Research Article

The Dependency Graph of Life

Winston Ewert*
Biologic Institute, Redmond, Washington, USA

Abstract

The hierarchical classification of life has been claimed as compelling evidence for universal common ancestry. However, research has uncovered much data which is not congruent with the hierarchical pattern. Nevertheless, biological data resembles a nested hierarchy sufficiently well to require an explanation. While many defenders of intelligent design dispute common descent, no alternative account of the approximate nested hierarchy pattern has been widely adopted. We present the dependency graph hypothesis as an alternative explanation, based on the technique used by software developers to reuse code among different software projects. This hypothesis postulates that different biological species share modules related by a dependency graph. We evaluate several predictions made by this model about both biological and synthetic data, finding them to be fulfilled.

Cite as: Ewert W (2018) The dependency graph of life. BIO-Complexity 2018 (3):1-27. **doi:**10.5048/BIO-C.2018.3.

Editor: Jonathan Bartlett

Received: January 31, 2018; **Accepted:** May 6, 2018; **Published:** July 17, 2018

Notes: A *Critique* of this paper, when available, will be assigned **doi:**10.5048/BIO-C.2018.3.c.

*Email: evoinfo@winstonewert.com

INTRODUCTION

Darwin cited the hierarchical classification of life as evidence for his theory [1], and the classification has continued to be cited as a central prediction of evolutionary theory [2]. However, modern research, especially in the area of molecular data, has complicated this picture. Prokaryotes do not fit a hierarchical scheme, leading Doolittle [3, p. 2226] to state: "Indeed, for prokaryotes, molecular data have falsified the [tree of life] hypothesis." Even amongst more complex lifeforms, data exist which are not congruent with the hierarchical pattern [4–8]. Some push for embracing a view of evolution no longer constrained by the tree of life [9]. Those who do not consider the tree of life falsified nevertheless hold to a modified version of it. Mechanisms have been added to explain deviations from the hierarchy such as horizontal gene transfer, incomplete lineage sorting, differential gene loss, gene resurrection, gene flow, and convergent evolution. Darwin referred to a "single progenitor" as the common ancestor of a clade, but, according to modern evolutionary theory, "Rather, the last universal common ancestor may have comprised a population of organisms with different genotypes that lived in different places at different times. [10, p. 220]."

However, the nested hierarchy pattern has not been abandoned. Some push back on the widespread inference of horizontal gene transfer [11, 12]. While Doolittle argues that the tree of life is a falsified hypothesis, he still argues that macroscopic life's fit to a hierarchy confirms evolutionary theory, [3, p. 2224]. A number of papers have attempted to quantitatively support common descent by showing a hierarchical signal in biological data [10, 13–15]. Defenders of evolutionary theory still commonly invoke the nested hierarchy as a successful prediction of and compelling evidence for common descent.

It may seem contradictory to both call for evolutionary theory to move beyond the tree of life and continue using the nested hierarchy as evidence for common descent. But the question is not whether life forms a nested hierarchy pattern, but rather to what degree life resembles a nested hierarchy. Life exhibits an *approximate* nested hierarchy pattern rather than forming an exact nested hierarchy. Even if the resemblance is weak or the approximation very loose, it is still undeniably present and thus must be explained. Defenders of common descent argue that even if many other mechanisms complicate the picture, the resemblance to a nested hierarchy is still best explained by common descent.

Intelligent design advocates critique evolutionary theory's account on a number of grounds [16–19]. Regardless of the merit of these critiques of common descent, advocates of design are left with a question: how do they explain the approximate hierarchical pattern? While some hold to common descent [20, 21], the more common position is separate ancestry [17, 22, 23]. They explain similarities among species by appealing to functional constraints or common design instead of common descent [17, 22]. Many species face similar functional constraints and thus must solve some problems in similar ways. Common design leads to similarity because designers frequently reuse components, modules, and solutions leading to similarity. As Nelson [17] writes: "An intelligent cause may reuse or redeploy the same module in different systems, without there necessarily being any material or physical connection between those systems. Even more simply, intelligent causes can generate identical patterns independently." Intelligent design proponents who hold to separate ancestry have no problem explaining similarities in biological organisms.

Nevertheless, while similarity per se can be explained by functional requirements and common design, this does not explain the approximate hierarchy. Defenders of common design argue that the products of design tend to fit hierarchical patterns because, "Common design predicts re-usage of parts in a non-random manner that fulfills design constraints required by the system. [24]" However, no hypothesis has been put forth to explain why this "non-random" re-usage of parts would exhibit the appearance of a nested hierarchy pattern.

Even if the defenders of common design had an explanation for the approximate hierarchical pattern, this would not be sufficient. Such an explanation would almost certainly be less parsimonious than common descent in accounting for the hierarchy and just as unparsimonious as common descent in accounting for the exceptions to the hierarchy. The only way a new explanation can claim to be a better explanation than common descent is if life exhibits another pattern, and has only been interpreted as an approximate hierarchy because of the similarity of the two patterns. An explanation of the true pattern could be more parsimonious than common descent overall because it would avoid the need for ad-hoc assumptions to explain deviations from the hierarchical pattern. Some have tried to propose such extended patterns [23, 25], but in practice the predicted pattern is just one of a nested hierarchy with deviations from that hierarchy. Deviations from a pattern do not constitute a pattern; there must be a pattern to the deviations. Since no explanation has been put forward that demonstrates such an extended pattern, no explanation so far can claim to be a better explanation for the approximate hierarchy than common descent.

We propose a new hypothesis to explain the approximate nested hierarchy pattern: *the dependency graph*. This hypothesis is drawn from the techniques used to reuse code in software development. Briefly, instead of a species descending from a single ancestral species (at least in the conventional account), each depends on multiple modules which themselves may depend on other modules. This brings together many of the ideas proposed by design-oriented critics of common descent. It uses the idea of common design by having modules reused in different species. It also draws on functional constraints, by having the valid combinations of modules restricted by which modules depend on other modules. Furthermore, it predicts an extended pattern which is the result of the proposed dependency graph mechanism.

The primary purpose of the current paper is to introduce this new hypothesis, with a goal of introducing a foundation for future research. We do test some predictions of the hypothesis. However, given that this is the first evaluation of the hypothesis, it is limited and subject to various assumptions. Future research will be required to address these limitations. Furthermore, many unanswered questions will be raised by the ideas presented in this paper, and fleshing out those answers is also left to future research.

Typically, advocates of separate descent nevertheless allow for some degree of common descent. For example, many advocates of separate ancestry would still hold that the members of *Canis* or *Felis* descend from a common ancestor. For the purposes of this paper, we assume that all species that we investigate were independently designed. The databases evaluated contain at most a few hundred metazoan species, and as such likely contain few examples of species that would descend from a common ancestor under limited common ancestry. Any consideration of the effect of limited common ancestry is left to future research.

When considering common descent, we assume a conventional understanding of common descent closer to Darwin's "single progenitor" than to Theobald's "population of organisms with different genotypes that lived in different places at different times." This paper restricts its consideration to the metazoans where such a conventional understanding is still typically thought to be approximately correct. In particular, we assume that all members of a metazoan taxonomic category trace their ancestry to a particular population living at a particular time with relatively homogeneous genomes. Every new species arises by the splitting of an existing species into multiple species, leaving every species with a single ancestral species. We assume that horizontal transfer of genetic information is sufficiently rare that it can be neglected in understanding the overall pattern of metazoan life.

DEPENDENCY GRAPHS

It is common for one thing to require or depend on another. It is impossible to learn calculus unless you first know algebra: calculus depends on algebra. A modern kitchen cannot be added to a building that lacks plumbing: kitchens depend on plumbing. A society cannot invent the internet without first inventing computers: the internet depends on computers. These dependency relationships can be found in many spheres.

While many spheres of life implicitly involve dependency relationships, in software development they are made explicit. Each collection of code, termed a module, will depend on other modules in order to perform its task. For example, if a software developer needs their program to download a file from the internet, they will not write the large amount of code necessary to perform this task. Instead, they will add a dependency on a module which already has the necessary code and reuse that code.

The structure that results from considering all the modules and the dependencies between them is called a *dependency graph.* The dependency graph of two JavaScript software modules, `jsdom` and `node-gyp`, is depicted in Figure 1. These modules perform two very different tasks, `jsdom` simulates part of a web browser and `node-gyp` compiles, or builds, software. Nevertheless, both modules depend on the `request` module, which downloads files from the internet, and consequently share both the `request` module and many modules depended on by the `request` module.

Common descent postulates that life is related by a tree, such as the one depicted for a selection of mammalian species in Figure 2. In contrast, the dependency graph hypothesis postulates that life is based on a dependency graph as depicted in Figure 3. Every species is a top-level module; nothing else depends on a species. A species depends on a variety of other modules, each providing some of the genes necessary for the final genome of the species. A module may contribute a single gene, or a large collection of genes. The arrows are reversed between the tree and the graph because in the tree an ancestral species splits into child species, but in the graph the species depend on modules, reversing the direction of the relationship.

Crucially, every module may depend on other modules. For example, the Carnivora module depends on the Laurasiatheria module. As a consequence, the genes contained in the Laurasiatheria module are also inherited by all species that depend on the Carnivora module. So all Carnivora species are Laurasiatheria species by virtue of that dependency relation. Thus we provide an alternative explanation for the nesting pattern observed in biology.

Both theories have important similarities. Taxonomic categories appear in both: in common descent they are represented by their most recent common ancestral species. In the dependency graph, they are modules. Both can explain the nested relationship of taxonomic categories. In common descent, this is because one species descended from another. In the dependency graph, this is because one module depends on another.

They differ on one crucial point: a species conventionally has one ancestral species, but a module typically has multiple dependencies. For example, see “echolocation” and “marine” in Figure 3. All marine species depend on a marine module, and the echolocating species depend on the echolocation module. The dependency graph is essentially a tree with extra flexibility; the modules can explain genes shared between species thought to be only distantly related by common descent. A module is not restricted to reusing code from a single source, but can freely reuse from multiple sources. Compare this to common descent where each species must almost exclusively draw from a single source: its ancestral species.

If the true explanation for life’s pattern of reuse is the dependency graph, why has it been interpreted as a nested hierarchy? According to the dependency graph hypothesis, the tree is simply a subset of the true dependency graph. Attempts to determine the correct tree of life have simply been uncovering the tree which best approximates the entire dependency graph. This works because some modules contribute much more similarity to species which depend on them than others. Life resembles a nested hierarchy because a nested hierarchical structure is similar enough to a dependency graph structure to approximate it.

However, while the nested hierarchy structure resembles a dependency graph it is not exactly the same. The dependency graph hypothesis does not simply predict the same pattern as common descent, nor common descent with unspecified deviations from the general pattern. Instead, it predicts instances of module reuse across taxonomic boundaries. Examples include the molecular convergence found in echolocating mammals [26] or marine mammals [27, 28]. Others have argued that mammals in general show a similar level of convergence [29, 30]. Moreover, virtually all sequenced genomes contain genes which have been interpreted as having arrived by horizontal gene transfer due to not fitting the hierarchical pattern [8, 11, 12, 31]. If the dependency graph hypothesis is correct, we should expect to find numerous examples of modules that appear to have been reused across taxonomic boundaries.

The dependency graph hypothesis draws on the idea of common design, by having reusable modules, as well as functional requirements, by restricting the reuse of modules via the dependency graph. The concept of a dependency graph draws not from an ad-hoc attempt to explain the data, but the actual process used to develop software. It is based on behaviors and practices that intelligent agents are known to use, not simply processes necessary to explain the data.

It is conceivable that a designer might follow a dependency graph purely out of the intrinsic constraints of design. However, it is more compelling if we adopt the working hypothesis that DNA is the product of a compiler. A compiler is a program used by software developers to automate, as much as possible, the process

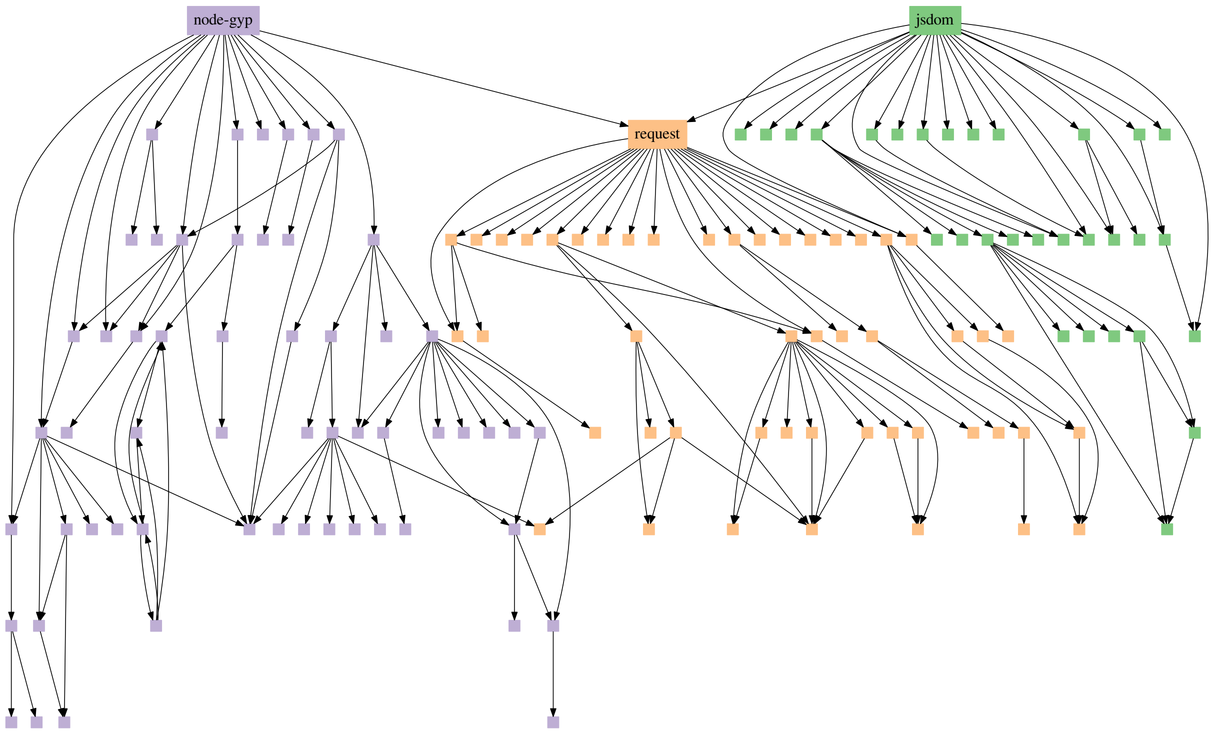

Figure 1: `jsdom` and `node-gyp` Dependency Graph. The dependency graph of `jsdom` and `node-gyp`. Green boxes are modules used by `jsdom`. Purple modules are used by `node-gyp`. Orange modules are used by both. Modules of particular interest have been labelled with the name of that module. The arrows go from a module to other modules that it depends on. **doi:**10.5048/BIO-C.2018.3.f1

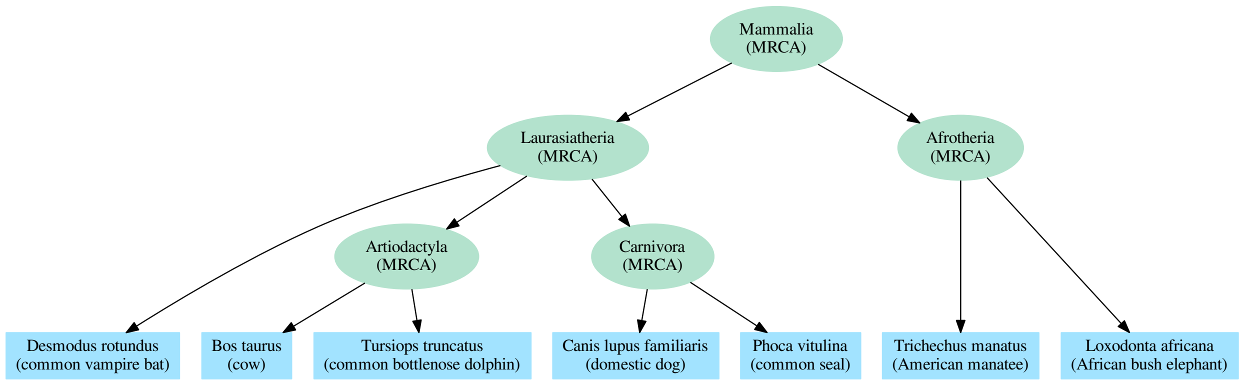
Mammalia (MRCA)
Laurasiatheria (MRCA)
Afrotheria (MRCA)
Artiodactyla (MRCA)
Carnivora (MRCA)
Desmodus rotundus (common vampire bat)
Bos taurus (cow)
Tursiops truncatus (common bottlenose dolphin)
Canis lupus familiaris (domestic dog)
Phoca vitulina (common seal)
Trichechus manatus (American manatee)
Loxodonta africana (African bush elephant)

Figure 2: A subset of the mammalian tree of life. Rectangles are extant species, and ellipses are postulated ancestral species, the most recent common ancestor (MRCA) of each taxonomic category. **doi:**10.5048/BIO-C.2018.3.f2

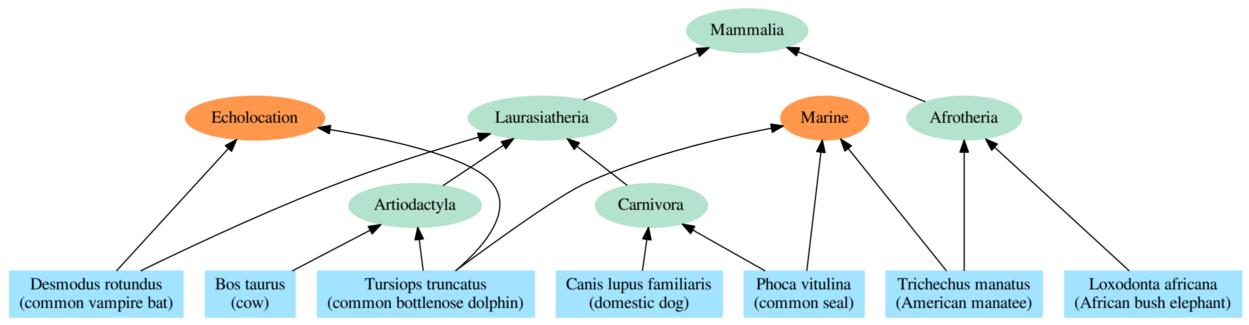
Mammalia
Echolocation
Laurasiatheria
Marine
Afrotheria
Artiodactyla
Carnivora
Desmodus rotundus (common vampire bat)
Bos taurus (cow)
Tursiops truncatus (common bottlenose dolphin)
Canis lupus familiaris (domestic dog)
Phoca vitulina (common seal)
Trichechus manatus (American manatee)
Loxodonta africana (African bush elephant)

Figure 3: A possible subset of the mammalian dependency graph of life. Rectangles are species, and ellipses are postulated modules. The orange elipses are modules postulated in addition to the standard taxonomic modules. doi:10.5048/BIO-C.2018.3.f3

of software development. It integrates the modules, performs optimizations, and determines the precise sequence of bytes that constitute a program. We postulate that a similar automated process was used to construct DNA.

PREDICTIONS

As mentioned, the dependency graph predicts a pattern similar to but distinct from common descent. Is this prediction actually borne out by the data? For this paper, we will focus on the distribution of genes amongst various species. There are a number of different techniques to represent this data. The simplest is to record the presence or absence of each gene family in each species [32–34]. Some models extend this idea by taking into account the number of members of a gene family, not simply their presence or absence [35–39]. Quantitative evaluations of common descent often consider exact DNA sequences [10, 13, 15]. For this study, we adopt the simplest representation: the presence/absence of gene families, leaving other representations and data to future research.

This choice is partially made for simplicity: as this is the first introduction of the dependency graph as an account for the pattern of life, we wish to avoid introducing any unnecessary complication. However, it also provides a number of advantages. This approach allows incorporating data from the whole genome and many species. It does not require either focusing on a few selected genes or a small selection of species. By operating at the resolution of a gene family, it will also not be affected by processes which produce within-family deviations from the standard tree. Addressing the gene family sizes or exact sequences is left to future research.

Given the distribution of gene families amongst species, what prediction does the dependency graph make about it? Firstly, we would expect that the biological data should fit a dependency graph better than a tree. We will evaluate the fit by using Bayesian model selection as described in the section below. The essential idea is that the better fitting model is the one that explains the data with least complexity, quantified as improbability. If the dependency graph hypothesis is correct, postulating modules should help explain otherwise improbable distributions of genes. On the other hand, if the dependency graph hypothesis is incorrect, data that deviates from the tree should not fit the graph either, and thus should not be made more probable by the hypothesis.

Furthermore, the dependency graph should not be too similar to the tree of life. We expect the tree to be present as a subset of the dependency graph, explaining how life has been interpreted as a hierarchy. The dependency graph hypothesis predicts that there will be many modules which do not correspond to a taxonomic category, and a substantial portion of the genes should be attributed to these non-taxonomic modules rather than taxonomic modules. On the other hand, common descent predicts there would be few non-taxonomic modules, and most genes would remain attributed to the taxonomic modules. Therefore, if the inferred dependency graph is simply the tree of life with a few minor additions, that would suggest that the dependency graph hypothesis is incorrect.

The method for determining which model is the better fit should be able to identify tree patterns as well as dependency graphs. In particular, we should be able to look at data produced by models or simulations of common descent, and see that they exhibit the tree pattern, not the pattern predicted by the dependency graph. Note that we do not claim that the dependency graph is the only way to produce data which fits the dependency graph better than a tree; given the wide variety of mechanisms that might be invoked along with common descent, it would not be surprising that some combination of those could produce data which fit a dependency graph better than a tree. In such a case, we would need to separately compare how that alternative evolutionary hypothesis performs against the dependency graph. But what would be surprising is if a model of common descent following a strict branching process nevertheless fit a dependency graph better than a tree. If it did, this would suggest that just about any dataset will fit the dependency graph better than a tree, no matter how it was produced, and would indicate that the model selection method is broken. Thus, we predict that as long as the process of common descent is approximately a strict branching process, then it will fit a tree better than a dependency graph.

Additionally, if we analyze a dependency graph inferred from data known to have been produced by common descent, it should remain similar to the tree. Any data that resembles modules should be a statistical fluke and thus rare and insignificant. There should be relatively few non-taxonomic modules inferred from the data. Most genes should remain attributed to the taxonomic modules.

The dependency graph hypothesis derives from dependency graphs used in software development. Consequently, we can offer predictions about software produced utilizing a dependency graph. Software does not have an exact equivalent to gene families, but we can take the nearest analogue and analyze software projects in the same manner as genomes. Like the biological data, the dependency graph hypothesis predicts that software should readily fit a hierarchical pattern, and yet fit the dependency graph better than a tree. In order for the dependency graph to be the correct explanation of the pattern of life, the proposed pattern must plausibly be mistaken for a hierarchical pattern. Consequently, it is necessary that software can be shown to fit a tree better than a null hypothesis. Moreover, the software must fit a dependency graph better than a tree.

To summarize, we have the following predictions:

- Biological data should fit the dependency graph better than a tree.
- Data produced by a process dominated by common descent or branching should fit a tree better than a dependency graph.
- Inferred graphs for biological data should contain many more non-taxonomic modules with many more genes than dependency graphs inferred from such data known to have been produced by common descent.
- Software should fit a dependency graph better than a tree, but a tree better than a null model.

Parsimony and Fitting

Most of the predictions require evaluating whether the data is a better fit to a tree or to a dependency graph. This paper utilizes Bayesian model selection to evaluate which model is a better fit. Many readers will be more familiar with statistical hypothesis testing which involves rejecting a null hypothesis on the basis of a p-value. However, this approach has been the subject of much criticism [40, 41], albeit with some defense [42].

Bayesian model selection proceeds by defining a model which assigns a probability to each possible set of species which could share a gene family. For example, common descent will assign higher probabilities to gene families being found in groups of species corresponding to taxonomic groups. On the other hand, the dependency graph will assign higher probabilities to gene families being found in groups of species that depend on the same modules. We evaluate the fit of the model to the data by assessing which model assigns a higher probability to the data.

For example, *Gallus gallus* (chickens) and *Meleagris gallopavo* (turkeys) are closely related birds, and thus are expected to share many genes by common descent. On the other hand, despite the similarity in names, *Taeniopygia guttata* (zebra finch) and *Danio rerio* (zebra fish) are only distantly related because one is a bird and the other a fish. As such, it should be relatively improbable to find genes shared *only* between these two species. But according to the Hogenom [43] dataset, there are nineteen gene families found only in this pair of species. The dependency graph model can assign high probabilities to both of these combinations by postulating a module shared between the pairs of species.

The astute reader will realize that a dependency graph model has an advantage over common descent in fitting the data because it can postulate modules to explain otherwise inexplicably distributed gene families. Consequently, it may seem that the dependency graph model will have a better fit to the data for almost any possible dataset. This is why we must also take into account the parsimony or complexity of the model. In particular, a model with a few simple parameters is preferred to one with many complicated parameters. One important reason for this is that a model with many complicated parameters can be tweaked to fit anything and thus tell us nothing about the data. Common descent postulates a relatively simple tree structure; the dependency graph postulates a much more complicated dependency graph. As such, the dependency graph is much less parsimonious than a tree model.

The need to consider both goodness-of-fit and parsimony is a general feature when comparing models. More complex models typically provide a better fit to the data even when those models do not correspond to reality. In such cases the more complicated model is not a better explanation of the data; instead it is *overfit*. To be a better explanation, the increase in fit to the data must outweigh the decreased parsimony of the more complicated model. The question thus is: does the dependency graph fit the biological data better enough to warrant its additional complexity?

Model selection theory provides the tools to answer this question. The fundamental idea is to penalize a model's degree of fit based on the complexity of the model [44]. A more complex model is considered less fit to make up for its lack of parsimony. In the case of Bayesian analysis, this is done by treating each parameter as a random variable. Instead of evaluating the goodness-of-fit of the biological data to a particular dependency graph, we evaluate the goodness-of-fit to a random dependency graph. Generally, this means that we compute the probability of the data using the law of total probability. For example, in the case of the dependency graph:

$$\sum_{g \in G} \Pr[\hat{G} = g] \Pr[D|\hat{G} = g] \tag{1}$$

where G is the set of all possible dependency graphs, g is a particular graph, D is that data being explained, and $\hat{G}$ is the actual dependency graph as a random variable. This gives us the fit of the probabilistic average graph instead of a particular graph, thus avoiding postulating a particular complex graph. In practice, this summation is often intractable and will be bounded or estimated. Effectively, this leaves us with a new parameterless model which can then be compared to other parameterless models.

Once the parameters have been resolved, we are left with two parameterless models that can be directly compared. This is done by computing the log Bayes factor:

$$\log_2 \frac{\Pr[D|A]}{\Pr[D|B]} \tag{2}$$

where D is the data, A is one model, and B is the other model. This is the log-ratio of the likelihoods of the data under both models after removing all parameters by use of the law of total probability. The log-ratio will be positive if the data fits model A better than model B, and negative if the data fits model B better than model A. The numbers are on a logarithmic scale and, according to a widely cited table [45, p. 432], 6.6 bits is considered decisive.

It is important to note that when a Bayes factor favors one model over the other, it does not necessarily mean that the favored model is more likely to be correct or is the better explanation. It means that the particular evidence under consideration fits one model better than the other, but the other model may still be more likely to be correct if we take into account other information. Here we simply wish to test predictions about which model better fits different datasets. The argument for the dependency graph hypothesis rests on fulfilled predictions rather than Bayesian inference.

Biological Data

In order to characterize biology we took data about how genes are classified into families and distributed in biological species, from nine different databases. Each of the databases considered classifies genes into gene families, groups, or clusters. Many of the databases included a broader selection of species, but were filtered to only include animals (metazoa).

Table 1 gives details on the nine databases used in this study. There is a variety both in the number of species as well as the number of gene families. Each database has its own classification of genes into families. Genes classified as belonging to the same family by one database are not necessarily classified as the same family in other databases. Some databases have much more fine grained gene families, dividing related genes into many families instead of a single family.

Pfam [51] primarily classifies domains, portions of genes, rather than the entire gene. Genes are classified into architectures, which correspond to a particular combination of domains on a gene. Thus genes are considered to be in the same family if they have the same combination of domains.

Ensembl Compara [48] and TreeFam [49] classify genes into a hierarchy instead of families. For this study, the hierarchy is collapsed, only keeping the broadest categories of classification. That is, only genes considered entirely unrelated are classified as different families. This groups genes together into families as broadly as possible. This is not intended to diminish the importance of modelling the greater structure found in such hierarchies, but explaining that structure is beyond the scope of this study.

UniRef [46] provides a number of different clusterings at different levels of similarity. The UniRef-50 clusters used here require that the members of a cluster have at least 50% similarity. The UniRef clusters are not intended to accurately reflect homologous genes, but are included for the sake of including a wide variety of possible definitions of gene families.

OMA [52] provides two different classifications: OMA Groups and OMA HOGs. The OMA Groups were used in this study because it follows a stricter strategy for which genes are considered to belong to the same family, and thus provided a more unique perspective on the classification of genes than the OMA HOGs, which would have been more similar to the other databases.

Each of the databases utilizes different techniques in the classification of gene families and are based on somewhat different species and biological data. Consult the documentation for the individual databases for more detail on these classifications.

Simulated Evolutionary Data

We analyzed a number of datasets produced using Evol-Simulator [54], an evolution simulator. It simulates the

Table 1: A summary of the gene databases used in this study.

Dataset	Families	Species	URL
UniRef-50 [46]	1,831,986	242	ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Eukaryota/
OrthoDB [47]	92,420	330	http://www.orthodb.org/v9/download/odb9_OG2genes.tab.gz
Ensembl [48]	22,972	69	ftp://ftp.ensembl.org/pub/release-87/xml/ensembl-compara/homologies/Compara.87.protein_default.alltrees.orthoxml.xml.gz
TreeFam [49]	23,491	104	http://www.treefam.org/static/download/xml/treefam9.protein.alltrees.orthoxml.xml.gz
Hogenom [43]	50,125	62	ftp://pbil.univ-lyon1.fr:21/pub/hogenom/release_06/FAMS_SEQ_SPECIES_UNIPROT.gz
EggNOG [50]	38,965	89	http://eggnogdb.embl.de/download/eggnog_4.5/data/euNOG/euNOG.members.tsv.gz
Pfam [51]	84,410	145	ftp://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam31.0/database_files/pfamseq.txt.gz
OMA [52]	311,486	132	http://omabrowser.org/All/oma-groups.txt.gz
HomoloGene [53]	29,323	13	ftp://ftp.ncbi.nih.gov/pub/HomoloGene/build68/

Table 2: The parameters for the EvolSimulator simulations

Parameter	EvolSim 1	EvolSim 2	EvolSim 3	EvolSim 4	EvolSim 5
numIterations	10000	10000	10000	10000	10000
selectionModel	0	1	2	1	2
numberOfHabitats	1	1	1	5	3
numberOfHabitatNiches	1	1	1	15	9
randomLGTProb	1.0	1.0	1.0	1.0	0.125
divergenceLGTProb	0.0	0.0	0.0	0.0	0.125
geneCompLGTProb	0.0	0.0	0.0	0.0	0.125
gcLGTProb	0.0	0.0	0.0	0.0	0.125
habitatLGTProb	0.0	0.0	0.0	0.0	0.125
orthologReplacementProbability	1.0	1.0	1.0	1.0	0.5

evolution of genes amongst a collection of species descended from a single common ancestor. It includes models of mutation, selection, and lateral gene transfer. It is very customizable, allowing a large variety of parameters to control the simulation. For this simulation, we ran the simulation using five different sets of parameters summarized in Table 2. All parameters not listed in table are set to the values in the example parameters file provided by EvolSimulator. To understand the meaning of the parameters, see the documentation for EvolSimulator. The resulting genes were processed by OrthoFinder [55] using the actual species tree produced by EvolSimulator, to determine gene families. Apart from the species tree, default parameters were used. This produces a synthetic dataset of genes which is known to have been produced by a process of common descent.

Software Data

We also produced a dataset from a number of JavaScript single page applications produced by a compiler. The compiler utilizes a dependency graph of modules to determine the code that is inserted into each application. We analyze the resulting files for units of structure which come as near as possible to the idea of a gene. We treat these units as the equivalent of genes, and gene families as units with the same basic "shape." The essential idea is that we have taken an analog to genes in the output of a compiler and used it to produce a dataset of gene families and genes for these "species" of JavaScript single page applications. This allows us to test predictions about how well software fits the various models. See the Methods section for details on how this data was collected and interpreted.

Models

This study will consider three models, depicted in Figure 4. The first model, on the left, is the null model which postulates that there is no pattern to the reuse of gene families in biology. Unlike common descent or the dependency graph, there are no relationships between any species. Instead, any genes shared between multiple species are attributed to a "toolbox": a collection of genes that can be reused across all species. This toolbox is depicted as the circle in the middle of the figure, whereas all the squares represent species that use some genes from the toolbox.

The second model, in the middle, is the tree or common descent model. It postulates that any genes shared between two species derive from another species ancestral to the extant species. The circles are the ancestral species, and the squares are the extant species. Each ancestral species itself (other than the root) has another species ancestral to it.

For the tree, we adopt a simple model of gene arrival and propagation. Each gene family arises in exactly one species. Somewhat similar genes may arise in different species, but we assume that they will not be classified as belonging to the same family. A gene family is inherited by any descendant species, where it may be lost. Once lost, it cannot be regained. This means that if two species share a gene family, their most recent common ancestor must also have had that gene family. If a species loses a gene family, none of its descendent species can have that gene family. This models a basic hierarchical pattern to the distribution of genes.

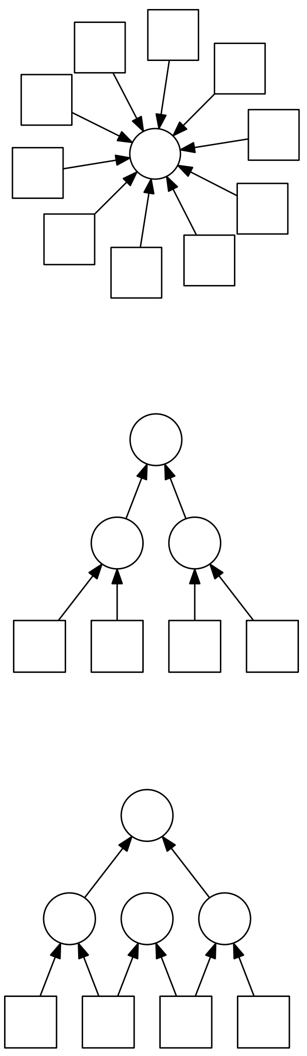

Figure 4: A depiction of three models: null (top), tree (middle), and the dependency graph (bottom). The squares represent species, and the circles represent a shared toolbox, ancestral species, or modules depending on the model.
doi:10.5048/BIO-C.2018.3.f4

Genes within families do undergo splitting, sequence changes, and functional changes. However, such events are below the resolution of the model. This model is only concerned with the presence or absence of the gene family in a particular species. However, a gene may evolve into a new gene family. The assumption of the model is that a particular gene family can only arise once, whether de novo or from an existing gene family.

Critics will be quick to point out that there are a variety of mechanisms to explain deviations from the hierarchical pattern, such as incomplete lineage sorting, gene flow, horizontal gene transfer, convergent evolution, and gene resurrection. These mechanisms occur in nature, but are not included in this model. Recall that we are testing predictions about whether a particular dataset will more closely fit a tree or the dependency graph. Mechanisms which produce deviations from the tree are not relevant to that prediction.

We assume that the true tree of life would be similar to the NCBI [53] hierarchy. The NCBI database contains a disclaimer, “The NCBI taxonomy database is not an authoritative source for nomenclature or classification - please consult the relevant scientific literature for the most reliable information.” However, the NCBI database is the only source for lineage information we found that consistently contained all species IDs referred to in the various gene family databases. We make some adjustments, as described in the Methods section, to the NCBI hierarchy, where this allows the biological data to fit the tree of life better.

The dependency graph model is as similar as possible to the tree model. Instead of ancestral species, the model has modules. Instead of a single ancestor, each module may have multiple modules that it depends on. These are called *dependencies*. Each species is a top-level module: no other modules depend on a species module. Every gene family is introduced in a single module and inherited by all modules that depend on that module. Even if a designer were to design two different genes for the same purpose, we would expect them to end up being different enough to be classified into different families. A module may lose a gene that was inherited from a dependency. Structurally, both models are the same except for common descent having a single ancestor where the dependency graph has multiple dependencies.

As part of the analysis, we infer a dependency graph. We do not claim that it is the true dependency graph, or the best dependency graph; it is used only to provide a lower bound on the performance of dependency graph hypothesis, as will be explained in the Methods section. Its validity as a bound does not depend on how the graph was obtained. The graph is obtained by starting with a dependency graph based on the tree from NCBI. Modules are added to and removed from the graph when this would tighten the lower bound on the dependency graph hypothesis. The exact algorithm for this inference is described in the Methods section.

RESULTS

Synthetic Data

Table 3 presents the log Bayes factors for the synthetic datasets. Positive log Bayes factors show support for the first model over the second model, whereas negative log Bayes factors show the opposite. In the case of EvolSimulator, both the tree and the dependency graph are consistently a better fit than the null model. However, the tree is also consistently better supported than the dependency graph. This confirms one of the predictions, data actually produced by a branching process does in fact fit a tree better than the dependency graph. The analysis was unable to postulate enough modules in order to obtain a better fit to the data.

The JavaScript applications fit the tree or the dependency graph better than the null model. However, the dependency graph is preferred to the tree. This again confirms one of the predictions, software can exhibit a hierarchical signal while being produced by a dependency graph. Nevertheless, it still fits the dependency graph better than the hierarchical pattern.

Figure 5 shows the tree used for the common descent model of JavaScript applications. The process used to determine this tree is described in the Methods sections. It involves trying many possible trees to determine the tree which gives the best fit of the data to the tree model. Figure 6 shows a simplified version of the true dependency graph for these same applications. The colors correspond to frameworks. A framework is a module that provides the basic tools necessary to build an application. It is like a module that defines a body plan shared amongst several species. Each individual species builds on this basic plan to define the actual species. A framework will contain a large amount of code, consequently an application will share much code with other applications which use the same framework.

Upon inspection, we can see that the tree is actually reflecting the reality of the dependency graph. In the tree, applications using the same framework tend to cluster together, due to the shared code. However, applications which share non-framework modules are put closer together in an attempt to explain that reuse. The applications using the `angular2` framework (tan) are oddly placed, branching out of the `react` based applications (pink). However, this is because `hn-ng2` and `react-news` both use the large `firebase` module. `Mamba`, `vim-awesome`, and `sound-redux` are pulled closer to the root because they reuse the `lodash` and `immutable` modules which are also used by the `angular` applications (green). The tree is working to approximate the dependency graph.

Biological Data

Table 4 depicts the log Bayes factor for three models and nine different biological databases. It shows how well the three models: the null model, tree, and dependency graph fit the biological data. Common descent has log Bayes factors ranging from 10^5 to 10^7 bits over the null model. Common descent is overwhelmingly a better fit than the null model. This confirms the results of previous quantitative analyses of the nested hierarchy. The biological data unavoidably does exhibit some resemblance to a hierarchy.

Table 3: The log Bayes factors for the models in the synthetic datasets

Dataset	Tree vs Null	D. Graph vs Null	D. Graph vs Tree
JavaScript	9,791	15,078	4,749
EvolSim 1	83,301	83,023	-886
EvolSim 2	45,939	45,005	-1,582
EvolSim 3	29,851	28,485	-1,598
EvolSim 4	27,982	26,554	-1,793
EvolSim 5	36,120	35,553	-874

Table 4: The log Bayes factors for combinations of models and datasets

Dataset	Tree vs Null	D. Graph vs Null	D. Graph vs Tree
UniRef-50	6,193,801	6,308,988	111,823
OrthoDB	9,214,606	9,730,055	515,450
Ensembl	875,350	962,274	86,924
TreeFam	1,362,985	1,403,952	40,967
Hogenom	884,815	1,022,243	137,428
EggNOG	1,497,174	1,579,650	82,476
Pfam	1,173,599	1,251,841	78,244
OMA	3,265,608	3,451,745	184,777
HomoloGene	106,010	116,080	10,064

However, while the tree is a better fit than the null model, the dependency graph is a better fit than the tree. Even in the biological gene database least favorable to the dependency graph, HomoloGene, the log Bayes factor is in favor of the dependency graph by over 10,000 bits. Recall that 6.6 bits is commonly considered decisive. The data is over 10^{3000} times more likely to be produced by the dependency graph model than the tree model. This is very far beyond decisive, delivering a clear confirmation of the prediction. However, keep in mind that a clear confirmation of a particular prediction, while providing evidence for the hypothesis, is not the same thing as a clear confirmation of the hypothesis itself.

Biological Graphs

Introduction

This section will present graphs inferred from the biological data. However, we must urge caution in interpreting these graphs. These graphs constitute the first attempt

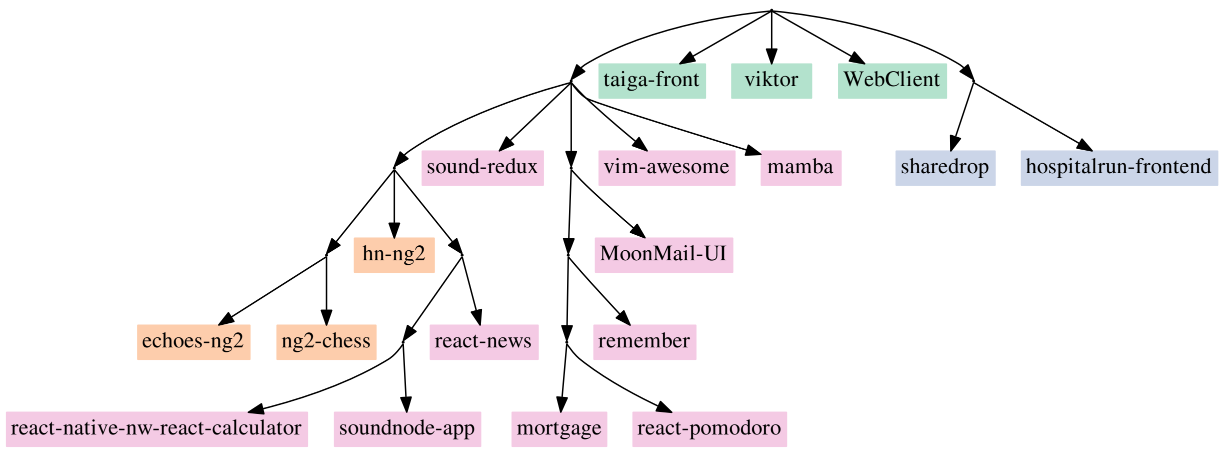

Figure 5: The phylogenetic tree for the JavaScript applications doi:10.5048/BIO-C.2018.3.f5

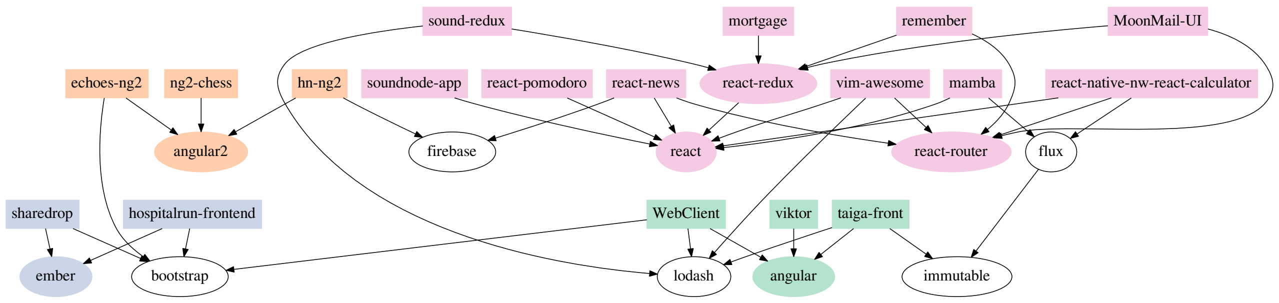

Figure 6: A simplified dependency graph for the JavaScript applications. The rectangles correspond to individual programs, whereas coloured ellipses correspond to javascript 'frameworks' and white ellipses correspond to other modules. doi:10.5048/BIO-C.2018.3.f6

to fit a dependency graph to biology, not any sort of attempt at a definitive reconstruction of the graph. The best fitting dependency graph might be substantially different than the graphs presented here. All that we can conclude with certainty is that the best fitting graph fits at least as well as the graphs presented here. Furthermore, the strength of the case for the dependency graph hypothesis rests on getting consistent results across many gene databases. Examples of inferred modules, however striking, come from one database and are thus less supported than the overall result.

Statistics

One of the predictions was that the inferred dependency graph should not simply be the tree of life with a few additions. Table 5 shows the number of modules of each type in the graphs inferred from the various biological databases. The dependency graph includes a variety of modules, the *species modules* which correspond to each species, the *taxonomic* modules which correspond to taxonomic categories, and the *non-taxonomic* modules which do not correspond to either individual species or taxonomic categories. The biological data set shows over ten times more non-taxonomic modules than taxonomic modules. Compare this to Table 6; the corresponding statistics for the data produced by EvolSimulator. In contrast, the EvolSimulator datasets show a much smaller proportion of non-taxonomic modules. This confirms our prediction: we find many more non-taxonomic modules in biological data than in data known to have been produced by common descent.

Table 7 shows the number of genes which the fitting method attributed to each type of module for the biological datasets. The species column contains gene families that are only found in one species. It varies substantially from database to database depending on how they have defined family. The taxonomic column contains gene families introduced in modules corresponding to taxonomic categories. The non-taxonomic column contains gene families introduced in modules that do not correspond to taxonomic categories or species. In all of the biological databases except HomoloGene, there are more genes in the non-taxonomic modules than in the taxonomic modules. HomoloGene still has a substantial number of gene families in non-taxonomic modules. Table 8 shows the corresponding results from the EvolSimulator datasets. In contrast to the biological case, only a small fraction of the genes in the EvolSimulator datasets are assigned to non-taxonomic modules. This again confirms the prediction; many genes are attributed outside of the taxonomic modules in biological datasets but not in the data known to have been produced by common descent.

The inferred biological dependency graphs are not simply the tree of life with a few modules tacked on.

Small Examples

We will now consider a few small examples of cases which our model-fitting method inferred to be explained by modules. A striking example can be found in *Nematostella vectensis* (scarlet sea anemone) and *Branchiostoma floridae* (Florida lancelet). These are distantly related organisms with the anemone being in the phylum Cnidaria and the lancelet being in the phylum Chordata. Nevertheless, they contain between 25 and 564 (depending on the database consulted) gene families found in both species but in no other metazoan species in the database. In all datasets where both species are present, a module is inferred to exist to explain the genes found in both species.

Figure 7 depicts a case from the EggNog [50] database. Three species, *Gadus morhua* (atlantic cod), *Gasterostus aculeatus* (three-spined stickleback), *Xiphophorus maculatus* (southern platyfish) share fifty-four genes not found elsewhere in any other metazoan species. Note especially, their absence from the other closely related species depicted in Figure 7. Consequently, a module is inferred to exist to explain the distribution of these fifty-four genes.

One might suspect that any combination of three of these species would yield a similar number of gene families found only in those species. However, Table 9 shows that this is not the case. The table shows, for every possible combination of three species, how many gene families are found in all of those species, but in no other metazoan species. We see that most combinations have few gene families, but a few have many gene families. The dependency graph model fits the data because it is able to postulate a large module with many genes to explain this. If every combination had many genes, the dependency graph model would not fit because it would have to postulate a large number of modules to explain every combination.

Figure 8 depicts the inferred dependency graph from the HomoloGene database for *Gallus gallus* (chickens), *Danio rerio* (zebrafish), and *Mus musculus* (mice). HomoloGene is the smallest database, only containing thirteen metazoan species. Consequently, the number of modules inferred from the data is also small, and thus we can depict all of the inferred modules in a figure. This graph shows all the modules depended on by these three species. Each species has a number of modules not shared with the other two species. There are modules shared between all possible combinations of these three species.

Slices of the Graph

There is a challenge in presenting the inferred dependency graphs because they are very large. Even the smallest dataset, HomoloGene, has 145 modules and would be infeasible to present visually in a readable way. Instead, we present a subset of the overall graph, showing only a selected subset of the species and the most significant

Table 5: Number of modules of various types for the inferred biological dependency graphs

Dataset	Species	Taxonomic	Non-Taxonomic
UniRef-50	242	93	7,874
OrthoDB	330	181	9,386
Ensembl	69	53	1,328
TreeFam	104	74	919
Hogenom	62	40	1,388
EggNOG	89	64	1,743
Pfam	145	77	2,601
OMA	132	60	9,831
HomoloGene	13	11	142

Table 6: Number of modules of various types for the inferred EvolSimulator dependency graphs

Dataset	Species	Taxonomic	Non-Taxonomic
EvolSim 1	43	30	74
EvolSim 2	43	24	64
EvolSim 3	43	23	59
EvolSim 4	40	27	61
EvolSim 5	43	29	43

Table 7: Number of gene families found in each type of module for biological data

Dataset	Species	Taxonomic	Non-Taxonomic
UniRef-50	1,441,028	171,063	219,895
OrthoDB	662	24,637	67,121
Ensembl	52	8,948	13,972
TreeFam	281	10,425	12,785
Hogenom	11,810	17,867	20,448
EggNOG	2,559	16,154	20,252
Pfam	50,169	4,685	29,556
OMA	39,679	70,928	200,879
HomoloGene	3,538	17,481	8,304

Table 8: Number of gene families found in each type of modules for EvolSimulator

Dataset	Species	Taxonomic	Non-Taxonomic
EvolSim 1	25,594	10,149	746
EvolSim 2	18,039	5,907	811
EvolSim 3	17,124	3,248	615
EvolSim 4	29,871	3,949	763
EvolSim 5	15,685	4,276	246

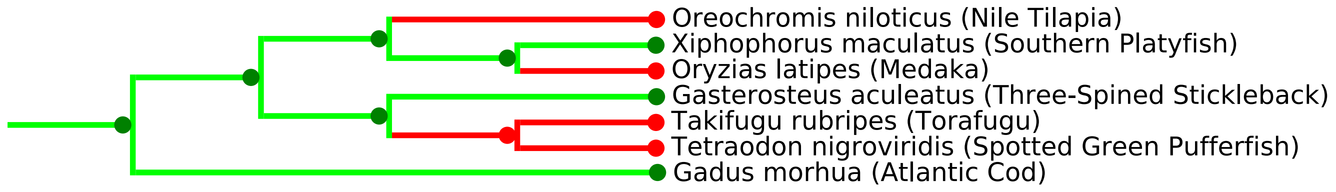
Oreochromis niloticus (Nile Tilapia)
Xiphophorus maculatus (Southern Platyfish)
Oryzias latipes (Medaka)
Gasterosteus aculeatus (Three-Spined Stickleback)
Takifugu rubripes (Torafugu)
Tetraodon nigroviridis (Spotted Green Pufferfish)
Gadus morhua (Atlantic Cod)

Figure 7: The tree of life for several species of fish from EggNog. Species in green share fifty-four genes not found elsewhere. The most parsimonious common descent model is that the same set of fifty-four genes were lost in three different lineages. doi:10.5048/BIO-C.2018.3.f7

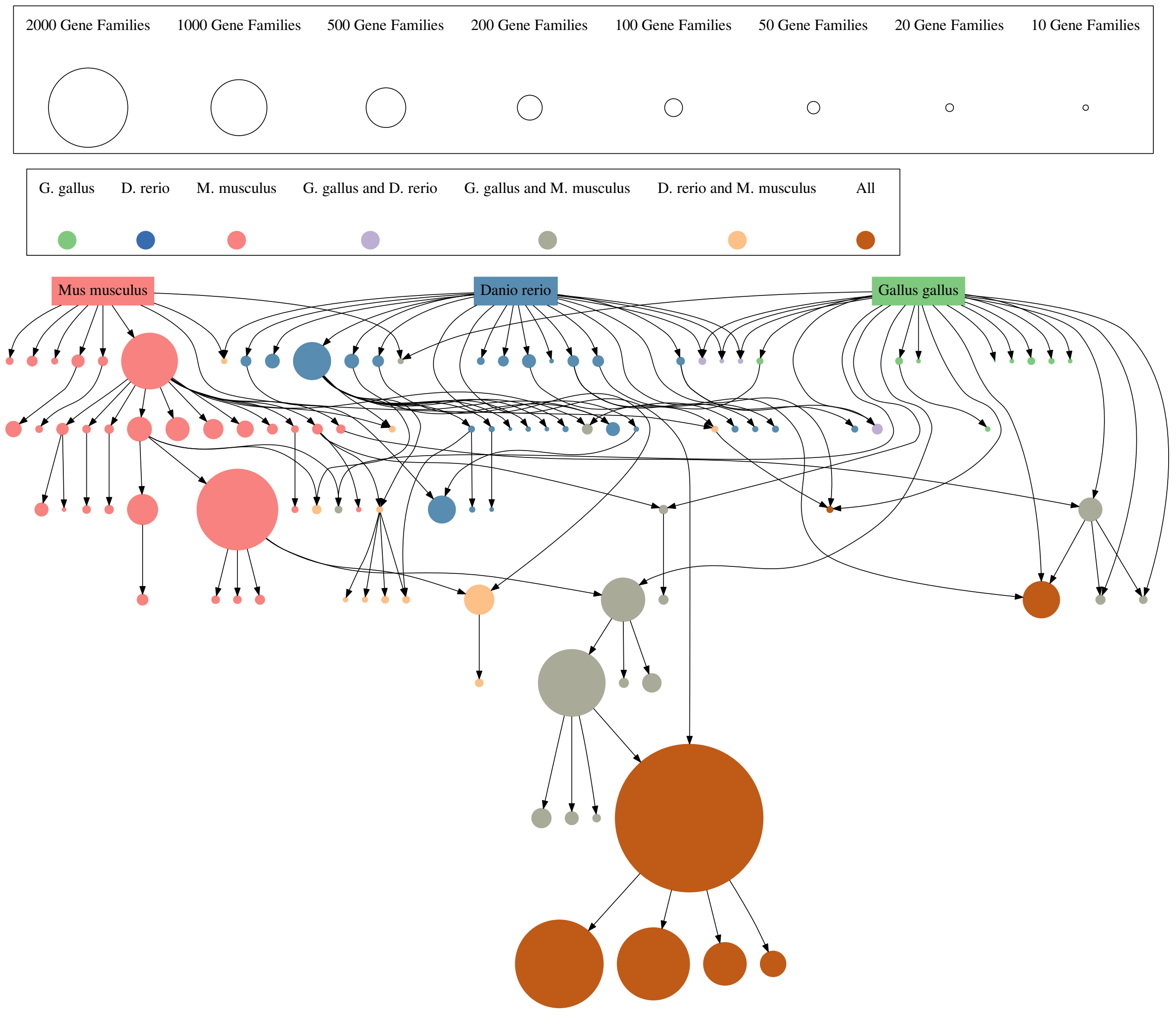

Figure 8: HomoloGene inferred dependency graph for *Gallus gallus*, *Danio rerio*, and *Mus musculus*. The colors represent which species depend on the module. The area of the circles represent the number of gene families gained in a module. doi:10.5048/BIO-C.2018.3.f8

Table 9: Number of gene families found only in a combination of three species from closely related species of fish.

Species Combination			Gene Families	Clade?
Gadus morhua	Xiphophorus maculatus	Oryzias latipes	2	No
Gadus morhua	Xiphophorus maculatus	Oreochromis niloticus	8	No
Gadus morhua	Xiphophorus maculatus	Takifugu rubripes	1	No
Gadus morhua	Xiphophorus maculatus	Gasterosteus aculeatus	**54**	No
Gadus morhua	Xiphophorus maculatus	Tetraodon nigroviridis	0	No
Gadus morhua	Oryzias latipes	Oreochromis niloticus	2	No
Gadus morhua	Oryzias latipes	Takifugu rubripes	0	No
Gadus morhua	Oryzias latipes	Gasterosteus aculeatus	**25**	No
Gadus morhua	Oryzias latipes	Tetraodon nigroviridis	1	No
Gadus morhua	Oreochromis niloticus	Takifugu rubripes	1	No
Gadus morhua	Oreochromis niloticus	Gasterosteus aculeatus	0	No
Gadus morhua	Oreochromis niloticus	Tetraodon nigroviridis	0	No
Gadus morhua	Takifugu rubripes	Gasterosteus aculeatus	4	No
Gadus morhua	Takifugu rubripes	Tetraodon nigroviridis	0	No
Gadus morhua	Gasterosteus aculeatus	Tetraodon nigroviridis	6	No
Xiphophorus maculatus	Oryzias latipes	Oreochromis niloticus	8	**Yes**
Xiphophorus maculatus	Oryzias latipes	Takifugu rubripes	1	No
Xiphophorus maculatus	Oryzias latipes	Gasterosteus aculeatus	7	No
Xiphophorus maculatus	Oryzias latipes	Tetraodon nigroviridis	3	No
Xiphophorus maculatus	Oreochromis niloticus	Takifugu rubripes	2	No
Xiphophorus maculatus	Oreochromis niloticus	Gasterosteus aculeatus	10	No
Xiphophorus maculatus	Oreochromis niloticus	Tetraodon nigroviridis	2	No
Xiphophorus maculatus	Takifugu rubripes	Gasterosteus aculeatus	0	No
Xiphophorus maculatus	Takifugu rubripes	Tetraodon nigroviridis	2	No
Xiphophorus maculatus	Gasterosteus aculeatus	Tetraodon nigroviridis	2	No
Oryzias latipes	Oreochromis niloticus	Takifugu rubripes	5	No
Oryzias latipes	Oreochromis niloticus	Gasterosteus aculeatus	4	No
Oryzias latipes	Oreochromis niloticus	Tetraodon nigroviridis	0	No
Oryzias latipes	Takifugu rubripes	Gasterosteus aculeatus	0	No
Oryzias latipes	Takifugu rubripes	Tetraodon nigroviridis	1	No
Oryzias latipes	Gasterosteus aculeatus	Tetraodon nigroviridis	6	No
Oreochromis niloticus	Takifugu rubripes	Gasterosteus aculeatus	3	No
Oreochromis niloticus	Takifugu rubripes	Tetraodon nigroviridis	3	No
Oreochromis niloticus	Gasterosteus aculeatus	Tetraodon nigroviridis	0	No
Takifugu rubripes	Gasterosteus aculeatus	Tetraodon nigroviridis	1	**Yes**

dependencies and modules for those species. Only modules depended on by the shown species are kept. We skip modules with only one dependant, showing the dependencies of that module as direct dependencies of the dependant. For example, if Module A depended on Module B which depended on Module C, but Module A is the only dependant of Module B, Module B is skipped, and Module A is depicted as depending directly on Module C. The smallest modules, in terms of total genes in the module including those inherited from dependencies, are removed.

Additionally, in each graph those modules which correspond to clades in the NCBI hierarchy are colored green. Modules inferred to exist but which do not correspond to taxonomic categories are colored orange. This allows the easy identification of those parts of the dependency graph that do not correspond to simply reinterpreting the tree of life.

Figure 9 shows the graph for the HomoloGene database. Many of the dependencies in the graph match the expectations of common descent. *Pan troglodytes* (chimpanzees) and *Homo sapiens* (humans) share a module as do *Rattus norvegicus* (rats) and *Mus musculus* (mice). However, there are also surprises. *Bos taurus* (cows) shares a module with the rodents, specifically *Rattus norvegicus* and *Mus musculus*. *Pan troglodytes* (chimpanzees) shares a module with *Macaca mulatta* (rhesus monkeys). Another module is shared between *Gallus gallus*, *Xenopus tropicalis*, and *Danio rerio*.

Figure 10 shows a simplified dependency graph inferred from the OrthoDB database for a selection of species classified as Euteleosteomorpha, a cohort of ray-finned fishes, by NCBI.

Figure 11 depicts the dependency graph inferred from the OMA database for six primate species. *Homo sapiens* share a module with *Pan troglodytes* (chimpanzees) as expected, but also share a module with *Macaca mulatta* (rhesus macaques). There are many non-taxonomic modules corresponding to various combinations of these six species.

Figure 12 depicts the dependency graph inferred between nine familiar mammalian species. *Mus musculus* (mice) and *Rattus norvegicus* (rats) share a module as would be expected. So do *Felis catus* (cats) and *Canis lupus familiaris* (dogs). Other species show some unexpected sharing, *Sus scrofa* (wild boar) shares a module with *Felis catus* and *Canis lupus familiaris*. *Mus musculus* and *Ratus norvegicus* share a module with *Felis catus* and *Canis lupus familiaris*.

CONCLUSIONS

Explaining the approximate nested hierarchy has been a long standing challenge to common design. No account of this pattern has achieved widespread acceptance amongst those holding to common design. We have proposed a novel explanation, the dependency graph. The predictions of the dependency graph hypothesis set out in this

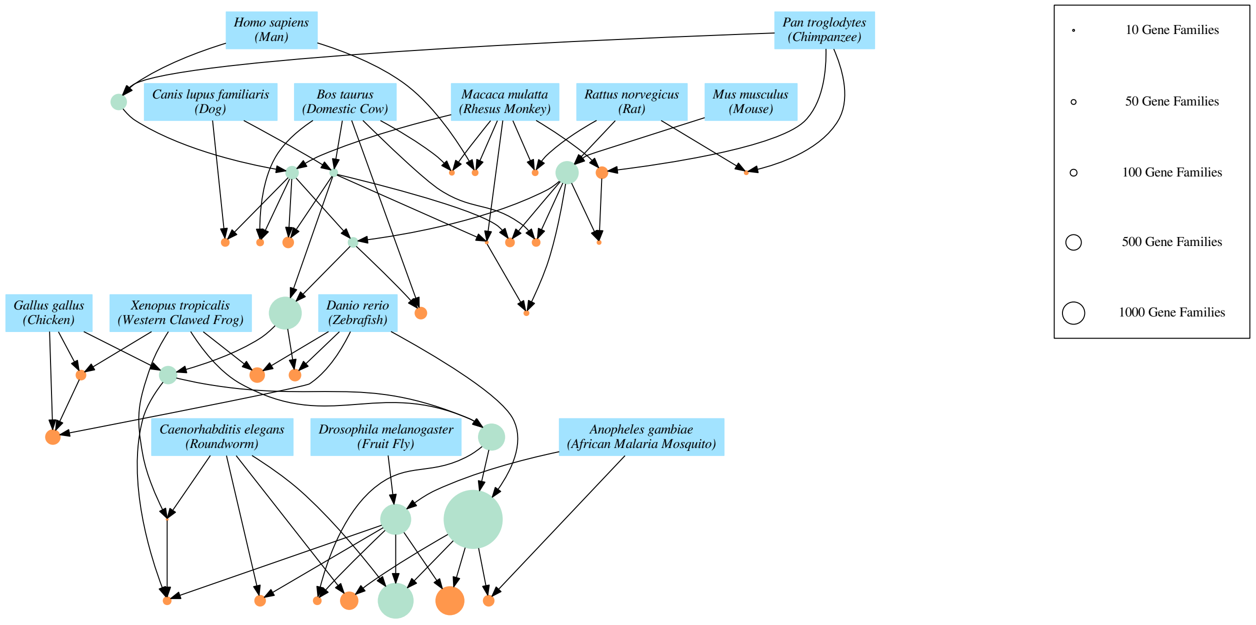
Homo sapiens (Man)
Pan troglodytes (Chimpanzee)
Canis lupus familiaris (Dog)
Bos taurus (Domestic Cow)
Macaca mulatta (Rhesus Monkey)
Rattus norvegicus (Rat)
Mus musculus (Mouse)
Gallus gallus (Chicken)
Xenopus tropicalis (Western Clawed Frog)
Danio rerio (Zebrafish)
Caenorhabditis elegans (Roundworm)
Drosophila melanogaster (Fruit Fly)
Anopheles gambiae (African Malaria Mosquito)
10 Gene Families
50 Gene Families
100 Gene Families
500 Gene Families
1000 Gene Families

Figure 9: Subset of the dependency graph inferred from the HomoloGene database The graph only shows modules with at least 100 genes, including genes inherited from dependencies. Rectangles correspond to species, circles correspond to modules. The size of a module is proportional to the number of genes gained in that module (genes inherited from dependencies are not counted). Green circles are modules corresponding to taxonomic categories. Orange circles are modules that do not correspond to a taxonomic category. **doi:**10.5048/BIO-C.2018.3.f9

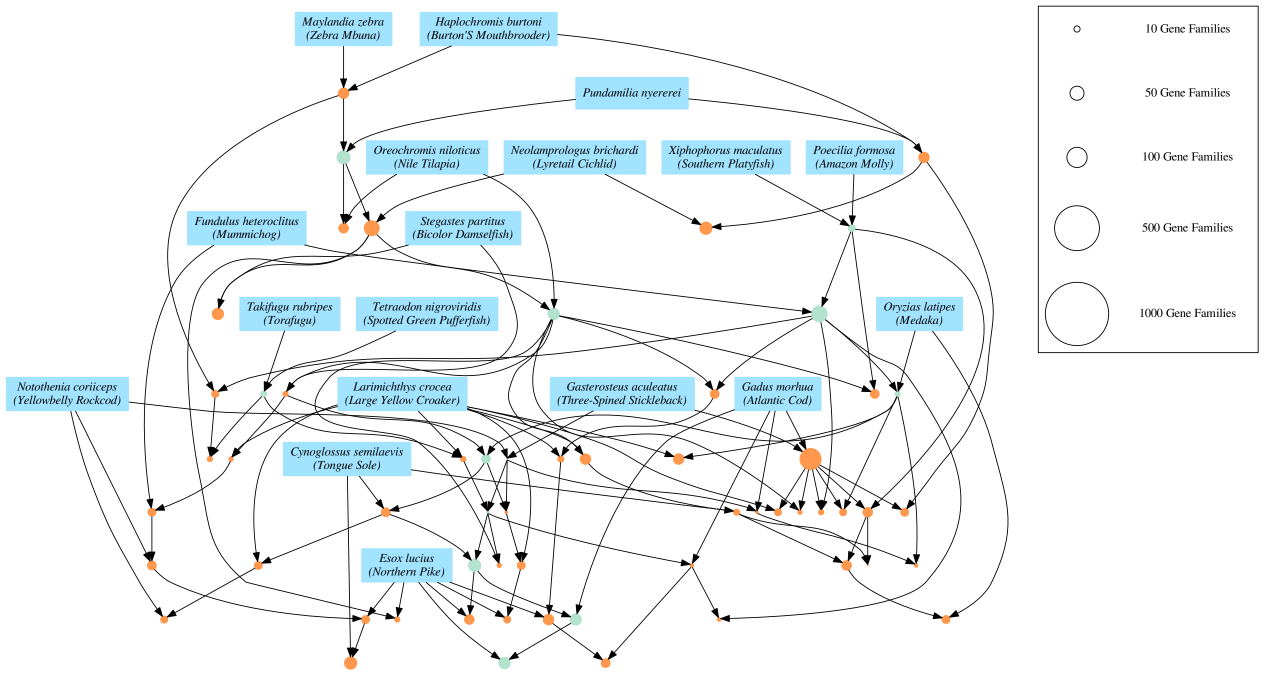
Maylandia zebra (Zebra Mbuna)
Haplochromis burtoni (Burton'S Mouthbrooder)
Pundamilia nyererei
Oreochromis niloticus (Nile Tilapia)
Neolamprologus brichardi (Lyretail Cichlid)
Xiphophorus maculatus (Southern Platyfish)
Poecilia formosa (Amazon Molly)
Fundulus heteroclitus (Mummichog)
Stegastes partitus (Bicolor Damselfish)
Takifugu rubripes (Torafugu)
Tetraodon nigroviridis (Spotted Green Pufferfish)
Oryzias latipes (Medaka)
Notothenia coriiceps (Yellowbelly Rockcod)
Larimichthys crocea (Large Yellow Croaker)
Gasterosteus aculeatus (Three-Spined Stickleback)
Gadus morhua (Atlantic Cod)
Cynoglossus semilaevis (Tongue Sole)
Esox lucius (Northern Pike)
10 Gene Families
50 Gene Families
100 Gene Families
500 Gene Families
1000 Gene Families

Figure 10: Subset of the dependency graph inferred from the OrthoDB database restricted to Euteleosteomorpha. The graph only shows modules with at least 50 genes, including genes inherited from dependencies. Rectangles correspond to species, circles correspond to modules. The size of a module is proportional to the number of genes gained in that module (genes inherited from dependencies are not counted). Green circles are modules corresponding to taxonomic categories. Orange circles are modules that do not correspond to a taxonomic category. **doi:**10.5048/BIO-C.2018.3.f10

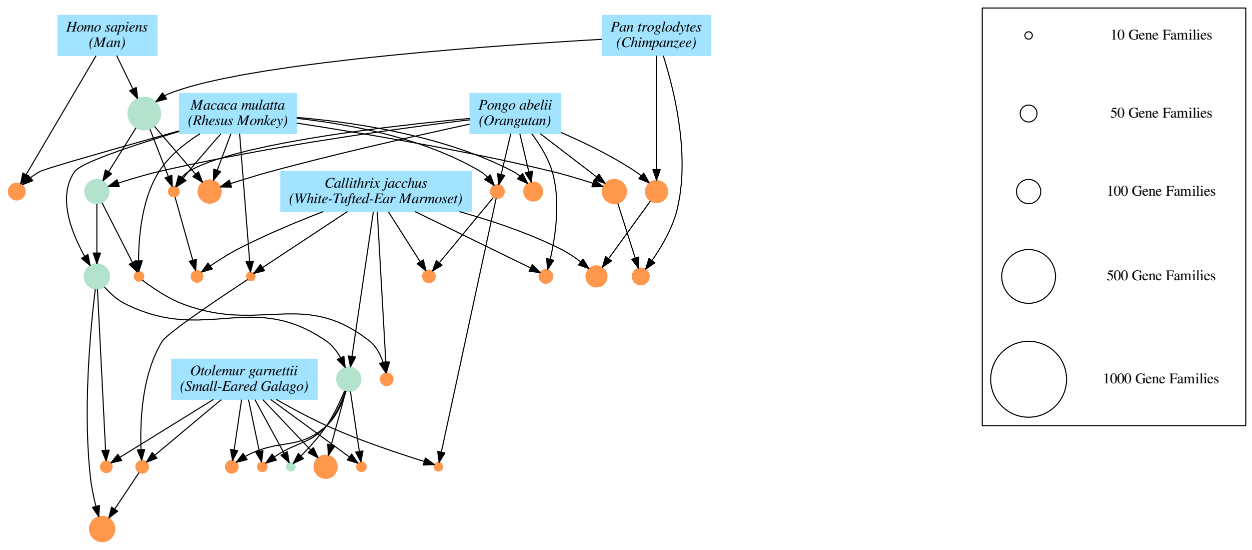
Homo sapiens (Man)
Pan troglodytes (Chimpanzee)
Macaca mulatta (Rhesus Monkey)
Pongo abelii (Orangutan)
Callithrix jacchus (White-Tufted-Ear Marmoset)
Otolemur garnettii (Small-Eared Galago)
10 Gene Families
50 Gene Families
100 Gene Families
500 Gene Families
1000 Gene Families

Figure 11: A subset of the dependency graph inferred from the OMA database, limited to six primate species. The graph only shows modules with at least 100 genes, including genes inherited from dependencies. Rectangles correspond to species, circles correspond to modules. The size of a module is proportional to the number of genes gained in that module (genes inherited from dependencies are not counted). Green circles are modules corresponding to taxonomic categories. Orange circles are modules that do not correspond to a taxonomic category. **doi:**10.5048/BIO-C.2018.3.f11

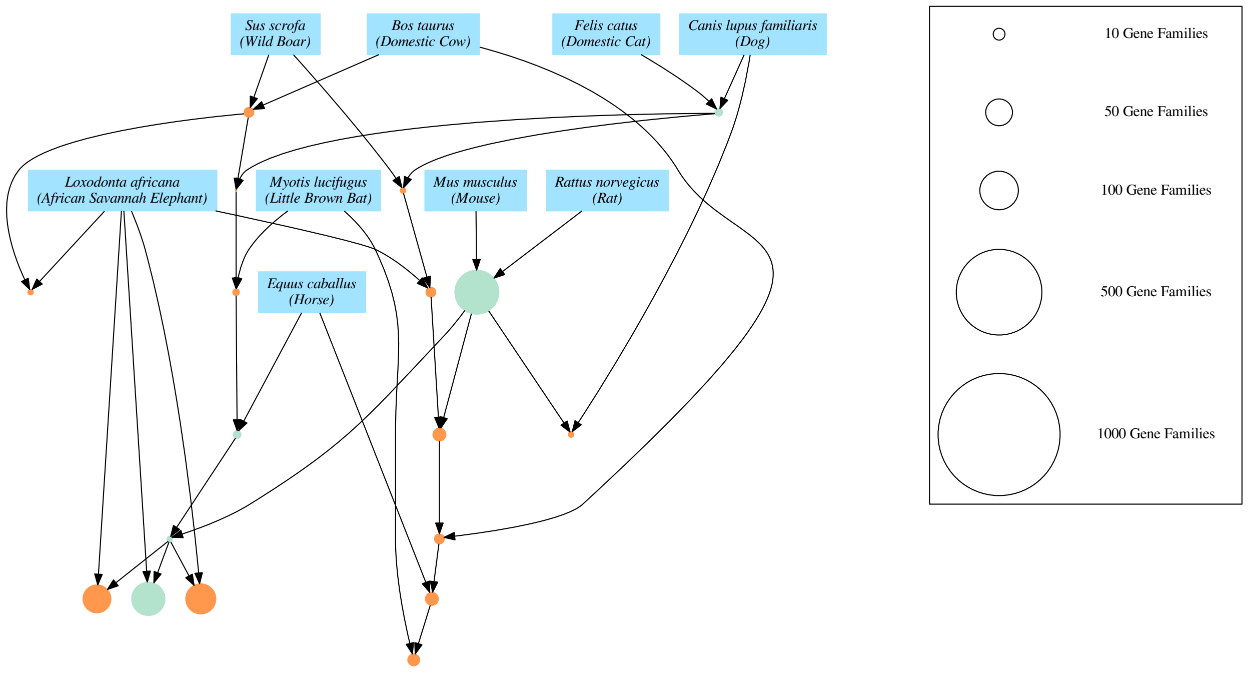

Figure 12: A subset of the dependency graph inferred from the Ensembl database, limited to nine familiar mammal species. The graph only shows modules with at least 50 genes, including genes inherited from dependencies. Rectangles correspond to species, circles correspond to modules. The size of a module is proportional to the number of genes gained in that module (genes inherited from dependencies are not counted). Green circles are modules corresponding to taxonomic categories. Orange circles are modules that do not correspond to a taxonomic category. **doi:**10.5048/BIO-C.2018.3.f12

paper have been shown to be correct. The biological data was a better fit to a dependency graph than to a tree. The data produced by a simulated process of common descent was a better fit to a tree than to a dependency graph. The data produced by a compiler was both a better fit to a dependency graph than a tree, and a better fit to a tree than to the null model. The inferred biological dependency graphs contained were not simply the tree of life with a few additions, but instead contained many additional modules.

Further research could challenge these results in a number of ways, potentially overturning the predictions. Examples include improving the quality and breadth of genetic data, improved techniques for classifying genes into families, or a better phylogenetic tree. Any of these could, in principle, reverse the status of the predictions made about the biological data. However, given that the results are similar across the different databases, this does not seem a likely outcome.

If the dependency graph hypothesis is correct, the pattern observed here should continue. Other genetic data which show hierarchical patterns should also be found to fit a dependency graph better than a hierarchy. This includes lines of evidence such as pseudogenes, transposons, and synteny. Furthermore, the dependency graph should serve as the foundation for a comprehensive theory explaining all apparent evidence for common descent. It remains to be seen whether or not these predictions will be borne out.

An obvious objection is that we have not included any of the mechanisms thought to account for non-hierarchical data such as incomplete lineage sorting, gene flow, convergent evolution, or horizontal gene transfer. As such, it might be argued that any of the features of the data interpreted as evidence for the dependency graph may also be explained by these mechanisms. The focus of this paper has not been to critique common descent, but to the test the predictions of the dependency graph hypothesis. The challenge to common descent lies not in the comparison of the tree and dependency graph models but in explaining the successful predictions of the dependency graph hypothesis.

Furthermore, if the hypothesis of common descent can explain a hierarchy, lack of a hierarchy, and the appearance of a dependency graph, what sort of predictive power can the hypothesis possibly hold? It is not enough to merely explain the features of the data, rather they must be explained within the context of a testable hypothesis. Critiquing those rejecting common descent, White et al [15] wrote “It is essential that any person who does not accept the continuity of evolution puts forward alternative testable models.” This paper has answered the challenge, and paraphrasing their sentiment, it is essential that any person who does not accept the dependency graph puts forward alternative testable models. This means that it is not enough to simply postulate mechanisms to explain deviations from a hierarchy; one must postulate a hypothesis with predictions about the data.

It is possible that the defenders of common descent will devise a testable model that can explain the successful predictions of the dependency graph hypothesis. Perhaps some sort of extended synthesis of common descent with additional mechanisms could be that model. However, this study was designed to minimize the influence of mechanisms that deviate from the tree. The clade Metazoa was studied specifically because horizontal gene transfer is held to be rare amongst this clade. Utilizing gene families rather than sequences ignores small deviations from the tree easily explained by convergent evolution or incomplete lineage sorting. This leaves any extended model with limited options for mechanisms to include.

Furthermore, the dependency graph model might improve. The estimate of the performance of the dependency graph model depends on an inferred dependency graph. We have focused on keeping the algorithm used to infer the graph relatively simple. Further research will likely uncover improvements to the inference, finding a better graph and thus increasing the performance of the dependency graph model without modifying the model itself. It is worth noting that refinements to the common descent model might also be applicable to the dependency graph model.

A possible, but incorrect, objection would be that the dependency graph will always be better than the tree of life. Since the dependency graph starts with the hierarchy and then improves upon it, it may seem that it will always outperform the tree regardless of the true explanation for the data. It may seem that the analysis forces modules onto the data. However, this is not true as is demonstrated by the EvolSimulator datasets which did not fit the dependency graph model. Bayesian model selection penalizes models for postulating complexity such as is inherent in the dependency graph. This prevents modules being forced onto data where they do not fit. The dependency graph is better only if the improved fit to the data outweighs the complexity.

Our key contribution is describing a proposed pattern of life expected under common design. Common design is not restricted to finding anomalies which are difficult for common descent to explain. Many of these anomalies now fit into the overall pattern of life expected under the dependency graph. Researchers can work to understand these modules and the dependency links between them. Our hope is that this research is a large step forward in developing common design not as a critique of common descent, but as a research program which produces testable models and increases our understanding of the biological world.

ACKNOWLEDGEMENTS

The following individuals provided valuable support, encouragement, and feedback during the development of this paper: Paul Brown, Ken Cannon, Andrew Jones, Casey Luskin, George Montañez, Paul Nelson, Gordon Wilson, and others who wished to remain anonymous. Without their help and support this project would never have been brought to completion.

METHODS

JavaScript Applications

Each application was taken from a GitHub repository:

- dougjohnston/angular-drum-machine
- orizens/echoes-ng2
- hswolff/hn-ng2
- HospitalRun/hospitalrun-frontend
- yangmillstheory/mamba
- microapps/MoonMail-UI
- paulhoughton/mortgage
- shlomiassaf/ng2-chess
- benoitvallon/react-native-nw-react-calculator
- echenley/react-news
- afonsopacifer/react-pomodoro
- paulhoughton/remember
- cowbell/sharedrop
- Soundnode/soundnode-app
- andrewngu/sound-redux
- taigaio/taiga-front
- nicroto/viktor
- vim-awesome/vim-awesome
- ProtonMail/WebClient

The compiled form of each application was parsed using the acorn[1] parser. The acorn parser provided an abstract syntax tree for the compiled applications. An abstract syntax tree is like an annotation one might use to understand the genome. It records the relationship between different pieces of code in the compiled program as well as the meaning of each individual piece.

We extract the individual functions found in the program. These functions are taken as being the nearest equivalent to a gene. Characteristics of the function not directly related to the "shape" of the function are removed. For example, comments, the values of literals, variable names, and the contents of internal functions are removed. Only the types of the abstract syntax tree nodes and the identities of the operators are kept. The result is taken to be the "gene family" of the function. Any two functions which both have the same basic shape will thus be considered to belong to the same "gene family".

[1]https://github.com/ternjs/acorn

Probability of the Data

The following Bayesian analysis will assume various priors over nuisance parameters. They will, for the most part, be uniform priors over the relevant parameter space. In some cases, other priors might be argued to be more appropriate. Nevertheless, for the purposes of this first analysis, we have chosen simple straightforward uninformed priors that are analytically tractable.

Given a particular graph (additional nodes and their relationships), we need to compute the probability of the data under that graph. The term "node" refers to a species, ancestral species, module, or toolbox depending on the model. Each node not only has a number of other nodes it depends on, but also a number of gene families present in that node. For each node, including species, each gene is in one of four possible states:

- **Gained** - the gene family is present in the node but not any of its dependencies.
- **Present** - the gene family is present in the node and at least one of its dependencies.
- **Lost** - the gene family is present in at least one of the node's dependencies, but not the node itself.
- **Absent** - the gene family is not present either in the node or any of its dependencies.

We assume that every gene family only emerges in one node; it is too improbable for the same gene family to emerge twice. We expect that some species underwent more evolution than others, and some modules are larger than others. We model this by having each node have a gain parameter which indicates the probability that a gene will be gained there. Effectively, the module each gene is introduced in is chosen by throwing a weighted many-sided die. The sum of all gain parameters across all nodes must be one so that the total probability of a gene being gained in some module is one. The probability of gene j being gained in module i is thus α_i where α_i is the gain parameter for module i. This gives us a discrete probability for all the gains of:

$$\prod_{j}^{M} \alpha_{f(j)} = \prod_{i}^{N} \alpha_i^{g_i} \tag{3}$$

where N is the number of modules, M is the number of genes, $f(j)$ is the index of module where gene j is gained, and g_i is the number of gains in module i. However, the values of α_i are unknown nuisance parameters. We will take a flat Dirichlet prior, which assumes that all possible combinations of α_i are equally probable. Using the law of total probability to eliminate the α_i parameters, we obtain the probability:

$$\frac{\gamma(N)}{\gamma(N+G)}\prod_{i}^{N}\gamma(g_i+1) \tag{4}$$

where G is the number of gene families, and γ is the gamma function.

While a gene can only be gained once, it can be lost many times in many different nodes. Under common descent, a gene is lost during ordinary evolutionary processes. Under the dependency graph, a module may remove or replace the genes that would have been inherited from one of its dependencies. In either case, it is expected that some species or modules will lose more genes than others. Thus each node will have a loss parameter indicating the probability that a gene will be lost. Effectively, each gene inherited from a dependency or ancestor is subject to a weighted coin flip to determine whether or not it is kept. The discrete probability that the particular set of genes will be lost in a module is given by the binomial:

$$\lambda^l(1-\lambda)^p \tag{5}$$

where λ is the loss parameter, l is the number of losses, and p is the number of present genes. The present genes are those genes which could have been lost, but were not. The loss parameter is another unknown nuisance parameter, and we will adopt a uniform prior assuming any possible value as equally likely. Utilizing the law of total probability we can obtain:

$$\int_0^1 \lambda^l(1-\lambda)^p\,\mathrm{d}\lambda = \beta(l+1,p+1) \tag{6}$$

where β is the beta function. The probability density function for λ is 1 between 0 and 1, and thus does not explicitly appear in the formula.

This gives us the probability of a particular assignment of genes given a graph of the relationships between the nodes. We wish to know the probability of the data under any assignment, given a particular graph.

$$\Pr[D \mid M,G] = \sum_{a\in A(D)}\Pr[a|G] \tag{7}$$

where M is the model, D is the observed biological data, G is the given graph, $A(D)$ is the set of assignment of gene families to nodes which are consistent with the data D, and $\Pr[a|G]$ is the probability of assignment a given graph G. This summation is problematic because it is not feasible to sum over all possible assignments. Consequently we will use importance sampling to estimate the summation.

We can rewrite the summation as:

$$\Pr[D \mid M,G] = \sum_{i=1}^{20}\frac{1}{20}\sum_{a\in A(D)}\Pr[a|G]\frac{f_i(a)}{f_i(a)} \tag{8}$$

where $f_1, f_2, f_3, \ldots, f_{18}, f_{19}, f_{20}$ is a family of probability density functions over assignments. This can in turn be rewritten as:

$$\Pr[D \mid M,G] = \sum_{i=1}^{20}\frac{1}{20}E[\frac{\Pr[X_i|G]}{f(X_i,i)}] \tag{9}$$

where X_i is a discrete random variable distributed according to the probability density function f_i. For the f_is we start with the best assignment found during the search process. Gains are moved to an ancestral species with probability 2^{-i} and losses are moved to all descendent species with probability 2^{-i}. We can thus use Monte Carlo sampling to estimate each of the X_is to form an overall estimate of the total summation. The consequence is that we sample a variety of assignments both similar and different from the best known assignment to form an estimate of the overall summation.

This gives us an approximation to the probability of the data given a particular graph. In order to obtain the probability of the data under the model in general, we must use the law of total probability.

$$\Pr[D \mid M] = \sum_{g\in G}\Pr[g]\Pr[D|M,g] \tag{10}$$

Recall that the different models make different assumptions with regards to the graphs. In the case of the null model, there is only one possible graph, thus the probability of the data under that model is the same as the probability of the data under that particular graph.

For the case of common descent, any tree is a valid model. However, it is not feasible to sum over all possible trees. Furthermore, it is not clear what prior would be appropriate. To resolve these problems, we will use the following bound which follows from Equation 10 because the maximum of a weighted average over a set is always lesser than or equal to the maximum.

$$\Pr[D \mid M_D] \leq \max_{g\in G}\Pr[D|M_D,g] \tag{11}$$

where M_D is the model of common descent. This will bias the results in favor of common descent, as it assumes that the probability of the data under any tree is the same as the probability under the best tree. In practice, however, determining the best possible tree is difficult. We will instead use a tree thought to be near-optimal as an approximation.

In the case of the dependency graph model, there are many possible dependency graphs, and we need to assign a probability to each one. This probability is only used in the case of the dependency graph model. The largest number of modules that could conceivably be used to fit the data would be one for each combination of species except the empty set or $2^{|S|} - 1$ where $|S|$ is the number of species. We will assign an equal probability to every possible number of modules less than or equal to that limit.

Given the number of modules in a graph, we need to assign probabilities to the possible edges or dependencies both between modules and from species to modules. The maximum number of edges in a directed acyclical graph is given by

$$\frac{n(n-1)}{2} \tag{12}$$

where n is the number of nodes in the graph. Furthermore, given an arbitrary topological ordering there are $\frac{n(n-1)}{2}$ possible edges. Any directed acyclical graph is isomorphic to at least one graph with any given topological ordering, thus we can restrict consideration only to those with a particular arbitrary topological ordering. We take there to be a certain probability, b, of an edge being present. This allows us to express the probability of a particular combination of edges being present as

$$b^d(1-b)^{\frac{n(n-1)}{2}-d} \tag{13}$$

where d is the number of edges or dependencies in the graph. However, the value of b is unknown. We will adopt a uniform prior assuming it equally likely to be any value between zero and one. We can use the law of total probability to evaluate this:

$$\int_0^1 b^d(1-b)^{\frac{n(n-1)}{2}-d}\,\mathrm{d}b = \beta(d+1, \frac{n(n-1)}{2} - d+1) \tag{14}$$

where β is the beta function. The probability density function for b is 1 between 0 and 1, and thus does not explicitly appear in the formula. If we combine this with uniform probability over possible numbers of nodes, we obtain:

$$\frac{\beta(d+1, \frac{n(n-1)}{2} - d + 1)}{2^{|S|} - 1} \tag{15}$$

This is the probability of a particular dependency graph.

Returning to the law of total probability for the dependency graph model:

$$\Pr[D \mid M_G] = \sum_{g \in G} \Pr[D|M_G, g] \Pr[g|M_G] \tag{16}$$

where M_G is the dependency graph model. We cannot feasibly compute the probability over all possible graphs. Instead, we use the following bound:

$$\Pr[D \mid M_G] \geq \Pr[D|M_G, \hat{g}] \Pr[\hat{g}|M_G] \tag{17}$$

for any arbitrary $\hat{g}$. This is valid because the summation over non-negative numbers (such as probabilities) is always at least as great as any individual term. Critically, this result applies to any graph, no matter how that graph was obtained.

Inferring Graphs and Assignments

The basic common descent tree is taken from the NCBI hierarchy with some necessary tweaks. The NCBI tree contained many species not included in a given database. We pruned all branches of the tree leading solely to species not included in the particular database under study. Redundant nodes with only one child were collapsed after the pruning. In some cases, the NCBI tree is not bifurcating. For example, *Pan*, *Homo*, and *Gorilla* are all classified immediately under *Homininae* despite the fact that *Pan* and *Homo* are held to be more closely related. In any case where grouping two species from a non-bifurcating parent increased the probability of the data, the tree was modified to group those species together. In other cases, the reverse operation, ungrouping species held to share a common ancestor was done when it increased the probability of the data.

For the EvolSimulator dataset, we used the tree reflecting the true history from the simulation. For the JavaScript applications dataset, we used dynamic programming to find the optimal bifurcating tree. By optimal, we mean the tree which assigns the highest probability to the data. The same tweaking processes discussed above where nodes may be added or removed from the tree was applied in either case.

Algorithm 1 Algorithm for Adding a Module

```
while exist pairs of modules not yet eliminated do
    for all pairs of modules not yet eliminated do
        compute the move-candidate gene families for
                a module depended on by the pair of
                modules (see Algorithm 2)
    end for
    take the pair of modules with the largest set of
            move-candidate gene families
    add a new module depended on by the pair
    add all move-candidate gene families to module
    if probability of graph and assignment is higher
            than before the module is added then
        keep the module
    else
        revert to state before module was added
        eliminate module pair from future consideration
    end if
end while
```

In order to determine the location of the genes in common descent, we follow Dollo parsimony [56, 57]. The technique is based on the assumption that a gene family

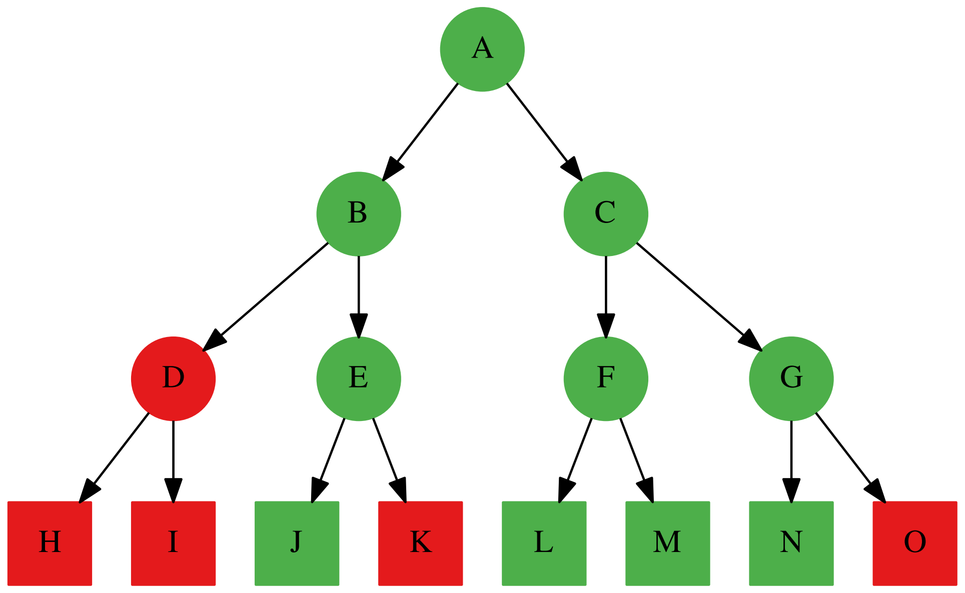

Figure 13: A tree with a depiction of the assignment of a gene family to nodes. Green nodes have the gene family, whereas the red nodes do not. The squares are extant species, but the circles are postulated ancestral species. doi:10.5048/BIO-C.2018.3.f13

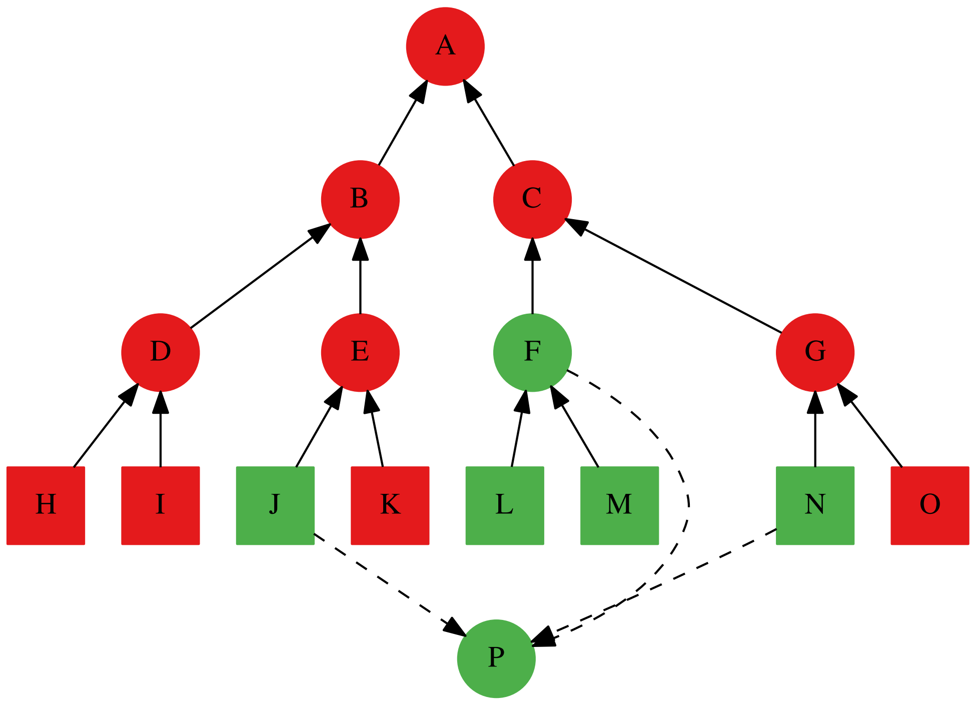

Figure 14: A dependency graph with a depiction of the assignment of a gene family to modules. doi:10.5048/BIO-C.2018.3.f14

Algorithm 2 Algorithm for determining move-candidate gene families for a new module

```
for all gene families do
    if gene family is only present in species which
            transitively depend on the new module then
        if total modules that depend on new module is
                less than total modules which depend on
                current module then
            gene family is a move-candidate gene family
        end if
    end if
end for
```

Algorithm 3 Algorithm for Removing Modules

```
for all modules do
    if module is not a species module then
        if module has dependencies then
            for all dependencies of module do
                move all gene family gains to dependency
                move all losses to all dependant modules
                for all dependants do
                    for all dependencies do
                        make dependant depend directly
                                on dependency
                    end for
                end for
                remove all dependencies on module
                remove module
                record resulting graph
            end for
            if any graph with the module removed has
                    a higher probability then
                take graph with the highest probability
            end if
        end if
    end if
end for
```

can only be gained once, and attempts to minimize the number of times it must be lost to explain the data. Figure 13 depicts the result for a single gene family. In a tree structure, there is a single species ancestral to all species which contain a gene family. The gene family cannot be gained in a module lower in the tree, because then the gene family would have to be gained in multiple nodes. In the case of the figure this is node A, the root. The gene family in question must be in the root, otherwise it would be impossible for it to be inherited by all the species which have it. Had species J not contained this gene family, it would have been gained in module C and modules A, B, and E would not have the gene family. In general, the gene family could be gained higher in the tree, but in most cases this would be a less probable assignment because it would imply the gene family was lost in more places. Thus, we initially place the gene family in that module.

Furthermore, a gene family is considered to be lost in the first ancestral node where none of the descendant species have the gene family. The gene family cannot have been lost in a higher node because that would require the gene family to be regained, and we assumed that each gene family is only gained once. In the case of the figure, the gene family is lost in D because neither H nor I contain the gene family. However, it cannot be lost in E because then J would be unable to inherit the gene family. The gene could have been lost in lower nodes, but in most cases this would be a less probable assignment because it would imply the gene was lost in more places. Thus we initially place the loss in these modules.

However, there are cases where placing the gain of a gene family higher or the losses lower will be an improvement. For example, if species H and I lost many gene families, it may make sense for the loss of the gene family in the figure to have taken place in both of the species nodes rather than the shared node. This happens when the higher or lower nodes have higher loss or gain parameters rendering individual gains or losses more probable in those nodes. We systematically test cases where we can push gains up or losses down that would increase the probability of the assignment. We repeatedly make the change producing the largest increase in probability until no such increases are possible. We take this assignment of genes to be approximately optimal.

The assignment of gene families to the null model is derived in the same way. It is identical to common descent where there is a single ancestral species which all extant species derive directly from.

The assignment of gene families to the dependency graph is more complicated. Our algorithm begins by converting the tree of common descent into a dependency graph. We follow the NCBI tree, pruned to remove any species not included in the dataset. The same heuristic for assigning gene families to nodes is used as in the common descent case. However, the process of moving gains up the tree or losses down the tree is not applied.

Starting with the common descent tree and gene family assignments may seem problematic. Recall, however, that the dependency graph hypothesis views the tree-of-life graph as approximately correct, but incomplete. The taxonomic categories discovered by biologists are the largest modules in the dependency graph of life. As such, the dependency graph should be expected to resemble the tree of life with various non-taxonomic modules added to it.

Furthermore, the purpose of the graph inferred by this process is for use in Equation 17 where it serves to compute a lower bound on the probability of the data under a randomly selected dependency graph. That equation is valid for any arbitrary graph, no matter how that graph was obtained. The dependency graph could

Table 10: Supplementary Files[*]

Description	File name	Size	doi link
All files in one large zip.	supplemental.zip	**91M**	**doi:**10.5048/BIO-C.2018.3.s1
Explains the file contents and format.	README.txt	1K	**doi:**10.5048/BIO-C.2018.3.s2
Script computes log-likelihood of graph / tree.	evaluate.py	2.5K	**doi:**10.5048/BIO-C.2018.3.s3
EggNOG database inferred dependency graph.	eggnog.graph.json	2.5M	**doi:**10.5048/BIO-C.2018.3.s4
EggNOG database adjusted ancestry tree.	eggnog.tree.json	3.6M	**doi:**10.5048/BIO-C.2018.3.s5
Ensembl database inferrred dependency graph.	ensembl.graph.json	1.2M	**doi:**10.5048/BIO-C.2018.3.s6
Ensembl database adjusted ancestry tree.	ensembl.tree.json	1.7M	**doi:**10.5048/BIO-C.2018.3.s7
Hogenom database inferred dependency graph.	hogenom.graph.json	2.7M	**doi:**10.5048/BIO-C.2018.3.s8
Hogenom database adjusted ancestry tree.	hogenom.tree.json	4.6M	**doi:**10.5048/BIO-C.2018.3.s9
HomoloGene database inferred dependency graph.	homologene.graph.json	387K	**doi:**10.5048/BIO-C.2018.3.s10
HomoloGene database adjusted ancestry tree.	homologene.tree.json	644K	**doi:**10.5048/BIO-C.2018.3.s11
OMA database inferred dependency graph.	oma.graph.json.zip	5.5M	**doi:**10.5048/BIO-C.2018.3.s12
OMA database adjusted ancestry tree..	oma.tree.json.zip	21M	**doi:**10.5048/BIO-C.2018.3.s13
OrthoDB database inferred dependency graph.	orthos.graph.json.zip	3.0M	**doi:**10.5048/BIO-C.2018.3.s14
OrthoDB database adjusted ancestry tree.	orthos.tree.json.zip	4.9M	**doi:**10.5048/BIO-C.2018.3.s15
Pfam database inferred dependency graph.	pfam.graph.json.zip	2.9M	**doi:**10.5048/BIO-C.2018.3.s16
Pfam database adjusted ancestry tree.	pfam.tree.json.zip	4.2M	**doi:**10.5048/BIO-C.2018.3.s17
TreeFam database inferred dependency graph.	treefam.graph.json	1.8M	**doi:**10.5048/BIO-C.2018.3.s18
TreeFam database adjusted ancestry tree.	treefam.tree.json	2.3M	**doi:**10.5048/BIO-C.2018.3.s19
UniRef-50 database inferred dependency graph.	uniref-50.graph.json.zip	15M	**doi:**10.5048/BIO-C.2018.3.s20
UniRef-50 database adjusted ancestry tree.	uniref-50.tree.json.zip	28M	**doi:**10.5048/BIO-C.2018.3.s21
JavaScript apps inferred dependency graph.	js.graph.json	1.3M	**doi:**10.5048/BIO-C.2018.3.s22
JavaScript apps inferred ancestry tree.	js.tree.json	2.0M	**doi:**10.5048/BIO-C.2018.3.s23
EvolSimulator dataset 1 inferred dependency graph.	evolsim1.graph.json	531K	**doi:**10.5048/BIO-C.2018.3.s24
EvolSimulator dataset 1 inferred ancestry tree.	evolsim1.tree.json	755K	**doi:**10.5048/BIO-C.2018.3.s25
EvolSimulator dataset 2 inferred dependency graph.	evolsim2.graph.json	386K	**doi:**10.5048/BIO-C.2018.3.s26
EvolSimulator dataset 2 inferred ancestry tree.	evolsim2.tree.json	565K	**doi:**10.5048/BIO-C.2018.3.s27
EvolSimulator dataset 3 inferred dependency graph.	evolsim3.graph.json	329K	**doi:**10.5048/BIO-C.2018.3.s28
EvolSimulator dataset 3 inferred ancestry tree.	evolsim3.tree.json	427K	**doi:**10.5048/BIO-C.2018.3.s29
EvolSimulator dataset 4 inferred dependency graph.	evolsim4.graph.json	502K	**doi:**10.5048/BIO-C.2018.3.s30
EvolSimulator dataset 4 inferred ancestry tree.	evolsim4.tree.json	688K	**doi:**10.5048/BIO-C.2018.3.s31
EvolSimulator dataset 5 inferred dependency graph.	evolsim5.graph.json	293K	**doi:**10.5048/BIO-C.2018.3.s32
EvolSimulator dataset 5 inferred ancestry tree.	evolsim5.tree.json	365K	**doi:**10.5048/BIO-C.2018.3.s33

[*]Distributed under the terms of the Creative Commons Atrribution License, as described on page 1.

have been obtained by reading tea leaves, and the logic would still be valid.

The process of inferring a module is described by Algorithm 1. The algorithm proceeds by identifying not yet existing modules that, when added, would increase the probability of the data taking into account both the probability of the graph and the probability of the assignment. For example, consider the module P in Figure 14. The same set of species have this gene family as in Figure 13. However, because the gene family is postulated to be introduced in the module P instead of module A, it avoids having to be lost multiple times.

The algorithm always postulates modules with exactly two dependent modules (this is a limitation of the algorithm, not the model). If the two modules already share a dependency, that dependency is moved to the newly created module. For example, if modules F and J were the dependents of a new module, the new module would depend on P, and F and J would no longer depend on P. The new module must have more species transitively dependent on it than either dependant by itself. For example, this rules out a module with dependants M and P. The new module would have the same dependent species as module P, and thus would be redundant.

When postulating a new module, we have to determine which gene families should be moved to that module. These are called the move-candidate gene families. The move-candidate gene families for a new module are determined by Algorithm 2. The algorithm determines which module to attempt adding by counting up the move-candidate gene families for all current candidate modules. Since gene families can only be gained once, any such gene family must be present only in species which would be dependent on the new module. For example, a module with dependants F and N cannot have the gene family depicted in Figure 14 because J would not be a dependent species. However, a module with dependants J and C could have the gene family added because all species with that gene family are dependants of either module J or C which would in turn be dependants of the new module. Additionally, moving the gene family to the new module must be an improvement over its current module. This is determined by whether the total number of modules dependent on the new module is less than the total number of modules dependent on the current module where the gene family is gained. For example, in Figure 13 the gene family is present in the root, and the root has 14 transitively dependent modules. The new module P has 6, which means that attributing this gene family to this new module would be an improvement, thus the gene family would be moved. However, if we started from Figure 14 where module P is already present and considered a new module with dependants J and C, the answer would be different. Module P has 5 dependent modules but the newly postulated module would have 8. Thus moving the gene family would not be an improvement and it will not be moved.

The algorithm identifies the potential module that would have the largest number of gene families added to it. It adds this module and evaluates whether or not the total probability, taking into account both the probability of the graph and the probability of the assignment, has increased. If it has, the module is kept and the algorithm continues. If it has not, the module is removed and rejected. The algorithm repeats this process, always adding the module which will have the largest number of gene families excluding those modules which have previously been rejected.

The next stage of the algorithm seeks to simplify the graph by removing modules as described in Algorithm 3. To remove a module, all its dependants must be made to depend directly on its dependencies. For example, if module D were to be removed, I and H would have to gain a dependency on B. If F were removed, L and M would have to gain a dependency on P and C. Any gene families gained in a module have to be moved to one of its dependencies. So, if F were being removed, any gene families gained in F have to be moved either to P or C. The algorithm considers moving all the gains to each dependency, and puts them in the one which gives the highest probability. If F were being removed, any gene families lost in F must now be lost in both L and M. Any losses in the removed module have to be pushed to all of its dependants.

The algorithm alternates between attempting to add modules and removing them until neither process is able to find a graph that improves the probability of the data.

1. Darwin C (1859) On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life
2. Eldredge N (1993) History, Function, and Evolutionary Biology. In Evolutionary Biology, 33–50. Springer US, Boston, MA. doi:10.1007/978-1-4615-2878-4_2
3. Doolittle WF (2009) The practice of classification and the theory of evolution, and what the demise of Charles Darwin's tree of life hypothesis means for both of them. Philosophical Transactions of the Royal Society B: Biological Sciences 364:2221–2228. doi:10.1098/rstb.2009.0032
4. Dávalos LM, Cirranello AL, Geisler JH, Simmons NB (2012) Understanding phylogenetic incongruence: lessons from phyllostomid bats. Biological Reviews 87:991–1024. doi:10.1111/j.1469-185X.2012.00240.x
5. Siler CD, Brown RM (2011) Evidence for repeated acquisition and loss of complex body-form characters in an insular clade of southeast Asian semi-fossorial skinks. Evolution 65:2641–2663. doi:10.1111/j.1558-5646.2011.01315.x
6. Aguilera F, McDougall C, Degnan BM (2017) Co-option and de novo gene evolution underlie molluscan shell diversity. Molecular Biology and Evolution msw294.

doi:10.1093/molbev/msw294

7. Crisp A, Boschetti C, Perry M, Tunnacliffe A, Micklem G (2015) Expression of multiple horizontally acquired genes is a hallmark of both vertebrate and invertebrate genomes. Genome Biology 16:50. doi:10.1186/s13059-015-0607-3
8. Huang W, Tsai L, Li Y, Hua N, Sun C, Wei C (2017) Widespread of horizontal gene transfer in the human genome. BMC Genomics 18:274. doi:10.1186/s12864-017-3649-y
9. Bapteste E, van Iersel L, Janke A, Kelchner S, Kelk S, McInerney JO, Morrison DA, Nakhleh L, Steel M, Stougie L, Whitfield J (2013) Networks: Expanding evolutionary thinking. Trends in Genetics 29:439–441. doi:10.1016/j.tig.2013.05.007
10. Theobald DL (2010) A formal test of the theory of universal common ancestry. Nature 465:219–222. doi:10.1038/nature09014
11. Salzberg SL (2001) Microbial Genes in the Human Genome: Lateral Transfer or Gene Loss? Science 292:1903–1906. doi:10.1126/science.1061036
12. Martin WF (2017) Too Much Eukaryote LGT. BioEssays 1700115:1700115. doi:10.1002/bies.201700115
13. Penny D, Foulds LR, Hendy MD (1982) Testing the theory of evolution by comparing phylogenetic trees constructed from five different protein sequences. Nature 297:197–200. doi:10.1038/297197a0
14. Baum DA, Ané C, Larget B, Solís-Lemus C, Ho LST, Boone P, Drummond CP, Bontrager M, Hunter SJ, Saucier W (2016) Statistical evidence for common ancestry: Application to primates. Evolution 70:1354–1363. doi:10.1111/evo.12934
15. White WTJ, Zhong B, Penny D (2013) Beyond Reasonable Doubt: Evolution from DNA Sequences. PLoS ONE 8:e69924. doi:10.1371/journal.pone.0069924
16. Luskin C (2017) Universal Common Descent: A Comprehensive Critique. In J Moreland, SC Meyer, C Shaw, AK Gauger, W Grudem, editors, Theistic Evolution: A Scientific, Philosophical, and Theological Critique, 363–401
17. Nelson P, Wells J (2003) Homology in Biology: Problem for Naturalistic Science and Prospect for Intelligent Design. In JA Campbell, SC Meyer, editors, Darwinism, Design and Public Education. Michigan State University Press
18. Bechly G, Meyer SC (2017) The Fossil Record and Universal Common Ancestry. In J Moreland, SC Meyer, C Shaw, AK Gauger, W Grudem, editors, Theistic Evolution: A Scientific, Philosophical, and Theological Critique, chapter 10, 333–361
19. Nelson P (2017) Five Questions Everyone Should Ask about Common Descent. In J Moreland, SC Meyer, C Shaw, AK Gauger, W Grudem, editors, Theistic Evolution: A Scientific, Philosophical, and Theological Critique, chapter 12, 403–430
20. Behe MJ (1996) Darwin's black box : the biochemical challenge to evolution. Free Press, New York
21. Denton M (2016) Evolution : still a theory in crisis. Discovery Institute Press, Seattle, WA
22. Gauger AK, Axe DD, Luskin C (2012) Science & Human Origins. Discovery Institute Press, Seattle, WA
23. Remine W (1993) The Biotic Message: Evolution Versus Message Theory. Saint Paul Science, 1st edition
24. Luskin C (2015) Cladistics to the Rescue? In Debating Darwin's Doubt: A Scientific Controversy that Can No Longer Be Denied, 115–124. Discovery Institute Press, Seattle, WA
25. Wise KP (1998) Is Life Singularly Nested Or Not? In 1998 International Conference on Creationism, 619–632
26. Parker J, Tsagkogeorga G, Cotton JA, Liu Y, Provero P, Stupka E, Rossiter SJ (2013) Genome-wide signatures of convergent evolution in echolocating mammals. Nature 502:228–231. doi:10.1038/nature12511
27. Foote AD, Liu Y, Thomas GWC, Vinař T, Alföldi J, Deng J, Dugan S, van Elk CE, Hunter ME, Joshi V, Khan Z, Kovar C, Lee SL, Lindblad-Toh K, Mancia A, Nielsen R, Qin X, Qu J, Raney BJ, Vijay N, Wolf JBW, Hahn MW, Muzny DM, Worley KC, Gilbert MTP, Gibbs RA (2015) Convergent evolution of the genomes of marine mammals. Nature Genetics 47:272–5. doi:10.1038/ng.3198
28. Chikina M, Robinson JD, Clark NL (2016) Hundreds of Genes Experienced Convergent Shifts in Selective Pressure in Marine Mammals. Molecular Biology and Evolution 33:2182–2192. doi:10.1093/molbev/msw112
29. Zou Z, Zhang J (2015) No Genome-Wide Protein Sequence Convergence for Echolocation. Molecular Biology and Evolution 32:1237–1241. doi:10.1093/molbev/msv014
30. Zhou X, Seim I, Gladyshev VN (2015) Convergent evolution of marine mammals is associated with distinct substitutions in common genes. Sci Rep 5:16550. doi:10.1038/srep16550
31. Ku C, Nelson-Sathi S, Roettger M, Sousa FL, Lockhart PJ, Bryant D, Hazkani-Covo E, McInerney JO, Landan G, Martin WF (2015) Endosymbiotic origin and differential loss of eukaryotic genes. Nature 524:427–432. doi:10.1038/nature14963
32. Bansal AK, Meyer TE (2002) Evolutionary Analysis by Whole-Genome Comparisons. Journal of Bacteriology 184:2260–2272. doi:10.1128/JB.184.8.2260-2272.2002
33. Dutilh BE, Huynen MA, Bruno WJ, Snel B (2004) The consistent phylogenetic signal in genome trees revealed by reducing the impact of noise. Journal of Molecular Evolution 58:527–539. doi:10.1007/s00239-003-2575-6
34. Lienau EK, Desalle R, Rosenfeld JA, Planet PJ (2006) Reciprocal Illumination in the Gene Content Tree of Life. Syst. Biol 55:441–453. doi:10.1080/10635150600697416
35. Qian J, Luscombe NM, Gerstein M (2001) Protein family and fold occurrence in genomes: power-law behaviour and evolutionary model. Journal of Molecular Biology 313:673–681. doi:10.1006/jmbi.2001.5079
36. Koonin EV, Wolf YI, Karev GP (2002) The structure of the protein universe and genome evolution. Nature 420:218–223. doi:10.1038/nature01256
37. Karev GP, Wolf YI, Rzhetsky AY, Berezovskaya FS, Koonin EV (2002) Birth and death of protein domains: a simple model of evolution explains power law behavior. BMC evolutionary biology 2:18. doi:10.1186/1471-2148-2-18
38. Karev GP, Wolf YI, Berezovskaya FS, Koonin EV (2004) Gene family evolution: an in-depth theoretical and simulation analysis of non-linear birth-death-innovation models. BMC evolutionary biology 4:32. doi:10.1186/1471-2148-4-32
39. Hahn MW (2005) Estimating the tempo and mode of gene family evolution from comparative genomic data. Genome Research 15:1153–1160. doi:10.1101/gr.3567505
40. Theobald DL (2011) On universal common ancestry, sequence similarity, and phylogenetic structure: the sins of P-values and the virtues of Bayesian evidence. Biology Direct 6:60. doi:10.1186/1745-6150-6-60

41. Gill J (2004) Grappling with Fisher's Legacy in Social Science Hypothesis Testing: Some Comments on Denis. Journal de la Société Française de Statistique 145:1–9
42. Murtaugh PA (2014) In defense of P values. Ecology 95:611–617. doi:10.1890/13-0590.1
43. Penel S, Arigon AM, Dufayard JF, Sertier AS, Daubin V, Duret L, Gouy M, Perrière G (2009) Databases of homologous gene families for comparative genomics. BMC bioinformatics 10 Suppl 6:S3. doi:10.1186/1471-2105-10-S6-S3
44. Townsend JT, Busemeyer JR, Vandekerckhove J, Matzke D, Wagenmakers EJ (2015) Model Comparison and the Principle of Parsimony. In The Oxford Handbook of Computational and Mathematical Psychology, 1–29. Oxford University Press. doi:10.1093/oxfordhb/9780199957996.013.14
45. Jeffreys H (1961) The Theory of Probability. Oxford University Press, third edition
46. Bateman A, Martin MJ, O'Donovan C, Magrane M, Apweiler R, Alpi E, Antunes R, Arganiska J, Bely B, Bingley M, Bonilla C, Britto R, Bursteinas B, Chavali G, Cibrian-Uhalte E, Da Silva A, De Giorgi M, Dogan T, Fazzini F, Gane P, Castro LG, Garmiri P, Hatton-Ellis E, Hieta R, Huntley R, Legge D, Liu W, Luo J, Macdougall A, Mutowo P, Nightingale A, Orchard S, Pichler K, Poggioli D, Pundir S, Pureza L, Qi G, Rosanoff S, Saidi R, Sawford T, Shypitsyna A, Turner E, Volynkin V, Wardell T, Watkins X, Zellner H, Cowley A, Figueira L, Li W, McWilliam H, Lopez R, Xenarios I, Bougueleret L, Bridge A, Poux S, Redaschi N, Aimo L, Argoud-Puy G, Auchincloss A, Axelsen K, Bansal P, Baratin D, Blatter MC, Boeckmann B, Bolleman J, Boutet E, Breuza L, Casal-Casas C, De Castro E, Coudert E, Cuche B, Doche M, Dornevil D, Duvaud S, Estreicher A, Famiglietti L, Feuermann M, Gasteiger E, Gehant S, Gerritsen V, Gos A, Gruaz-Gumowski N, Hinz U, Hulo C, Jungo F, Keller G, Lara V, Lemercier P, Lieberherr D, Lombardot T, Martin X, Masson P, Morgat A, Neto T, Nouspikel N, Paesano S, Pedruzzi I, Pilbout S, Pozzato M, Pruess M, Rivoire C, Roechert B, Schneider M, Sigrist C, Sonesson K, Staehli S, Stutz A, Sundaram S, Tognolli M, Verbregue L, Veuthey AL, Wu CH, Arighi CN, Arminski L, Chen C, Chen Y, Garavelli JS, Huang H, Laiho K, McGarvey P, Natale DA, Suzek BE, Vinayaka CR, Wang Q, Wang Y, Yeh LS, Yerramalla MS, Zhang J (2015) UniProt: A hub for protein information. Nucleic Acids Research 43:D204–D212. doi:10.1093/nar/gku989
47. Zdobnov EM, Tegenfeldt F, Kuznetsov D, Waterhouse RM, Simão FA, Ioannidis P, Seppey M, Loetscher A, Kriventseva EV (2017) OrthoDB v9.1: cataloging evolutionary and functional annotations for animal, fungal, plant, archaeal, bacterial and viral orthologs. Nucleic Acids Research 45:D744–D749. doi:10.1093/nar/gkw1119
48. Herrero J, Muffato M, Beal K, Fitzgerald S, Gordon L, Pignatelli M, Vilella AJ, Searle SMJ, Amode R, Brent S, Spooner W, Kulesha E, Yates A, Flicek P (2016) Ensembl comparative genomics resources. Database 2016:1–17. doi:10.1093/database/bav096
49. Ruan J, Li H, Chen Z, Coghlan A, Coin LJM, Guo Y, Heacute;riché JK, Hu Y, Kristiansen K, Li R, Liu T, Moses A, Qin J, Vang S, Vilella AJ, Ureta-Vidal A, Bolund L, Wang J, Durbin R (2008) TreeFam: 2008 Update. Nucleic Acids Research 36:735–740. doi:10.1093/nar/gkm1005
50. Huerta-Cepas J, Szklarczyk D, Forslund K, Cook H, Heller D, Walter MC, Rattei T, Mende DR, Sunagawa S, Kuhn M, Jensen LJ, Von Mering C, Bork P (2016) EGGNOG 4.5: A hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences. Nucleic Acids Research 44:D286–D293. doi:10.1093/nar/gkv1248
51. Finn RD, Coggill P, Eberhardt RY, Eddy SR, Mistry J, Mitchell AL, Potter SC, Punta M, Qureshi M, Sangrador-Vegas A, Salazar GA, Tate J, Bateman A (2016) The Pfam protein families database: Towards a more sustainable future. Nucleic Acids Research 44:D279–D285. doi:10.1093/nar/gkv1344
52. Altenhoff AM, Šunca N, Glover N, Train CM, Sueki A, Piližota I, Gori K, Tomiczek B, Müller S, Redestig H, Gonnet GH, Dessimoz C (2015) The OMA orthology database in 2015: Function predictions, better plant support, synteny view and other improvements. Nucleic Acids Research 43:D240–D249. doi:10.1093/nar/gku1158
53. Sayers EW, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, Dicuccio M, Edgar R, Federhen S, Feolo M, Geer LY, Helmberg W, Kapustin Y, Landsman D, Lipman DJ, Madden TL, Maglott DR, Miller V, Mizrachi I, Ostell J, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Shumway M, Sirotkin K, Souvorov A, Starchenko G, Tatusova TA, Wagner L, Yaschenko E, Ye J (2009) Database resources of the National Center for Biotechnology Information. Nucleic Acids Research 37:5–15. doi:10.1093/nar/gkn741
54. Beiko RG, Charlebois RL (2007) A simulation test bed for hypotheses of genome evolution. Bioinformatics 23:825–831. doi:10.1093/bioinformatics/btm024
55. Emms DM, Kelly S (2015) OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. Genome Biology 16:157. doi:10.1186/s13059-015-0721-2
56. Rogozin IB, Wolf YI, Babenko VN, Koonin EV (2006) Dollo parsimony and the reconstruction of genome evolution. In Parsimony, Phylogeny, and Genomics, volume 6, 190–200. Oxford University Press. doi:10.1093/acprof:oso/9780199297306.003.0011
57. Farris JS (1977) Phylogenetic Analysis Under Dollo's Law. Systematic Biology 26:77–88. doi:10.1093/sysbio/26.1.77