergo mitosis continually throughout life with estimated 23 mitotic divisions per year. If according to Moran, one assumes an average of 400 cell divisions, corresponding to an age of about 30 years, the male lineage contribution is $0.32 \times 400 \approx 128$ mutations. Consequently, a fertilized egg contains about 136 new mutations corresponding to a mutation rate of roughly 4×10^{-8}, which thereby is the same per person, that is, per generation.

However, since most new mutations found in the progeny come from the sperm rather than from the egg, the mutation rate is varying between 2×10^{-8} and 4×10^{-8} per generation according to Hartwell. Other studies

arrive even at lower values (Scally et al. 2012). We will see what rate implies a minimal number of generations and choose this rate in favor of natural selection.

4.2 Fertility, birth and mortality rates

Another parameter that is needed to know is the average number of live births of a woman during her life. This is the so-called fertility rate, which has varied in the course of history. It depends on a large number of parameters like diet, health, income, mortality, sociology, religion and so on. The countries with the highest rates today are located in central Africa varying from four to seven children, while in industrialized nations it lies mostly between one and three children according to the CIA World Factbook (2015).

As reported by Schultz et al. (1990), the time from one birth to another among modern forager women varies from three to four years because of the extended period of breastfeeding. This is caused, among other factors, by the foraging diet high in protein and low in carbohydrates. As a result, it lacks soft foods easily digestible by very young infants. By avoiding to find and prepare food for her baby, a mother extends the nursing period. In addition, the nursing stimulus triggers the secretion of a hormone that suppresses ovulation. Thereby, the fertility rate is lowered.

Furthermore, a single child who must be carried for some 3 to 4 years creates a heavy burden for the mother. A second or third child within this interval would create an unmanageable problem for her and jeopardize her health. The caloric requirements of nursing and the physical demands of mobility keep the mother's energy balance low, which can even constitute a net energy drain, resulting in a sharp drop in fertility. This is why the period when a woman is neither pregnant nor nursing becomes essential to building up her energy balance for future reproduction.

With complex foraging and increasing sedentism, these brakes on female fecundity have been eased. A diet increasingly rich in cereals significantly changes the ratio of protein to carbohydrate in the diet and enables mothers to feed their infants soft, high-carbohydrate porridges and gruels, which improves the levels of prolactin, a protein enabling mammals to produce milk, increases the positive energy balance, and leads to a more rapid growth of the infants, shortening thus the interbirth interval, that is, increasing the fertility rate. Based on microdemographic studies of traditional hunting and gathering people, this view is sustained by Bentley (1993) who considers the fertility of foraging groups as relatively low and, by implication, the fertility of agricultural groups as much higher and more variable.

Kaplan et al. (2000) collected demographic data about four modern foraging people from studies made during periods when these hunter-gatherers were almost completely dependent on wild foods, having little modern technology and no firearms, no significant outside interference in interpersonal violence or fertility rates, and no significant access to modern medicine. These studies confirm an average interbirth interval of 3.44 years. The mean time of reproduction lies between the ages of 19.7 and 39 years, which yields a fertility rate of 5.6 births. Bentley et al. (1993) arrive at the same result. As an integer is needed for the fertility rate, this result is rounded up to $d = 6$, which favors natural selection because the expected number of generations decreases if d increases (figure 6).

Kaplan's team also gathered data about chimpanzees, which surprisingly yield a longer average interbirth interval of 5.56 years, whereas the mean age at first reproduction is 14.3 years and the mean age at last reproduction is 27.7 years. This gives a fertility rate of about 2.4, a result that should yield a higher expected number of generations than for humans since the split between the human and chimpanzee lineages. This is why only the data for humans are taken into consideration, as these seem more in favor of low expected generation numbers even though the generation time for chimpanzees is lower than for humans.

While there is virtually known nothing about the causes of death in fossil hominins because few diseases leave any sign of abnormal bone change (Monge and Mann 2015), Paleolithic mortality rates can also be estimated

via modern hunter-gatherer societies. The mortality rate competes with the birth rate, which is not quite the same as the fertility rate. According to Turchin (2003 pp. 20-21.), a simple age-independent exponential population growth model can be derived from the differential equation

$$\frac{dN}{dt} = N\,b - N\,c = N\left(b - c\right) = N\,r$$

where N is the number of individuals of a population at time t, whereas b and c are the proportions of births and deaths per capita and year, in other words, the birth and mortality rates. Correspondingly, the difference between both is the growth rate r, which is zero in a stable population, which can be assumed for Paleolithic people (see next section). This yields the growth function

$$N\left(t\right) = N_0\,e^{r\,t}$$

where N_0 is the number of individuals at time $t = 0$. On the other hand, a growth model based on the total fertility rate d yields a geometric law: when mortality is not taken into account ($r = b$) and there are N_0 individuals at generation 0, there will be

$$N_1 = N_0 + N_0\,p\,d = N_0\left(1 + p\,d\right) = N_0\,q$$

persons at generation 1, where p is the proportion of fertile couples and $q = 1 + p\,d$. Extending this to an arbitrary number k of generations, there will be

$$N_k = N_0\,q^k$$

persons at generation k. Equating this with the exponential law $N\left(k\right) = N_0\,e^{b\,k\,T}$, where T is the generation time in years such that $t = k\,T$ is the time in years, one gets

$$N_0\,q^k = N_0\,e^{b\,k\,T} \;\Rightarrow\; q = e^{b\,T} \;\Rightarrow\; T = \frac{\log q}{b} = \frac{\log\left(1 + p\,d\right)}{b}$$

Assuming a linear age distribution and a stable population ($b = c$), one arrives at $T \approx 21$ years with the data furnished by Kaplan for hunter-gatherers (2000 p. 174). However, repeating the same calculation with the data from chimpanzees, one arrives at an unrealistic $T \approx 3.8$ (see codes.nb). Difficult to say whether the data are biased or the model is too simplistic. In any case, the generation time of chimpanzees, despite their shorter life expectancy, is considered to be only slightly shorter than the one of humans (Langergraber 2012).

Armelagos et al. (1991) argue that the very young and very old are most exposed to disease-related mortality in large agricultural groups because the pathogens are endemic. On the other hand, small hunter-gatherer populations are exposed to a variety of pathogens maintained by surrounding animal species, which is why transmission would be random and the mortality rate more uniformly distributed among all ages. However, the field studies of Kaplan's team and Hill et al. (2007) show higher mortality rates among the very young and very old compared to young and middle-aged adults.

Hill's team also gathered very detailed information about causes of death among the Hiwi (Venezuela), whom they regard as more representative of Paleolithic people. These causes were assigned to four major categories: disease, degenerative/congenital problems, accident, violence. They are age and sex dependent but otherwise occur rather randomly on individuals of the same age or sex. Any increased probability of survival from these causes of death due to genetic dispositions may be regarded as resulting in higher fertility rates of the survivors.

4.3 Population size and number

There are diverging opinions about the size of Paleolithic populations. Coale (1974) estimated that there was presumably an original gene pool of maximal some hundreds of thousands of individuals when the human species gradually became distinct from its hominid predecessors. At the time of writing his paper, it was

believed that this happened about one million years ago. Based on this figure, he calculated an average annual increase of the population size during the first 990'000 years equal to an additional 15 persons per year and million of population to arrive at a size of 8 million at the end of the last glaciation 10'000 years ago. With an exponential population growth of $N(t) = N_0 e^{rt}$, this yields $r \approx 0.00154\%$ per year (see codes.nb). This is a maximal rate as it implies the beginning from single parents. Thereby, if the initial population size was bigger than two persons, as supposed by most scholars, the rate must even have been smaller to arrive at the same size 10'000 years ago when the initiation of agriculture and the domestication of animals led to a dramatic increase of the population size, especially in Mesopotamia. Such a rate is small enough to assume that averaged over large time scales Paleolithic populations have been stable.

At present, the genus *Homo* is considered to be about 2.8 million years old, appearing at first with the species *Homo habilis*, who emerged from the genus *Australopithecus*, which itself had previously split from the lineage of *Pan* at dates discussed above. Extending this split from 1 to 2.8 million years ago makes the growth rate even smaller. This is why Armelagos et al. (1991) question these figures, criticizing that oscillations in size already pointed to by Birdsell (1968) are not given enough attention. Hassan (1978) and Bentley (1993) both cite rates from different sources ranging from 0.00007% to 0.011% per year, while Hassan also provides data based on his own calculations depending on population, area, density and increasing growth rates for the lower, middle and upper Paleolithic, yielding population sizes ranging from 800'000 to 6 million individuals.

Whatever these population sizes have been, let us discuss the question what influence they have on the expected number of generations necessary to achieve a genetic target. As seen in section 1.1, this heavily depends on whether all persons shall reach a certain genetic target or at least one person. An intermediate situation between these both can be imagined when at least a small population reaches a target. Let us first examine the first situation: already Ernst Mayr thought in 1942 that smaller population have better chances to evolve rapidly, for instance, when they are geographically separated from the bulk population. This is what later became known as *allopatric speciation* and is based on the fact that a small minority of genetically advanced persons is not able to bring an entire population on the same level if one supposes that the genetic pool is continually mixed up. Dawkins describes this as follows:

> He [Mayr] believed that, of the two geographically separated races, the original large ancestral population is less likely to change than the new, 'daughter' population (on the other side of the mountains…). This is not just because the daughter population is the one that has moved to new pastures, where conditions are likely to be different and natural selection pressures changed. It is also because there are some theoretical reasons […] for thinking that large breeding populations have an inherent tendency to resist evolutionary change. A suitable analogy is the inertia of a large heavy object; it is hard to shift. Small, outlying populations, by virtue of being small, are inherently more likely, so the theory goes, to change, to evolve (2006 p. 243).

The weasel algorithm is based on a single lineage: the best kid (male or female) tacitly chooses a partner among a tacit population and only his or her genetic information is transmitted to the next generation. So this may rather be compared to reproduction by cell division performed by unicellular organisms. In other words, there is no sexual recombination and its utility compared to a more realistic situation of an entire population, where several lineages are exchanging their biological information, can be questioned. However, it is very useful as a lower limit for more realistic algorithms that take into account a fertility rate per couple depending on their proximity to the genetic target, which follows the evolutionary concept that fitter people produce more offspring (Klein 2009 p. 3).

Genetic variation inside a population also has to be taken into consideration, for that all individuals of a population should be on the exact same genetic level is too strong a requirement. According to Levy et al. (2007), genetic variation is indeed only about 0.5% among humans. Based on this, a more realistic situation of evolution by natural selection can be formulated as follows: let d_k be the fertility rate of the k^{th} couple among

a population of n couples. Each couple has a score s_k, which corresponds to the number of their correct nucleotide sites. We postulate that d_k linearly depends on s_k, therefore $d_k = x\,s_k + y$, where x and y are constants that have to be determined with regard to the arithmetic mean fertility rate $d = 5.6$ and its standard deviation $\sigma_d = 1.386$ (Bentley et al. 1993, Kaplan et al. 2000)[20]. With these requirements one gets

$$d = \langle d_k \rangle = \frac{1}{n}\sum_{k=1}^{n} d_k = x\underbrace{\frac{1}{n}\sum_{k=1}^{n} s_k}_{=s} + y = x\,s + y \;\Rightarrow\; y = d - x\,s$$

where the arithmetic mean score $s = \langle s_k \rangle$ averaged over all couples is used. The variance of the individual fertility rates is

$$\sigma_d^2 = \frac{1}{n}\sum_{k=1}^{n}\left(d_k - d\right)^2 = \frac{1}{n}\sum_{k=1}^{n}\left(d_k^2 - 2\,d_k d + d^2\right) = \frac{1}{n}\sum_{k=1}^{n} d_k^2 - 2\,d\underbrace{\frac{1}{n}\sum_{k=1}^{n} d_k}_{=d} + d^2 = \frac{1}{n}\sum_{k=1}^{n} d_k^2 - d^2$$

$$\begin{aligned}\left\langle d_k^2 \right\rangle &= \frac{1}{n}\sum_{k=1}^{n} d_k^2 = \frac{1}{n}\sum_{k=1}^{n}\left(x\,s_k + y\right)^2 = \frac{1}{n}\sum_{k=1}^{n}\left(x\,s_k + d - x\,s\right)^2 = \frac{1}{n}\sum_{k=1}^{n}\left(x\left(s_k - s\right) + d\right)^2 \\ &= \frac{1}{n}\sum_{k=1}^{n}\left(x^2\left(s_k - s\right)^2 + 2\,x\,d\left(s_k - s\right) + d^2\right) = x^2\underbrace{\frac{1}{n}\sum_{k=1}^{n}\left(s_k - s\right)^2}_{=\sigma_s^2} + 2\,x\,d\,\frac{1}{n}\sum_{k=1}^{n}\left(s_k - s\right) + d^2\end{aligned}$$

$$= x^2\sigma_s^2 + 2\,x\,d\left(\underbrace{\frac{1}{n}\sum_{k=1}^{n} s_k}_{=s} - s\right) + d^2 = x^2\sigma_s^2 + d^2 \;\Rightarrow\; \sigma_d^2 = x^2\sigma_s^2$$

where the standard deviation σ_s of the individual scores s_k is defined. Thereby, $\sigma_d^2 = 0$ in the rare event of $\sigma_s^2 = 0$, in which case x cannot be determined. But in this case $d_k = d\ \forall\ k = 1,\ldots,n$ because only then $\sigma_d^2 = 0$. In the much more likely event of $\sigma_s^2 > 0$ we get

$$x = \frac{\sigma_d}{\sigma_s} \;\Rightarrow\; d_k = \frac{\sigma_d}{\sigma_s}\left(s_k - s\right) + d$$

With these formulas, the following simulation of evolution by natural selection can be programmed:

Code 26:

```
ClearAll["Global`*"]
a=4; t=2; m=0.281; d=6; n=2; z=10^3; V=200;
mr=Rationalize[m]; mn=Numerator[mr]; md=Denominator[mr];
w=ConstantArray[0,V]; limit=Round[0.9*2 n t];
Do[
 v=1;
 lp=RandomInteger[{1,a},{n,2,t}];
 score=Map[Count[Flatten[#],a]&,lp];
 ts=Total[score];
 While[ts<limit,
  v++; If[v==V,Break[]];
  s=ts/n; sgs=Sqrt[Sum[score[[k]]^2,{k,1,n}]/n-s^2]; dls=ConstantArray[Round[d],n];
  If[sgs!=0,Do[dls[[k]]=Round[1.386/sgs (score[[k]]-s)+d],{k,1,n}]];
  kid=ConstantArray[0,{Total[dls],t}];i2=0;
  Do[
   i1=i2+1;i2=i2+dls[[k]];
   Do[
    Do[
     If[1 <= RandomInteger[{1, md}] <= mn,kid[[i,j]]=RandomInteger[{1,a}],
     kid[[i,j]]=RandomChoice[{lp[[k,1,j]],lp[[k,2,j]]}]],
     {j,1,t}],
    {i,i1,i2}],
   {k,1,n}];
  lk=Floor[Length[kid]/2];
  lp=RandomSample[Table[{kid[[k]], kid[[k + lk]]}, {k, 1, lk}], n];
  score=Map[Count[Flatten[#],a]&,lp];
```

```
26    ts=Total[score]];
27   w[[v]]++,
28   {z}];
29  ls=Table[w[[v]]/z//N,{v,1,V-1}];
30  ListPlot[ls,PlotRange->All]
```

On line 2, a number of variables is initialized as usual. The mutation rate *m* is set to a value in order to ensure that $P(1) < P(2)$ of the weasel distribution, otherwise an unrealistic outliner is produced (see codes.nb for details). The maximal possible number of scores is $2nt$ as there are *n* couples and a single person can maximally reach *t* good nucleotides. The variation is 0.5% according to Levy, as seen above. This value is set to a higher value of 10% as this considerably accelerates the calculations. In other words, as long as 90% of the maximal possible number of scores is not reached by the total score of all couples, the genetic target is not achieved. Accordingly, the variable *limit* is defined to the rounded value of $0.9 \times 2nt$ on line 4 because it needs to be an integer. It is used as test for the *While* loop between lines 10 and 26, inside which generations of kids are generated until a sufficient number of them is sufficiently close to the genetic target. When this happens on generation *v*, the winner *w*[[*v*]] is incremented on line 27.

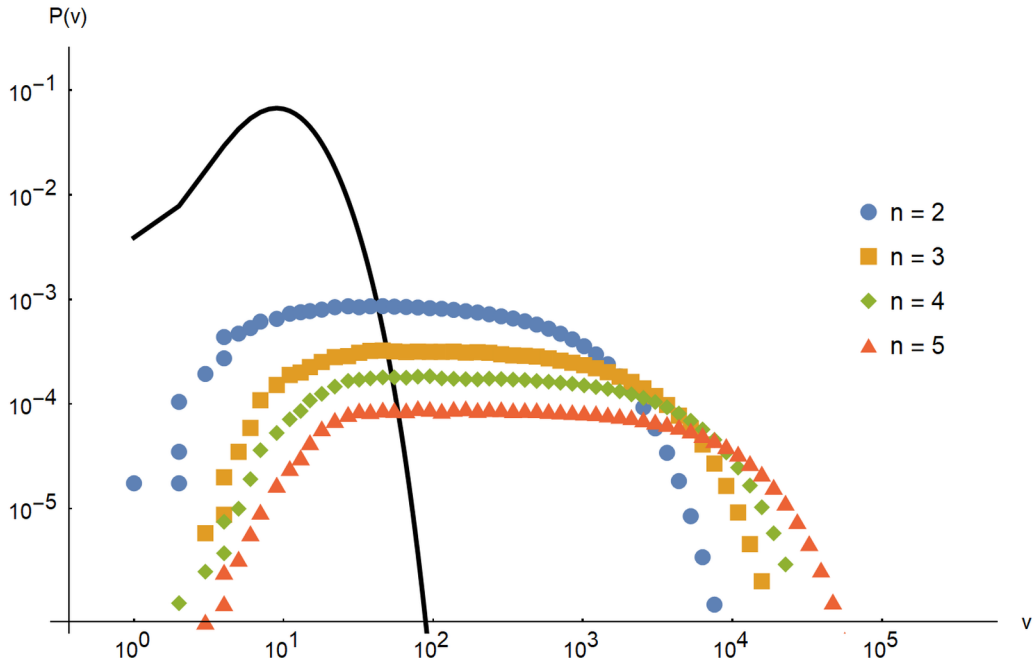

Figure 11: Log-log plots of distributions for different population sizes *n* calculated by code 26 with $a = 4$, $t = 4$, $m = 0.137$ and $d = 6$. As expected, the curves become flatter as *n* increases. Consequently, the mean increases as well. The black curve is the weasel distribution with the lowest mean.

The outer *Do* loop between lines 5 and 28 repeats this calculation *z*-times until a sufficient number of winners is reached. On line 7, the random list *lp* of *n* parents is created and their scores evaluated on the next line, considering that the partner of one parent is simply the neighbor in this list. On line 12, the mean *s* and the standard deviation $\sigma_s = sgs$ of the scores of each couple is evaluated. They are used on the next line in order to create the list *dls* of the number of kids for each couple, using the formulas as discussed above with $\sigma_d = 1.386$.

The *Do* loop between lines 18 and 20 either mutates a nucleotide of a single kid or randomly copies it from either parent. The *Do* loop between lines 17 and 21 ensures that this happens for every kid of the k^{th} couple, taking into account the previously created list *dls*. On the other hand, the *Do* loop between lines 15 and 22 ensures that this happens for all couples. On lines 23 and 24, a new list of parents is created out by halving the kids and superposing them for coupling to prevent incest. Then, a random sample of these kids of length *n* is taken because death is rather random and also hits the fittest, as discussed at the end of section 4.2. Their score

is then evaluated on the next line and the outer Do loop retakes. This code can be compiled and parallel evaluated (see codes.nb).

Results are shown in figure 11, which clearly shows that the distributions for increasing n flatten in such a way that a log-log plot is needed to bring them closer together. The flatter a distribution, the higher the mean and also the calculation time, which is why only values for small n can be calculated on a desktop computer. The black curve is the weasel distribution with the smallest mean near its highest value. Therefore, this figure shows that the weasel distribution can be used as a lower limit for the above algorithm, which is very simple. There are certainly more sophisticated algorithms, but it would come as a very big surprise if one of them provided lower means as those provided by the weasel distribution.

Figure 12 shows the mean values for increasing population sizes n. As can be seen, they are always the lowest at $n=1$ for the weasel distribution. Since these curves are nearly linear in a log plot for $n>1$, they increase exponentially in a normal plot as expected from the math of dice rolling (eq. 13). For larger values of n they would probably slightly deviate downwards from a straight line because a 10% genetic variation is allowed. But here too, it would be very surprising if from some n on this deviation would cause the curves to start decreasing.

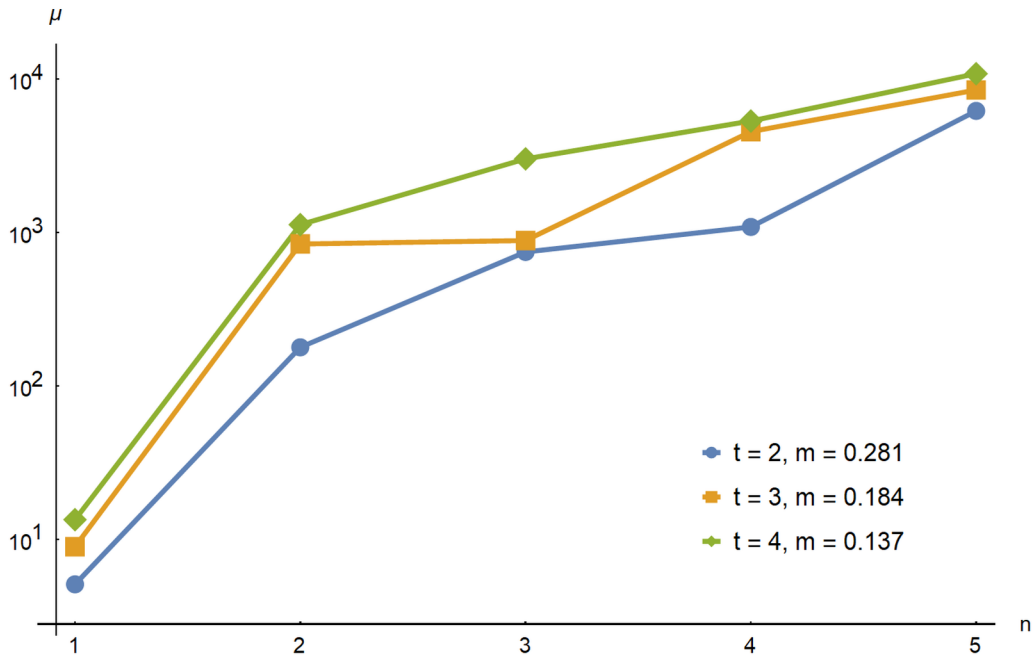

Figure 12: This log plot shows the mean values for different nucleotide lengths t and population sizes n for $a=4$ and $d=6$. The mean of the weasel distribution is at $n=1$. For $n>1$, the increase is nearly linear, which shows that the mean increases exponentially for increasing n. The intersection points of these straight lines with the vertical line where $n=1$ is far above the mean values of the weasel distribution, which shows that it is a lower limit.

4.4 A single parents model

It is estimated that hunter-gatherers moved around in small cohorts of about 25 to 50 persons. In some special cases, where food resources were abundant, the size may have reached several hundred people. These bands were exogamous, that is, they found their mating partners outside the group in a vast genetic pool (Birdsell 1968, Wobst 1974). In an extreme situation, where such cohorts find their partners inside their group, the number of independent populations would be very large and each one has the potential to evolve more quickly than the entire humanity of this time, as seen in the precedent section. Would this heighten the odds of producing new species?

Let us first compare the math of this scenario with dice rolling as discussed in section 1.1: the probability that after *v* trials at least one person gets the right numbers out of a dice rolling population of *c* persons is

$$P(v) = q^{c(v-1)}\left(1 - q^{c}\right)$$

according to equation 15 with $q = 1 - p$ and $p = P(\text{one person rolls the right number}) = 1/a$, where a is the number of regular faces of the die. At the same time, it is also the mean of a single player (eq. 7). Now, if *c* persons are replaced with *c* populations each consisting of *b* persons, one has

$P(\text{all persons of one population roll the right number})$

$= P(\text{the } 1^{st} \text{ person gets the right number and ... and the } b^{th} \text{ person gets the right number}) = p^{b}$

instead of *p*. Therefore $q = 1 - p^{b}$ and

$$P(v) = \left(1 - p^{b}\right)^{c(v-1)}\left(1 - \left(1 - p^{b}\right)^{c}\right)$$

One can show that the mean of this probability is approximately $\mu = a^{b}/c$ (see codes.nb). Let then $n = b \cdot c$ be the total number of persons participating in this thought experiment of dice rolling, wherefore $\mu = b \cdot a^{b}/n$. As we wish to achieve the target faster than a single person rolling a die, of which the mean is *a*, the right part of this equation is set to be smaller than *a* to get $n > b \cdot a^{b-1}$. As we will see in the next section, the expected number of generations provided by the weasel distribution, which is the probability that a single lineage reaches a genetic target, is several billions of generations. Since both the die and weasel distributions have a similar form (eqs. 3 and 60), the mean *a* can be replaced with this number in order to get a rough estimation of what is the minimal number of persons necessary to achieve the target for a given *a*.

As the term $b \cdot a^{b-1}$ increases with *b* increasing, assuming that *a* is held constant and $a > 1$, a minimal value for *b* must be taken. The smallest possible cohort able to reproduce consists of a male and female. Therefore, $b = 2$ and $n > b \cdot a^{b-1} = 2a$, which means that *n* must be twice as big as *a*, that is, bigger than several billions of persons if one assumes that *a* is equal to billions of generations. So this yields more people than presently live on our planet, which is a very rough estimation of the probability of such a scenario of independent populations.

On a genetic level, this scenario is much more complex, of course. It can be simulated by slightly modifying code 26, which has shown that the expected values increase exponentially when the population size grows. Hence, it is straightforward to ask what happens when the best single parents are picked out of a large population. It is indeed legitimate to imagine that the whole of humanity descends from single parents who separated themselves from their cohort, lived somewhere on their own and finally lost any contact with the rest of their compatriots. This natural Adam and Eve model can be simulated as follows:

Code 27:

```
ClearAll["Global`*"]
a=4; t=2; m=0.281; d=6; n=2; z=10^4; V=100;
mr=Rationalize[m]; mn=Numerator[mr]; md=Denominator[mr];
w=ConstantArray[0,V]; limit=Round[0.995*2t];
Do[
 v=1;
 lp=RandomInteger[{1,a},{n,2,t}];
 score=Map[Count[Flatten[#],a]&,lp];
 best=Ordering[score,-1][[1]];
 While[score[[best]]<limit,
  v++; If[v==V,Break[]];
  s=Total[score]/n;
  sgs=Sqrt[Sum[score[[k]]^2,{k,1,n}]/n-s^2];
  dls=ConstantArray[Round[d],n];
  If[sgs!=0,Do[dls[[k]]=Round[1.386/sgs (score[[k]]-s)+d],{k,1,n}]];
  kid=ConstantArray[0,{Total[dls],t}];i2=0;
  Do[
   i1=i2+1;i2=i2+dls[[k]];
   Do[
```

```
20      Do[
21       If[1 <= RandomInteger[{1, md}] <= mn,kid[[i,j]]=RandomInteger[{1,a}],
         kid[[i,j]]=RandomChoice[{lp[[k,1,j]],lp[[k,2,j]]}]],
22       {j,1,t}],
23      {i,i1,i2}],
24     {k,1,n}];
25    lk = Floor[Length[kid]/2];
26    lp = RandomSample[Table[{kid[[k]], kid[[k + lk]]}, {k, 1, lk}], n];
27    score=Map[Count[Flatten[#],a]&,lp];
28    best=Ordering[score,-1][[1]]];
29   w[[v]]++,
30   {z}];
31  ls=Table[w[[v]]/z//N,{v,1,V-1}];
32  ListPlot[ls,PlotRange->All]
```

On line 4, the limit is set to a more realistic value of 99.5% of correct nucleotides that must be reached according to Levy et al. (2007). And as only one couple shall reach the target, the maximum value of the score is only 2 *t* instead of 2 *n t*. On line 9, not the total score of the whole population is taken but the highest score of a single couple. As long as this score is lower than the limit, the *While* loop continues to run. On lines 27 and 28, the code is adapted accordingly.

In this code, a couple must be neighbors in the list of parents (lines 7 and 26). This simulates the fact that there is a geographical distance between couples before they meet each other. One could also take the two best persons in the list and bring them together. But this is not realistic because the bigger a population the more distance occurs between persons, not only geographical distance but also distance by the fact that it is not possible that each person knows each other if the population size is big enough. So the probability that the genetically most advanced couple come together is reduced with an increasing population size.

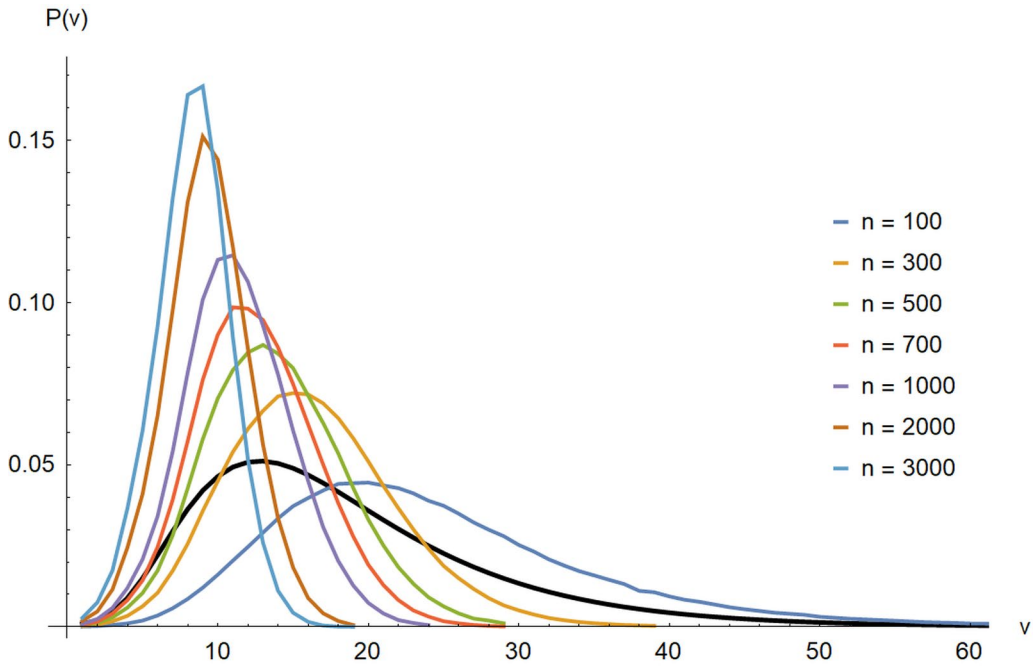

Figure 13: Under a single parents model, a given nucleotide target is reached in less generations *v* than under the weasel model (black curve) for a sufficiently high population size *n*. Parameters are $a = 4$, $t = 5$, $m = 0.109$ and $d = 6$.

As can be seen in figure 13, these distributions outpace the weasel distribution (black curve) for the same parameters and sufficiently high *n*. However, here the weasel distribution must be modified in order to use it as a lower limit. We take $\mu = b \cdot a^b / n$ as discussed above. Quantitatively, this is, of course, not the exact formula for the mean of the above algorithm, but qualitatively it has some similarity with it as both mean decrease with *n* as suggested by figure 13. This is why $\mu = b \cdot a^b / n$ can be used as a lower limit if *a* is replaced by the mean μ_w of the weasel distribution. After all, *a* is the mean of a single die player, whereas μ_w yields analogously the

mean of a single lineage. Since in a single parents model, the number of persons in a population is $b=2$, the formula for the lower limit is therefore $\mu_{\min}=2\mu_w^2/n$ using the same parameters *a*, *t*, *m* and *d* of course.

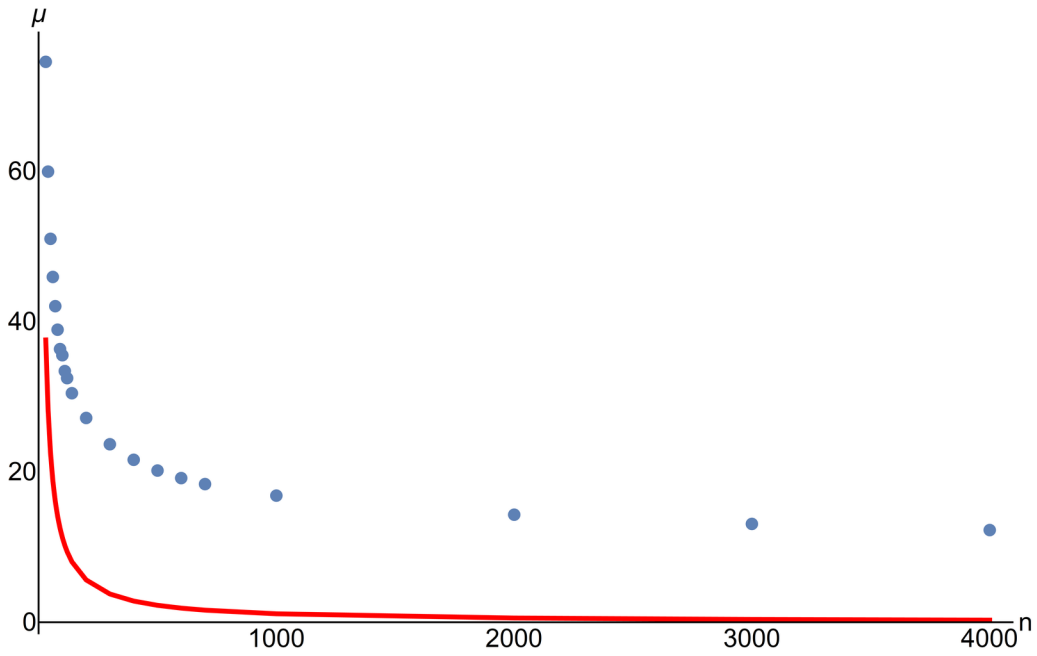

Figure 14: The blue points represent the expected values of the single parents distribution for different *n* using the parameters $a=4$, $t=6$, $m=0.0907$ and $d=6$. The red curve is a lower limit based on the weasel distribution that cannot be crossed by the blue points.

As shown in figure 14, the blue points represent the expected values of the single parent algorithm (code 27) for different *n*, whereas the red line is the lower limit. Is it possible that for very big *n* the blue points get under the red curve? No, because the lower limit converges to zero for increasing *n* (a constant divided by *n*), whereas the blue points converge to a positive value greater than zero. This can be explained by the fact that the single parents distribution is not defined for $v=0$, in other words $P(v=0)=0$, which implies that the mean must be greater than zero. Furthermore, $P(v=1)<1$ as long as *n* is not infinite. Only in the case where *n* tends to infinity, $P(v)$ tends to a Dirac delta distribution, that is, $P(v=1)=1$ and 0 elsewhere because with *n* increasing the probability that one couple reaches the target in the first generation is increasing as well, and there are no brakes to this increase. In that case the mean is equal to 1 too, which therefore is the lowest possible value. But since *n* cannot be infinite, the curve of the blue points must always be well above 1, which can also be seen numerically (see codes.nb).

Taking $\mu_w=6\times10^9$ years as calculated in the next section and rounding up the highest value of Paleolithic population size mentioned in the literature to $n=10^7$, as discussed in the previous section, one gets $\mu_{\min}=2\mu_w^2/n=2\left(6\times10^9\right)^2/10^7=7.2\times10^{12}$ years, which is over 500 times the age of the universe (13.8×10^9 years). Therefore, to the despair of Darwinists, a natural Adam and Eve model – the fastest imaginable realistic scenario – does not have the potential to produce any miracles.

4.5 Dating the Pan / Homo split

With the parameters $a=4$, $t=3.5\times10^7$, $m=4\times10^{-8}$ and $d=6$, we can now proceed to the calculation of the minimal expected number of generations necessary to generate the genetic difference between the *Pan* and *Homo* split according to Dawkins' weasel algorithm. However, a direct calculation of the mean via the matrix *F* of such a huge nucleotide sequence would take too long on a desktop computer. This is why the fact will be exploited that the mean $\mu(t)$ as a function of *t* can be approximated by a logarithmic law, as discussed in section 3.4. Therefore, *B* and *F* are calculated for different *t* and then the corresponding mean in order to be

able to find a logarithmic function $\mu(t) = \log_b(st + c)$ with these data by extrapolation. This can be done within a reasonable time for $t = 100$, 200, 300, 400, 500 and 600 (see codes.nb). Furthermore, this is not only done for $m_2 = 4 \times 10^{-8}$ but also for $m_1 = 3 \times 10^{-8}$ and $m_3 = 5 \times 10^{-8}$ in order to be able to determine where the optimal mutation rate is located.

The result can be seen in figure 15, which shows that on the average about 2.94×10^8 generations are needed to generate a target sequence of $t = 3.5 \times 10^7$ nucleotides, corresponding to the genetic difference between chimps and humans. In order to transform this value into time, the generation time must be known. In section 4.2, we got about 21 years based on a simple model. But there are also other methods based on complex demographic mathematics (Coale 1972 pp. 18-19). Intuitively speaking, the generation time is the average time from the birth of a woman to her reproductive period, which has a certain length and the number of births obeys some distribution, increasing slowly from zero near age 15 following menarche, reaching a maximum around ages 20 to 25 and then slowing down to vanish at about age 50.

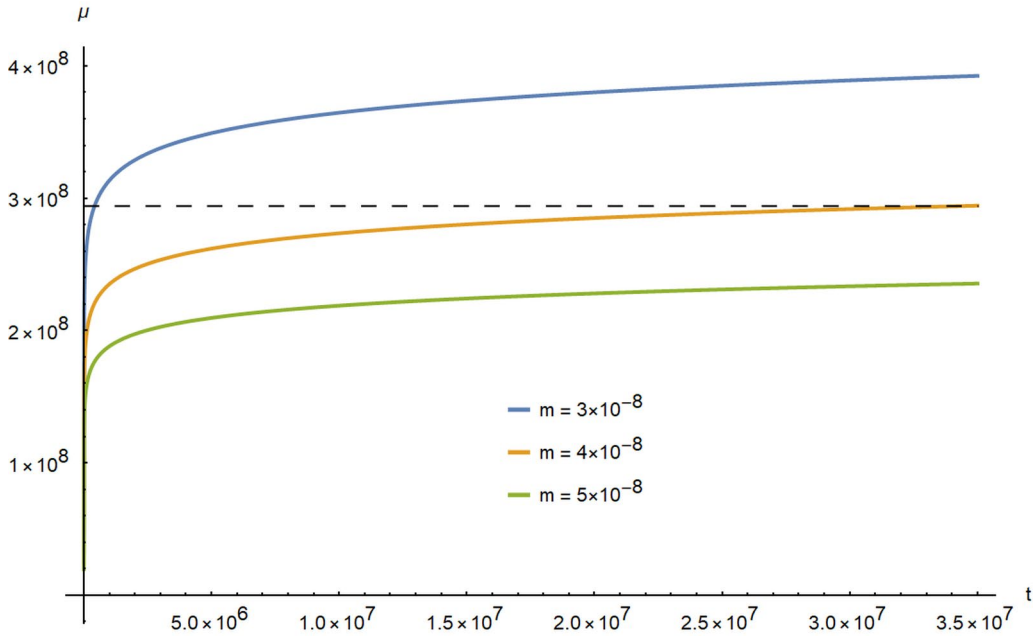

Figure 15: This plot shows that the mean μ can be approximated by a logarithmic law. For $t = 3.5 \times 10^7$, the curve in the middle yields an average of about 3×10^8 generations, which corresponds to the *Pan* / *Homo* split date. The plot also shows that the optimal mutation rate is far from those involved here ($a = 4$ and $d = 6$).

The shape of this curve depends on a number of parameters. According to Coale (1972 pp. 5-6), the major one of them are contraceptives, which are widely used in modern societies. This is why modern women get fewer children after the age of 30 compared to communities like prehistoric people in which contraception plays a minor role. This is why the distribution curve for industrialized societies falls down more steeply after that age. This shortens the generation time if the start and end of the reproduction time remain the same. Furthermore, in prehistoric populations the start is earlier than in modern societies. Hence, there is a lot of uncertainty inside this parameter. Nevertheless, the generation time is normally taken to be between 20 and 30 years (Fenner 2005).

What should also be mentioned is that we do not exactly need the generation time because our calculations are based on the length from the kid closest to the target DNA, "the best", to another kid of this kind of the next generation. This should yield a somewhat longer generation time than the normal length. Besides this, with a minimal generation time of 20 years, the chimpanzee–human last common ancestor should at least have lived $20 \times 2.94 \times 10^8$ years ≈ 6000 Ma ago, which is very far away from the longest estimate of 13 Ma. And this is only a lower limit far below the exact value natural selection would take. If one argues that our earliest

ancestors may have had a much shorter generation time than both chimpanzees and humans, it cannot be shorter than 10 years. With this value, one arrives at 3 billion years, which corresponds roughly to the time when life itself began to spread on earth.

The error on this result can be estimated with two curves that pass above and below the data points that have been found for the approximation of the mean (see codes.nb for details). This shows that the absolute error is not more than about 30 generations, which is vanishing low for a result of the order of 10^8 generations. As for the optimal mutation rate, $\mu_1(m_1) > \mu_2(m_2) > \mu_3(m_3)$ and $m_1 < m_2 < m_3$ from figure 15. This means that the optimal rate must be greater than $m_3 = 5 \times 10^{-8}$ because there is only one minimum and the mean decreases for increasing *m* (see sec. 3.4). Therefore, it is not possible that the optimal rate lies somewhere between the values 2×10^{-8} and 4×10^{-8} given in the literature (sec. 4.1). So there is no hope for Darwinists that such a hypothetical rate might miraculously dive the mean within the range of the empirical *Pan* / *Homo* split date.

The mean of a distribution does not say anything about the probability that an event may occur before it. For instance, the highest probability of the single kid distribution is at the first generation, as suggested by figure 4, even though the mean is on the right of it. This is why it would be interesting to be able to calculate the cumulative distribution function (CDF) of the weasel distribution for the *Pan* / *Homo* split date of 13 Ma (corresponding to 650000 generations) because this would yield the probability of the achievement of the genetic difference between chimps and humans in 650000 generations or less. So the CDF is taking into account that it can also happen before 650000 generations. This is necessary as there is a big uncertainty concerning this date. Mathematically, this means

$$\begin{aligned}\mathrm{CDF}(650000) &= P(\text{difference achieved in 1 generation or ... or in 650000 generations}) \\ &= P(v \le 650000) = \sum_{v=1}^{650000} P(v) > P(650000)\end{aligned}$$

even though counting from the first generation is not realistic. But it does not change a lot and is simpler. The CDF could be calculated directly with the formula[21]

$$\mathrm{CDF}(V) = \frac{1}{a^t} + \left| \left(I - F^{V-1} \right) \cdot A \right|$$

Unfortunately, Mathematica makes errors for small *m* and higher *t* with this formula, so it cannot be used for up to $t = 600$ in order to be able to extrapolate the CDF to $t = 3.5 \times 10^7$ (see codes.nb, also for the rest of this section). But an upper limit can be calculated in extrapolating $P(650000)$ to $t = 3.5 \times 10^7$ and multiplying this result with 650000. This can be done with $\log(1/P(650000))$, which yields a straight line for increasing *t*. The outcome is a number of something less than $10^{-19 \text{ million}}$.

This is a surrealistic small number difficult to compare with anything in the real world. For example, the observable universe has a radius of about 46.6 billion light years. This is more than the distance light can travel in about 14 billion years corresponding to the age of the Big Bang because the universe has always expanded since then. This yields a spherical volume of about 10^{80} cubic meters. The smallest physical distance that can be measured is the Planck length of the order of 10^{-35} meters, which is much smaller than atoms, protons and even quarks. Now, if one fills the observable universe with cubes of an edge length equal to the Planck length, this yields about 10^{186} cubes. If each of these cubes has a unique identity, then the probability to find a very special one of them by a fortuitous undirected search is the reciprocal of 10^{186}, in other words 10^{-186}, which has an about $10^{19 \text{ million}}$ times higher probability than a probability of something less than $10^{-19 \text{ million}}$ since something over 19 million minus 186 does not change significantly.

What could be objected to such an outcome? That a linear extrapolation up to $t = 3.5 \times 10^7$ based only upon $t = 100, ..., 500$ of $\log(1/P(650000))$ is like climbing on a mountain, looking around and infer from the reduced sight that the earth is flat? As mentioned in the introduction, this is indeed a justified objection. However, one does not have to circumvent the whole earth in order to realize that it has a nearly spherical form.

Indeed, when beaming a laser horizontally over a calm lake, it reaches a height of almost eight meters over the water surface only after ten kilometers. For this reason, outgoing ships seem to sink behind the horizon when observed from the shore. Analogously, one does not need to calculate $\log\left(1/P(650000)\right)$ directly in order to show that it also yields a straight line for t in the order of 10^7 or more, as we are going to see.

As a first approach, one could argue that $10^{-19\text{ million}}$ is grosso modo in the same realm as $P(1)=1/a^t \approx 10^{-21\text{ million}}$ and $P(2)\approx 10^{-19\text{ million}}$. As suggested by figure 5, the weasel distribution is in fact very low before its peak. This is confirmed by figure 16, which shows that it is wandering from the left to the right with increasing t, while its shape does not change a lot, letting on the left of the peak a kind of desert where nothing grows, in other words, where natural selection is not able to produce anything since the probability there is vanishing low. As we will see, this comes from the fact that the standard deviation (blue line) remains almost constant with increasing t.

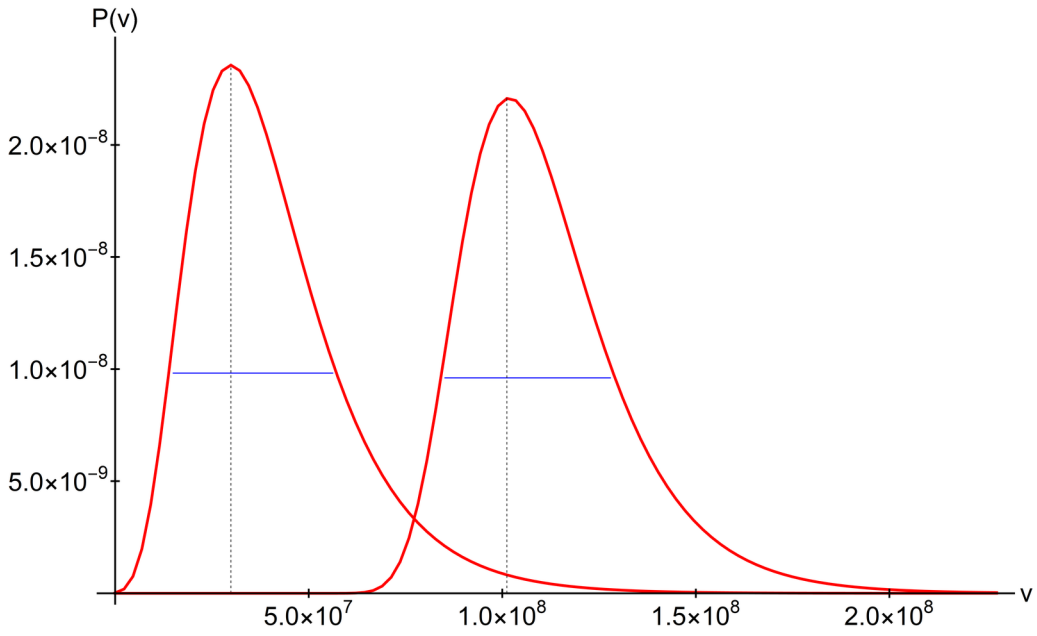

Figure 16: The weasel distribution wanders from the left to the right for increasing t (8 and 600 here), maintaining approximately its shape ($a=4$, $m=4\times10^{-8}$ and $d=6$).

We will now approximate the width of the peak for $t=3.5\times10^7$ and see what consequences it has for $P(650000)$. For this purpose, the variance and $P(\text{mode})$ need to be extrapolated, that is, the height of the peak at its highest point. They behave in a similar way, for both are rapidly converging to a horizontal asymptote with increasing t. Twice the standard deviation (the square root of the variance) yields the horizontal width of the peak. Since it converges to a constant, the peak wanders to the right (figure 16) with a logarithmic speed since the mean is near the mode and is a logarithmic function of t. Multiplication of twice the standard deviation with $P(\text{mode})$ rapidly converges to a constant near 1, which means that the area enclosed by the peak remains almost constant for t going to infinity and that the bulk of all probabilities lies in this region.

As a consequence, also the area outside the peak must remain nearly constant. So the area beneath $P(v)$ on the left of the peak must remain constant as well. At the same time, it is increasingly stretched horizontally because the peak wanders to the right. This means that its average height must decrease and thereby also all its probabilities. This does not prove that they are decreasing exponentially. But, except for the transition from $P(1)$ to $P(2)$, the weasel distribution yields a smooth and continuous curve. So the decreasing cannot change abruptly from one generation to the next. Furthermore, it is almost horizontal on the left of the peak. In other words, since $P(1)=1/a^t$ and $P(2)$ are decreasing exponentially with t, then also the other probabilities nearby must decrease in the same manner.

So the next question to answer is whether one can consider $P(650000)$ near $P(2)$. This is indeed the case because one can show that the hole on the left of the peak occupies a part of about 13/14 of the space between $v=1$ and $v=\text{mode}$ for $t=3.5\times10^7$, whereas the mean for the same value is about $v=3\times10^8$. So the length occupied by the hole is about 2.8×10^8 generations. The 650000 generations divided by this number yield something more than 0.002. Therefore, the 650000 generations are on the far left, in other words, near the first generations where the decrease of $P(v)$ is exponential, which is why $P(650000)$ must decrease exponentially as well. So $\log\left(1/P(650000)\right)$ must also yield nearly a straight line for up to $t=3.5\times10^7$. The data suggest indeed that they converge to an oblique asymptote, which confirms a probability of the order of $10^{-19\text{ million}}$ for $\text{CDF}(650000)$. In other words, the probability that the *Pan* / *Homo* split date comes close to 13 Ma by means of natural selection is not close to zero, it is de facto equal to zero.

Conclusion

As mentioned in the introduction, I favored natural selection as a null hypothesis in order to test it against empirical data. Thereby, let us first resume all the parameters and concepts that are favorable to random evolution:

- As admitted by Dawkins himself, his weasel program does not really simulate natural selection but what he calls "selective breeding" (p. 48), which does not occur in nature even though one may claim that mating partners feel mutually attracted to each other if they are similar on a genetic level. Furthermore, artificially selecting the "best" kid in every family as parent for the next generation is even more efficient than breeding because an allele in principle beneficial but recessive will not in every case produce a visible phenotype and thereby not automatically procure an advantage for surviving. However, recessive alleles are more numerous than dominant ones. Genetic drift and other processes can reduce allele frequency and thereby increase the fixation probability of a beneficial gene. But the weasel algorithm as well as code 26 sets this probability to one. Thereby, more sophisticated algorithms and concepts that include such processes, like the neutral theory of Kimura for instance, cannot deliver better results.
- The weasel algorithm does not implement population size, which makes it a lower limit for more complex algorithms that implement such properties since population size exponentially increases the expected number of generations, as shown by code 26.
- The weasel distribution starts with parents that have an unrealistic high initial genetic variation of 25% on average. This favors natural selection because the target is achieved faster if there is a high number of already correct nucleotides. In principle, there should be zero correct nucleotides in the beginning as we are considering only the genetic difference between chimps and humans. In other words, not the entire DNA sequence is submitted to random mutations, which would slow down the process considerably. In the algorithm taking into account population size (code 26) and in the single parents model (code 27), this plays a minor role as the target is only achieved after the variation is below 10% in the first and a more realistic 0.5% in the second case.
- Genes are only functional if they are complete. A partial achievement of a target sequence is not functional and thereby does not procure an advantage for survival. Partial code may even be disadvantageous as mutations are often detrimental. A minimal gene length of three base pairs, corresponding to the number of codons, should at least be taken. But because of reduced computer performance, gene length was not taken into consideration and every single nucleotide was allowed to be beneficial, which boosts the weasel algorithm unrealistically.
- We assumed the genetic differences to be a consecutive sequence of base pairs without having holes of code that are the same from one species to another. As can be seen from figure 9, differences are not consecutive yet. This only happens when completely new genes are generated. Thereby, ignoring this fact, as we did, leads to a considerable increase of the probability to achieve a target sequence.
- We used conservative parameters: The number of diverging nucleotides $t = 3.5 \times 10^7$ is based on the assumption that there is a lot of non-coding DNA in both the chimpanzee and human genomes. This is contested and other studies arrive at much higher differences. The mutation rate $m = 4 \times 10^{-8}$ is the highest mentioned and is far from an optimal rate. The fertility rate is rounded up from $d = 5.6$ to $d = 6$, as an integer is needed. A generation time of 20 years has been chosen, which is the lowest mentioned in the literature.

Because of all these concepts and parameters favoring natural selection, one should arrive at a time span far below the maximal empirical value of 13 Ma for the time when the human and chimpanzee lineages split from each other – if this null hypothesis was true. However, the contrary is indisputably the case, for the outnumbering does not only consist in some additional million years but 6 billion years. This is only what

produces the weasel distribution, which is much faster than natural selection. For the fastest realistic model, the single parents model, the weasel distribution yields a lower limit of about seven trillion years for the *Pan* / *Homo* split to happen through natural selection. The worst-case scenario from the perspective of Darwinists yields the cumulative distribution function, which calculates the probability that an event occurs *at least* at a given date, that is, 13 Ma in our case. The result is a probability as low as $10^{-19\ \text{million}}$ (ten to the power of minus nineteen million). In other words, the probability that the *Pan* / *Homo* split occurred by means of natural selection is not close to zero. It is de facto *equal* to zero. Thereby, while it is true that natural selection is capable of changing skin color and the like, which is often called microevolution, extrapolating this creativity to macroevolution hold responsible for the generation of new species is not based on science but, at best, on naïve wishful thinking; at worst, it is untruthful propagandistic ideology hidden as science.

One may argue that the human species is only one possible target among many others and natural selection might just have found a way to this special target, while it could also have produced other humanlike species. Therefore, if one follows the imagination of science-fiction movies, imaginary humanoids like Vulcans, Romulans and other Klingons could just as well have been a genetic target. It is difficult to say whether this would have increased the probability of their appearance. On one hand, a directed algorithm like the weasel program is like guiding someone out of a labyrinth, which evidently goes faster than if the exit has to be found by trying out. On the other hand, if there are several possible outcomes that must be found by trying out without guidance, they may be discovered faster than through a single directed exit. All depends on the number of possible exits and the complexity of the labyrinth.

However, what one observes is an evolutionary tree at the top of which are humans and not a circular spreading of the species, which would be observable if evolution were undirected. Furthermore, the genetic differences between such humanoids would be very low as they are supposed to have distinct human traits. Hence, the target to achieve would not have been very different, such that in any case the genetic deviation from the common ancestor of the *Pan* and *Homo* genera would have been of the same order of magnitude. Furthermore, if human evolution were undirected as claimed by Dawkins, one should find such variants among the human population, at least to some extent, if this concept was true. This is by far not the case, though. There are certainly different human races, but the genetic variation among them is so low that any racist theory claiming superiority of some races over others can easily be refuted, as shown for instance by Jorde and al. (2004).

Apart from this, the archeological and fossil record shows that human evolution is indeed directional. Modern humans have evolved from Australopiths step by step, each one generating species with higher abilities. This is clearly documented by the increasing sophistication of stone and other tools of the Oldowan (*Homo habilis*), the Acheulean (*Homo erectus*), the Mousterian (*Homo Neanderthalensis*) and finally the Aurignacian (*Homo sapiens*), even though there are also intermediate industries (Clark 1977). This is correlated with a progressive increase of brain size (Holloway 2015), despite the fact that the brain of Neanderthals was even slightly greater compared to modern humans. However, the Mousterian tools are clearly less sophisticated than those of the Aurignacian and also show less art objects and personal ornamentation, which most paleoanthropologists attribute to limited cognitive abilities and language of the Neanderthals, even though genetically they are very close to modern humans (Harvati 2015 p. 2260). Hence, a parallel evolution of different humanoids with same skills is not observable. Furthermore, humanoids would have branched off long before the *Pan* / *Homo* split if non-directional random evolution had been taking place.

One may argue that evolution is an optimization process towards a unique target that is optimally adapted to the special conditions prevailing only on this planet. However, it is easy to imagine species that are much better in surviving than humans. Why do we not have more than two eyes (especially on the back of the head to better detect danger), arms and legs like spiders or echo sounders like bats or other very useful survival capacities that abound in other species? If there were an important climatic or other catastrophe, as happened several times in the past, microorganisms, insects, rats and so on would survive much better than humans.

While it is true that humanity is capable of destroying every life on Earth through modern technology and thereby has an undeniable superiority over all other species, there has never been a selection pressure in the past that would have led to the intellectual potential to develop such technologies: the Aurignacian tools are very sophisticated, but why should the evolution that led to brain capacities necessary to produce and use them at the same time generate the potential of conceiving, for instance, the general relativity of Einstein or the 9th symphony of Beethoven? This is the thought of Francis Crick, the codiscoverer of DNA, who wrote: "Our highly developed brains, after all, were not evolved under the pressure of discovering scientific truths but only to enable us to be clever enough to survive and leave descendants" (1995 p. 262; Dembski 2008 p. 33). Survival driven evolution can indeed not at all explain the complex individual and social behavior of humans because they are far beyond simple survival.

After all these considerations, the conclusion is that natural selection as a species producing process must be considered an impossible event. If one is still willing to defend it against this evidence, one is left with no other choice than to resort to the anthropic principle, infinite number of parallel universes or other esoteric non-testable models. This, however, has nothing to do with objective scientific-driven research but rather makes me think of pathological gambling addiction or desperate clutching to straws because certain alternatives are categorically and axiomatically rejected even though they are blindingly obvious.

Natural selection as a species producing mechanism must be ranged on the same level of absurdity as, for instance, the widespread attempts to construct perpetual motion machines in the Middle Ages, the impossibility of which was only refuted in the early 19th century through the laws of thermodynamics. Spontaneous generation is another such typical black box (Behe p. 41), prevailing from antiquity until 1859 when it was refuted by Louis Pasteur. It was then immediately replaced by natural selection through Charles Darwin's publication of *On the Origin of Species* in the very same year. Since then, natural selection has become like a blind passenger embarked on a vessel of otherwise sound science like common descent, genetics, geological ages and so forth, which gives it a fallacious camouflage of legitimacy and credibility. This is why the time is since long overdue to unmask this stowaway and let him sail back to his country of origin – the land of fantasy.

Unfortunately, I am convinced that many scientists in the field of biology are well aware of the presence of this illegitimate passenger but protect him tacitly, consciously and willingly. Darwinists are eternal know-alls who always want to have the last word. They unhesitatingly treat as pseudo-scientists all those who criticize them in order to divert attention from the fact that macroevolution driven by natural selection is pseudoscience. I have the strong impression that Darwinists, if they were forced to make a choice, would prefer to rot in hell rather than admitting, in accordance with the overwhelming evidence, that life cannot be explained naturally, in other words, that it is a miracle.

Notes

[1] To mention just a few:

en.wikipedia.org/wiki/Weasel_program
rationalwiki.org/wiki/Dawkins_weasel
rosettacode.org/wiki/Evolutionary_algorithm

[2] Demonstration of equation 4:

$$s_n = 1 + q + q^2 + ... + q^n \Rightarrow$$
$$q\,s_n = q + q^2 + ... + q^n + q^{n+1} \Rightarrow s_n - q\,s_n = 1 - q^{n+1}$$

At the same time

$$s_n - q\,s_n = (1-q)\,s_n \Rightarrow s_n = \frac{1-q^{n+1}}{1-q} \Rightarrow$$
$$\sum_{k=0}^{\infty} q^k = \lim_{n\to\infty} \sum_{k=0}^{n} q^k = \lim_{n\to\infty} s_n = \lim_{n\to\infty} \frac{1-q^{n+1}}{1-q} = \frac{1}{1-q} \text{ if } 0 < q < 1$$

which is the case because q is a probability.

[3] The mean of a probability distribution is based on the arithmetic mean, which takes z numbers, sums them and divides by z (retaking the notation in the simulations). For a distribution $P(v)$, we calculate the mean of the positive integer random variable v, which occurs $w(v)$ times in the course of $z = \sum w(v)$ trials. Assuming that V is the maximal value v can take, its arithmetic mean is calculated as follows:

$$\frac{\underbrace{1+...+1}_{w(1)\text{ times}} + \underbrace{2+...+2}_{w(2)\text{times}} + ... + \underbrace{v+...+v}_{w(v)\text{ times}} + ... + \underbrace{V+...+V}_{w(V)\text{ times}}}{z} = \frac{\sum_{v=1}^{V} v\,w(v)}{z} = \sum_{v=1}^{V} v\frac{w(v)}{z}$$

But v can take any value, so $V \to \infty$. And since the relative frequency $w(v)/z \to P(v)$ for $z \to \infty$, we get equation 5.

[4] Demonstration of equation 6: deriving the left term of equation 4 yields

$$\frac{d}{dq}\sum_{v=0}^{\infty} q^v = \sum_{v=0}^{\infty} \frac{d}{dq} q^v = \sum_{v=0}^{\infty} v\,q^{v-1} = \sum_{v=1}^{\infty} v\,q^{v-1}$$

while deriving the right term gives

$$\frac{d}{dq}\left(\frac{1}{1-q}\right) = \frac{1}{(1-q)^2}$$

[5] Demonstration of equation 8:

$$\sum_{v=1}^{\infty} \left(v - \langle v \rangle\right)^2 P(v) = \sum_{v=1}^{\infty} \left(v^2 - 2v\langle v \rangle + \langle v \rangle^2\right) P(v)$$

$$= \underbrace{\sum_{v=1}^{\infty} v^2 P(v)}_{=\langle v^2 \rangle} - 2\langle v \rangle \underbrace{\sum_{v=1}^{\infty} v P(v)}_{=\langle v \rangle} + \langle v \rangle^2 \underbrace{\sum_{v=1}^{\infty} P(v)}_{=1} = \langle v^2 \rangle - 2\langle v \rangle^2 + \langle v \rangle^2 = \langle v^2 \rangle - \langle v \rangle^2$$

6 The demonstration of equation 9 is similar to that of equation 6: multiplying equation 4 with q and deriving the left term twice yields

$$\frac{d^2}{dq^2}\sum_{v=0}^{\infty} q^{v+1} = \sum_{v=0}^{\infty} \frac{d^2}{dq^2} q^{v+1} = \sum_{v=0}^{\infty} (v+1)\frac{d}{dq} q^v = \sum_{v=0}^{\infty} (v+1) v q^{v-1} = \sum_{v=1}^{\infty} v^2 q^{v-1} + \sum_{v=1}^{\infty} v q^{v-1} = \sum_{v=1}^{\infty} v^2 q^{v-1} + \frac{1}{(1-q)^2}$$

while deriving twice the right term of equation 4 gives

$$\frac{d^2}{dq^2}\left(\frac{q}{1-q}\right) = \frac{d}{dq}\left(\frac{1-q-q(-1)}{(1-q)^2}\right) = \frac{d}{dq}(1-q)^{-2} = -2(1-q)^{-3}(-1) = \frac{2}{(1-q)^3}$$

$$\Rightarrow \sum_{v=1}^{\infty} v^2 q^{v-1} = \frac{2}{(1-q)^3} - \frac{1}{(1-q)^2} = \frac{1+q}{(1-q)^3}$$

7 Equation 4 is again used in this calculation:

$$\sum_{v=1}^{\infty} P(v) = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{v=2}^{\infty} q^{v-2} = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{k=0}^{\infty} q^k = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \frac{1}{1-q} = \frac{1}{a} + \left(1-\frac{1}{a}\right) = 1$$

8 Demonstration of equation 19:

Let $k-1 = v-2 \Rightarrow k(v) = v-1 \Rightarrow k(2) = 1$ and $v(k) = k+1 \Rightarrow$

$$\sum_{v=1}^{\infty} v P(v) = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{v=2}^{\infty} v q^{v-2} = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{k=1}^{\infty} (k+1) q^{k-1} = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \left(\sum_{k=1}^{\infty} k q^{k-1} + \sum_{k=1}^{\infty} q^{k-1}\right)$$

$$= \frac{1}{a} + \left(1-\frac{1}{a}\right)(1-q)\left(\frac{1}{(1-q)^2} + \frac{1}{1-q}\right) = \frac{1}{a} + \left(1-\frac{1}{a}\right)\left(\frac{1}{1-q} + 1\right) = \frac{1}{a} + \frac{1}{1-q} + 1 - \frac{1}{a(1-q)} - \frac{1}{a}$$

$$= \frac{a + a(1-q) - 1}{a(1-q)} = \frac{a(2-q)-1}{a(1-q)} = \frac{a\left(1+\frac{m}{a}\right)-1}{a\frac{m}{a}} = \frac{a+m-1}{m}$$

where equations 4, 6 and 17 are used.

9 Demonstration of equation 20:

$$\langle v^2 \rangle = \sum_{v=1}^{\infty} v^2 P(v) = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{v=2}^{\infty} v^2 q^{v-2} = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \sum_{k=1}^{\infty} (k+1)^2 q^{k-1}$$

$$= \frac{1}{a} + \left(1-\frac{1}{a}\right) p \left(\sum_{k=1}^{\infty} k^2 q^{k-1} + 2\sum_{k=1}^{\infty} k q^{k-1} + \sum_{k=1}^{\infty} q^{k-1}\right) = \frac{1}{a} + \left(1-\frac{1}{a}\right) p \left(\frac{1+q}{(1-q)^3} + 2\frac{1}{(1-q)^2} + \frac{1}{1-q}\right)$$

$$= \frac{1}{a} + \left(1-\frac{1}{a}\right) p \frac{4-3q+q^2}{(1-q)^3} = \frac{1}{a} + \left(1-\frac{1}{a}\right)\frac{4-3q+q^2}{(1-q)^2} = \frac{2a^2 + a(m-2) + m(m-1)}{m^2}$$

where equations 4, 6 and 9 are used. So with equations 8 and 19 the variance is

$$\sigma^2 = \frac{2a^2 + a(m-2) + m(m-1)}{m^2} - \left(\frac{a+m-1}{m}\right)^2 = \frac{(a-1)(a-m+1)}{m^2}$$

[10] The calculation is almost the same as in note 7:

$$\sum_{v=1}^{\infty} P(v) = \frac{1}{a} + \left(1-\frac{1}{a}\right)\left(1-q^d\right)\sum_{v=2}^{\infty} q^{d(v-2)} = \frac{1}{a} + \left(1-\frac{1}{a}\right)\left(1-q^d\right)\sum_{v=2}^{\infty}\left(q^d\right)^{v-2}$$
$$= \frac{1}{a} + \left(1-\frac{1}{a}\right)\left(1-q^d\right)\sum_{k=0}^{\infty}\left(q^d\right)^k = \frac{1}{a} + \left(1-\frac{1}{a}\right)\left(1-q^d\right)\frac{1}{1-q^d} = 1$$

[11] Same calculation as in note 8. By comparison of distribution 21 with 18, one only has to replace q with q^d and $p = 1-q$ with $1-q^d$:

$$\sum_{v=1}^{\infty} v\,P(v) = \ldots = \frac{a\left(2-q^d\right)-1}{a\left(1-q^d\right)} = \frac{a\left(q^d-2\right)+1}{a\left(q^d-1\right)} = \frac{a\left(\left(1-\frac{m}{a}\right)^d - 2\right)+1}{a\left(\left(1-\frac{m}{a}\right)^d - 1\right)}$$

[12] Same calculation as in notes 8 and 9 by replacement of q with q^d, but *Mathematica* is used because of its length (see codes.nb):

$$\sigma^2 = \left\langle v^2\right\rangle - \left\langle v\right\rangle^2 = \ldots = \frac{1}{a} + \left(1-\frac{1}{a}\right)\frac{4-3q^d+q^{2d}}{\left(1-q^d\right)^2} - \left(\frac{a\left(q^d-2\right)+1}{a\left(q^d-1\right)}\right)^2$$
$$= \ldots = \frac{(a-1)\left(1+a\,q^d\right)}{a^2\left(q^d-1\right)^2} = \frac{(a-1)\left(1+a\left(1-\frac{m}{a}\right)^d\right)}{a^2\left(\left(1-\frac{m}{a}\right)^d - 1\right)^2}$$

where equation 8 is used.

[13] Demonstration of equation 32:

$$\left|(G+B)\cdot X\right| = \left|\begin{pmatrix} g_1+b_{11} & b_{12} & \cdots & b_{1,2^t-1} \\ b_{21} & \ddots & & \vdots \\ \vdots & & \ddots & b_{2^t-2,2^t-1} \\ b_{2^t-1,1} & \cdots & b_{2^t-1,2^t-2} & g_{2^t-1}+b_{2^t-1,2^t-1} \end{pmatrix}\cdot\begin{pmatrix} x_1 \\ \vdots \\ x_{2^t-1} \end{pmatrix}\right| = \left\|\begin{pmatrix} g_1 x_1 + \sum_{j=1}^{2^t-1} b_{1j}\,x_j \\ \vdots \\ g_{2^t-1}\,x_{2^t-1} + \sum_{j=1}^{2^t-1} b_{2^t-1,j}\,x_j \end{pmatrix}\right|$$

$$= \sum_{i=1}^{2^t-1}\left(g_i\,x_i + \sum_{j=1}^{2^t-1} b_{ij}\,x_j\right) = \sum_{i=1}^{2^t-1} g_i\,x_i + \sum_{i=1}^{2^t-1}\sum_{j=1}^{2^t-1} b_{ij}\,x_j = \sum_{j=1}^{2^t-1} g_j\,x_j + \sum_{j=1}^{2^t-1}\sum_{i=1}^{2^t-1} b_{ij}\,x_j = \sum_{j=1}^{2^t-1}\left(\underbrace{g_j + \sum_{i=1}^{2^t-1} b_{ij}}_{=1}\right)x_j = \sum_{j=1}^{2^t-1} x_j = \left|X\right|$$

[14] Demonstration of equation 42: from equation 32 one has

$$\left|(G+B)\cdot X\right| = \left|G\cdot X + B\cdot X\right| = \left|G\cdot X\right| + \left|B\cdot X\right| = \left|X\right| \Rightarrow$$
$$\left|G\cdot X\right| = \left|X\right| - \left|B\cdot X\right| = \left|(I-B)\cdot X\right|$$

Since this is valid for an arbitrary vector *X*, it is also valid for $(I-B)^{-1}\cdot X$, from which follows

$$\left|G\cdot(I-B)^{-1}\cdot X\right| = \left|(I-B)\cdot(I-B)^{-1}\cdot X\right| = \left|X\right|$$

Notice that it is not possible to conclude from this that $G\cdot(I-B)^{-1} = I$.

[15] For this purpose

$$\sum_{v=1}^{n} v\,q^{v-1} = \frac{1-(n+1)q^n + n\,q^{n+1}}{(1-q)^2}$$

is needed, which is obtained by deriving both sides of equation 4 as in the demonstration of equation 6 but applied to the finite sum. Now, presupposing that this equation is true for a matrix *B* instead of *q*, one gets

$$\sum_{v=1}^{n} v\,B^{v-1} = (I-B)^{-2}\cdot\left(I-(n+1)B^n + n\,B^{n+1}\right)$$

where $(I-B)^{-2}$ is the inverse of $I-B$ raised to the power of 2., Setting then $n=1$, we check whether the equation is true for this base case. On one hand, one gets

$$\sum_{v=1}^{1} v\,B^{v-1} = B^0 = I$$

On the other hand

$$(I-B)^{-2}\cdot\left(I-(1+1)B^1 + 1\cdot B^{1+1}\right) = (I-B)^{-2}\cdot\left(I-2B+B^2\right) = (I-B)^{-2}\cdot(I-B)^2 = I$$

Consequently, the base case is valid. The inductive step $n \to n+1$ must yield

$$\sum_{v=1}^{n+1} v\,B^{v-1} = (I-B)^{-2}\cdot\left(I-\left((n+1)+1\right)B^{n+1} + (n+1)B^{n+2}\right)$$

in which case the equation is definitely proven. In fact,

$$\sum_{v=1}^{n+1} v\,B^{v-1} = \sum_{v=1}^{n} v\,B^{v-1} + (n+1)B^n = (I-B)^{-2}\cdot\left(I-(n+1)B^n + n\,B^{n+1}\right) + (n+1)B^n$$
$$= (I-B)^{-2}\cdot\left(I-(n+1)B^n + n\,B^{n+1}\right) + (I-B)^{-2}\cdot(I-B)^2\cdot(n+1)B^n$$
$$= (I-B)^{-2}\cdot\left[\left(I-(n+1)B^n + n\,B^{n+1}\right) + (I-B)^2\cdot(n+1)B^n\right]$$
$$\vdots$$
$$= (I-B)^{-2}\cdot\left(I-\left((n+1)+1\right)B^{n+1} + (n+1)B^{n+2}\right)$$

As $\lim_{n\to\infty} B^n = 0$ for $n \to \infty$, where 0 means the matrix only composed of zeros, this finally yields

$$\sum_{v=1}^{\infty} v\, B^{v-1} = \left(I - B\right)^{-2}$$

[16] The demonstration is done in a similar way than for equation 19:

$$\begin{aligned}
\langle v \rangle &= \sum_{v=1}^{\infty} v\, P(v) = \frac{1}{a^t} + \sum_{v=2}^{\infty} v \left| G \cdot B^{v-2} \cdot A \right| = \frac{1}{a^t} + \left| \sum_{v=2}^{\infty} v\, G \cdot B^{v-2} \cdot A \right| \\
&= \frac{1}{a^t} + \left| G \cdot \left(\sum_{v=2}^{\infty} v\, B^{v-2} \right) \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(\sum_{k=1}^{\infty} (k+1) B^{k-1} \right) \cdot A \right| \\
&= \frac{1}{a^t} + \left| G \cdot \left(\sum_{k=1}^{\infty} k\, B^{k-1} + \sum_{k=1}^{\infty} B^{k-1} \right) \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(\left(I - B\right)^{-2} + \left(I - B\right)^{-1} \right) \cdot A \right| \\
&= \frac{1}{a^t} + \left| G \cdot \left(I - B\right)^{-2} \cdot A \right| + \left| G \cdot \left(I - B\right)^{-1} \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(I - B\right)^{-2} \cdot A \right| + 1 - \frac{1}{a^t} = \left| G \cdot \left(I - B\right)^{-2} \cdot A \right| + 1
\end{aligned}$$

where equations 41, 43 and 44 are used. With equation 42 we finally get equation 45.

[17] The demonstration is almost the same as for equation 20:

$$\begin{aligned}
\left\langle v^2 \right\rangle &= \sum_{v=1}^{\infty} v^2 P(v) = \frac{1}{a^t} + \sum_{v=2}^{\infty} v^2 \left| G \cdot B^{v-2} \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(\sum_{v=2}^{\infty} v^2 B^{v-2} \right) \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(\sum_{k=1}^{\infty} (k+1)^2 B^{k-1} \right) \cdot A \right| \\
&= \frac{1}{a^t} + \left| G \cdot \left(\sum_{k=1}^{\infty} k^2 B^{k-1} + 2 \sum_{k=1}^{\infty} k\, B^{k-1} + \sum_{k=1}^{\infty} B^{k-1} \right) \cdot A \right| = \frac{1}{a^t} + \left| G \cdot \left(\left(I - B\right)^{-3} \cdot \left(I + B\right) + 2\left(I - B\right)^{-2} + \left(I - B\right)^{-1} \right) \cdot A \right| \\
&= \frac{1}{a^t} + \left| G \cdot \left(I - B\right)^{-3} \cdot \left(I + B\right) \cdot A \right| + 2 \left| G \cdot \left(I - B\right)^{-2} \cdot A \right| + \left| G \cdot \left(I - B\right)^{-1} \cdot A \right| \\
&= \frac{1}{a^t} + \left| \left(I - B\right)^{-2} \cdot \left(I + B\right) \cdot A \right| + 2 \left| \left(I - B\right)^{-1} \cdot A \right| + 1 - \frac{1}{a^t} = \left| \left(I - B\right)^{-2} \cdot \left(I + B\right) \cdot A \right| + 2 \left| \left(I - B\right)^{-1} \cdot A \right| + 1
\end{aligned}$$

where equations 29, 41, 42, 44 and 46 are used. Thereby, with equations 8 and 45 one gets

$$\begin{aligned}
\sigma^2 &= \left| \left(I - B\right)^{-2} \cdot \left(I + B\right) \cdot A \right| + 2 \left| \left(I - B\right)^{-1} \cdot A \right| + 1 - \left(\left| \left(I - B\right)^{-1} \cdot A \right| + 1 \right)^2 \\
&= \left| \left(I - B\right)^{-2} \cdot \left(I + B\right) \cdot A \right| + 2 \left| \left(I - B\right)^{-1} \cdot A \right| + 1 - \left| \left(I - B\right)^{-1} \cdot A \right|^2 - 2 \left| \left(I - B\right)^{-1} \cdot A \right| - 1 \\
&= \left| \left(I - B\right)^{-2} \cdot \left(I + B\right) \cdot A \right| - \left| \left(I - B\right)^{-1} \cdot A \right|^2
\end{aligned}$$

[18] The binomial distribution yields the probability that *k* out of *t* nucleotides are correct, which yields $P(k) = C_k^t\, p^k\, q^{t-k}$ as explained. To calculate the mean of this distribution, we need

$$k\, C_k^t = k \binom{t}{k} = k \frac{t!}{k!(t-k)!} = \frac{t!}{(k-1)!(t-k)!} = \frac{t(t-1)!}{(k-1)!(t-k)!} = \frac{t(t-1)!}{(k-1)!\left((t-1)-(k-1)\right)!} = t\, C_{k-1}^{t-1}$$

as well as $k = h + 1 \Rightarrow h(k) = k - 1 \Rightarrow h(1) = 0$ and $h(n) = n - 1 \Rightarrow$

$$\langle k\rangle = \sum_{k=0}^{t} k\, C_k^t p^k q^{t-k} = \sum_{k=1}^{t} k\, C_k^t p^k q^{t-k} = \sum_{k=1}^{t} t\, C_{k-1}^{t-1} p^k q^{t-k} = t\sum_{k=1}^{t} C_{k-1}^{t-1} p^k q^{t-k}$$
$$= t\sum_{h=0}^{t-1} C_h^{t-1} p^{h+1} q^{t-(h+1)} = p\,t\sum_{h=0}^{t-1} C_h^{t-1} p^h q^{(t-1)-h} = p\,t\,(p+q)^{t-1} = p\,t$$

where the binomial theorem (eq. 11) is used.

[19] Let $\{x, y\}$ be the coordinates in the normal plot and $\{u, v\}$ the coordinates in the log-log plot (LLP). The "on paper" equal distances in the LLP are at $\ldots, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2} \ldots$, on both the abscissa and ordinate, so $x = 10^u$ and $y = 10^v$. From this follows $u = \log_{10} x$ and $v = \log_{10} y$. Now, if there is a straight line visible "on the paper" of a LLP, then

$$v(u) = s\,u + c = s\log_{10} x + \log_{10} r = \log_{10}\left(r\,x^s\right) = \log_{10} y \;\Rightarrow\; y(x) = r\,x^s$$

where we defined the intercept $c = \log_{10} r$. The origin of the axis in a normal plot is at $\{0, 0\}$ and the intercept is defined as the point where a straight line intersects the ordinate. In a LLP on the other hand, the axis origin is normally not at $\{0, 0\}$ because *u* and *v* can be negative. Mathematica chooses an origin automatically in order to fit the plotted data. Nevertheless, the intercept $c = \log_{10} r$ cuts the vertical line at $u = 0$ because in this case $v(0) = c$. Furthermore, one then has $x = 1$ and $y(1) = r$. So the intercept in the LLP must be determined at the intersection point with the vertical line going through $x = 1$.

[20] On page 274, Bentley et al. give a standard error (S.E.) of 0.4 in the augmented sample of foragers ($n = 12$). The standard error is defined as $\text{S.E.} = \sigma/\sqrt{n}$ where σ is the standard deviation. Therefore, $\sigma = \text{S.E.}\sqrt{n} = 0.4\sqrt{12} = 1.386$.

[21] This is almost the same calculation as for equation 43:

$$\text{CDF}(V) = P(v \le V) = \sum_{v=1}^{V} P(v) = \frac{1}{a^t} + \sum_{v=2}^{V} \left| H \cdot F^{v-2} \cdot A \right| = \frac{1}{a^t} + \left| \sum_{v=2}^{V} H \cdot F^{v-2} \cdot A \right| = \frac{1}{a^t} + \left| H \cdot \left(\sum_{k=0}^{V-2} F^k \right) \cdot A \right|$$
$$= \frac{1}{a^t} + \left| H \cdot (I - F)^{-1} \cdot \left(I - F^{V-1}\right) \cdot A \right| = \frac{1}{a^t} + \left| \left(I - F^{V-1}\right) \cdot A \right|$$

References

Armelagos, George J., Alan H. Goodman and Kenneth H. Jacobs. "The origins of agriculture: population growth during a period of declining health." *Population and Environment* 13.1 (1991): 9-22.

Arnason, Ulfur, Anette Gullberg and Axel Janke. "Molecular timing of primate divergences as estimated by two nonprimate calibration points." *Journal of Molecular Evolution* 47.6 (1998): 718-727.

Behe, Michael. *Darwin's Black Box: The Biochemical Challenge to Evolution*. New York: Free Press, 2006.

Bentley, Gillian R., Tony Goldberg and Grazyna Jasienska. "The Fertility of Agricultural and Non-Agricultural Traditional Societies." *Population Studies* 47.2 (1993): 269-281.

Birdsell, Joseph B. "Some Predictions for the Pleistocene Based On Equilibrium Systems Among Recent Hunter-Gatherers." *Man the Hunter*. Chicago: Aldine, 1968. 229-240.

Callaway, Ewen. "DNA clock proves tough to set." *Nature* 519.7542 (2015): 139-140.

Clark, Grahame. *World Prehistory: In New Perspective*. Cambridge University Press, 1977.

Coale, Ansley J. *The Growth and Structure of Human Populations: A Mathematical Investigation*. University of Princeton Press, 1972.

—. "The history of the human population." *Scientific American* 231.3 (1974): 40-51.

Cohen, Jon. "Relative differences: the myth of 1%." *Science* 316.5833 (2007): 1836.

Crick, Francis. *The Astonishing Hypothesis: The Scientific Search for the Soul*. New York: Touchstone, 1995.

Dawkins, Richard. *The Blind Watchmaker*. London: Penguin Books, 2006.

Dembski, William A. and Robert J. Marks. "Conservation of Information in Search: Measuring the Cost of Success." *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39.5 (2009): 1051-1061.

Dembski, William A. and Sean. McDowell. *Understanding intelligent design*. Harvest House Publishers, 2008.

Fenner, Jack N. "Cross-Cultural Estimation of the Human Generation Interval." *American journal of physical anthropology* 128.2 (2005): 415-423.

Hartwell, Leland H. *Genetics: From Genes to Genomes*. New York: McGraw-Hill Education, 2015.

Harvati, Katerina. "Neanderthals and their contemporaries." *Handbook of Paleoanthropology*. Berlin Heidelberg: Springer, 2015. 2243-2279.

Hassan, Fekri A. "Demographic archaeology." *Advances in archaeological method and theory* 1 (1978): 49-103.

Hill, Kim A., Magdalena Hurtado and Robert S. Walker. "High adult mortality among Hiwi hunter-gatherers: Implications for human evolution." *Journal of Human Evolution* 52.4 (2007): 443-454.

Holloway, Ralph L. "The Evolution of the Hominid Brain." *Handbook of Paleoanthropology*. Berlin Heidelberg: Springer, 2015. 1961-1987.

Jorde, Lynn B. and Stephen P. Wooding. "Genetic variation, classification and 'race'." *Nature genetics* 36.11 (2004): S28-S33.

Kaplan, Hillard, et al. "A theory of human life history evolution: diet, intelligence, and longevity." *Evolutionary Anthropology Issues News and Reviews* 9.4 (2000): 156-185.

Kimura, Motoo. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1984.

Klein, Richard G. *The Human Career: Human Biological and Cultural Origins*. University of Chicago Press, 2009.

Langergraber, Kevin E., et al. "Generation times in wild chimpanzees and gorillas suggest earlier divergence times in great ape and human evolution." *Proceedings of the National Academy of Sciences* 109.39 (2012): 15716-15721.

Levy, Samuel, et al. "The diploid genome sequence of an individual human." *PLoS biology* 5.10 (2007): 2113-2144.

Makalowski, Wojciech. *What is junk DNA, and what is it worth?* 12 February 2007. Scientific American. <www.scientificamerican.com/article/what-is-junk-dna-and-what>.

Mayr, Ernst. *Systematics and the origin of species, from the viewpoint of a zoologist*. Harvard University Press, 1999.

Mikkelsen, Tarjei S., et al. "Initial sequence of the chimpanzee genome and comparison with the human genome." *Nature* 437.7055 (2005): 69-87.

Monge, Janet and Alan Mann. "The paleodemography of extinct hominin populations." *Handbook of Paleoanthropology*. Berlin Heidelberg: Springer, 2015. 643-670.

Moran, Laurence A. *Human mutation rates - what's the right number?* 24 April 2015. <sandwalk.blogspot.ch/2015/04/human-mutation-rates-whats-right-number.html>.

Nachman, Michael W. and Susan L. Crowell. "Estimate of the mutation rate per nucleotide in humans." *Genetics* 156.1 (2000): 297-304.

Preuss, Todd M. "Human brain evolution: from gene discovery to phenotype discovery." *Proceedings of the National Academy of Sciences* 109.Supplement 1 (2012): 10709-10716.

Scally, Aylwyn and Richard Durbin. "Revising the human mutation rate: implications for understanding human evolution." *Nature Reviews Genetics* 13.10 (2012): 745-753.

Schultz, Emily A and Lavenda Robert H. *Cultural Anthropology: A Perspective on the Human Condition*. St. Paul: West Publishing Company, 1990.

Tomkins, J. "Comprehensive analysis of chimpanzee and human chromosomes reveals average DNA similarity of 70%." *Answers Research Journal* 6 (2013): 63-69.

Turchin, Peter. *Complex Population Dynamics: A Theoretical/Empirical Synthesis*. Princeton University Press, 2003.

White, Tim D., et al. "Ardipithecus ramidus and the paleobiology of early hominids." *Science* 326.5949 (2009): 64-86.

Wilkinson, Richard D., et al. "Dating primate divergences through an integrated analysis of palaeontological and molecular data." *Systematic Biology* 60.1 (2010): 16–31.

Wobst, Martin H. "Boundary conditions for Paleolithic social systems: a simulation approach." *American Antiquity* 39.2 (1974): 147-178.