SUPPLEMENTARY MATERIAL

1. Data

Analysis is based on a data set of 32 wild type C57BL/6J fetal mouse kidneys of embryonic age E11.5 to E15.5 and an additional 28 mutant and control samples (all on an inbred C57BL/6J background, see Table S1).

Table S1. Mouse ureteric tree data used.

Group	Approximate Age (embryonic day)	Limb Stage	Samples	tip number (mean ± stdev)
Wild type time series	11.5	7	6	13.5±5.4
Wild type time series	12.5	8	5	25.0±2.9
Wild type time series	13.25	9	7	85.7±18.6
Wild type time series	13.75	10	5	116.2±15.8
Wild type time series	14.5	11	5	224.8±45.1
Wild type time series	15.5	12	4	621.3±35.8
Wild type time series	total		32	
Tgfb2+/- mutant	14.5	11	4	106.0±23.2
Bmp7+/+ control	14.5	11	12	247.9±71.4
Bmp7-/- mutant	14.5	11	9	58.3±19.0
Spry1-/- mutant	13.5	9/10	3	121.3±12.6
Mutant / control total			28	

2. Statistical notes

Statistical analysis was performed using the R programming language and RStudio environment. All t-tests were two-sided and unpaired, using the Welch adjustment for possibly unequal variance (using the t.test function in R with default parameters). Linear regression and ANOVA was performed with the lm function in R (using the f-statistic).

Age and size matched controls

Dedicated control groups were available for the *Bmp7* mutants, while the *Spry1* and *Tgfb2* mutants were compared to age matched data from the wild type time series data. For *Spry1*, the combined E13.25 and E13.75 wild type data was used.

For some analyses, size matched alternative controls (based on tip number) were also used, to test whether observed differences in the mutant groups could be explained as the result of a general delay in development. The E14.5 *Tgfb2* mutant was compared to the combined E13.25 and E13.75 wild type data, and the *Bmp7* mutant to the combined E12.5 and E13.25 wild type data. The *Spry1* mutant did not differ substantially in tip number from the age matched control group, so no alternative control was used.

Mutant comparisons

Mutant phenotypes were quantified by a comparison of kidney level metrics between mutant and control groups. The large variation between samples in some mutant groups did not appear consistent with a normal distribution, so a Wilcoxon rank sum test was performed using the R function wilcox.exact from package exactRankTests. This function produced the estimate and 95% confidence interval for the median difference, used in Fig. 6, Fig. 7 and Fig. S3.

Surface area was calculated for a convex hull on the set of tip points, using the R function convhulln from the geometry package, which interfaces the Qhull library (Barber, Dobkin et al. 1996).

For the number of tips per unit surface area and balance metrics, we are interested in difference compared to both age and size (tip number) matched wild type kidneys. In Fig. 6 and 7, the standard (age matched) controls were used; the alternative comparisons using size matched controls are shown in Fig. S3.

3. Tip state models

A tip state model is defined by a finite set of *tip states*, a specified initial state, and a transition rule which deterministically maps each state to a set of one or more new states. The transition rule is applied to produce a sequence of *rooted trees* (Epp 2004) in which each tip has an associated state from the allowed set; the first tree in the sequence consists of a single tip with the specified initial state. At each iteration, the rule is applied independently and simultaneously to each tip. If the current state of a tip is mapped to a single new state, then only the state is changed. If the tip state is mapped to more than one state, then child nodes are added to the tip with the assigned states; the original node is no longer a tip and does not have an associated state. The order of child nodes is not important, as we are ultimately concerned with distinct structures only up to graph isomorphism (Epp 2004). A state which is mapped to more than one new state is a *branching* state.

Tip state is an internal parameter which is used to generate the trees but is ultimately discarded, as it is unobservable in real trees; we say that two trees are isomorphic (written $\equiv$) if they are graph isomorphic, with tip state ignored if either or both trees possess it. Another potential source of ambiguity is that two models may produce the same tree structure at different positions in their generated sequences. Hence we define the family of rooted trees associated with a model to be the set of trees that occur in the generated sequence, with tip state ignored, which are distinct up to graph isomorphism. Tip state models which produce the same family of trees (up to isomorphism) are considered *equivalent*, and a model which is equivalent to a model with fewer states is *degenerate*, and excluded from the enumeration.

It is possible that a tip state model will produce only a finite number of trees. If any branching state occurs which gives rise after some number of iterations to at least one tip with the same (branching) state, then the tree sequence produced is infinite. If there are *k* states, there can be at most $k-1$ iterations of the model in which branching occurs before such a loop arises. Thus models which fail to give an infinite sequence of trees give at most *k* distinct trees. We call such models *trivial*, and as with degenerate models we exclude them from enumeration.

Classification of binary tip state models

Since we are modelling bifurcating tree structures, from here on we only consider binary models, in which each state is mapped to either one or two states; a *bifurcating* state is one which is mapped to two new states. In the following we develop some properties of these models, in particularly classifying all non-equivalent binary tip state models with up to three states, and proving the asymptotic balance property of a class of tip state models.

Notation

We denote a tree consisting of a single tip of state a by just writing a, relying on context to distinguish from the state a, and let (T_1, T_2) denote a tree with more than one tip, in which the two children of the root node are the roots of subtrees of type T_1 and T_2. Note that this notation can be used recursively to specify any rooted binary tree together with tip states. We also use this notation with no state specified to denote tree structures where the tips do not have names or states; so for example () is the trivial tree, and $((\,,), (\,,))$ is the symmetric tree with 4 tips. This is known as Newick notation (Krane 2003).

For a given tip state model, the transition rule is formally an automorphism on the set of rooted trees with labelled tips. When the model in question is clear we denote this associated function f, so $f^n(T)$ is the result of n consecutive transformations of an initial (tip labelled) tree T. We write $T_1 \mapsto T_2$ as a shorthand for $f(T_1) = T_2$.

For a tree T generated from a tip state model with k states, the number of tips in each state can be represented by a *state vector* $\boldsymbol{s}(T) \in \mathbb{Z}^k$. This requires an ordering of the states; we typically denote states with lowercase Latin letters and assume alphabetical ordering (of course reordering states will not fundamentally change the model). Since the result of transforming a tip is deterministic and depends only on its initial state, the state vector of $f(T)$ is a linear transformation of the state vector of T. That is:

Lemma 1: For any binary tree state model, there is a unique $k \times k$ matrix M such that $\boldsymbol{s}\big(f(T)\big) = M\boldsymbol{s}(T)$.

We refer to M as the state transition matrix. Another immediate consequence of the transition function acting separately on each function is:

Lemma 2: $f\big((T_1, T_2)\big) = (f(T_1), f(T_2))$

We now introduce the first non-trivial family of rooted binary trees.

Definition: The *n*th *perfect* tree, denoted $P_T(n)$, is the completely balanced binary tree formed by $n-1$ consecutive bifurcations of every tip, starting with a single node. Recursively, $P_T(1) = ()$ and $P_T(n+1) = (P_T(n), P_T(n))$. $P_T(n)$ has 2^{n-1} tips and 2^n-1 nodes in total.

One-state models

We begin the enumeration of binary tip state models by supposing there is only one possible tip state. In this case the only non-trivial model produces the perfect trees.

Observe that any model with no bifurcating state will produce only the trivial tree, while a model where all states bifurcate will give $f^n(a) = P_T(n)$. This immediately resolves the one-state case, and also gives the following lemma. Recall that a degenerate model is one that is equivalent to (produces the same trees as) a model with fewer tip states, and we disregard such models:

Lemma 3: Given any binary tip state model with more than one state, then if all states are bifurcating the model is degenerate, while if no states are bifurcating then it is trivial.

Two-state models

We show that there are two distinct models with two tip states (excluding trivial and degenerate models). One is the Fibonacci model (see Fig. 2A); the other produces trees consisting of a single "trunk" and a series of tips branching from it, which we call the singleton domain model. This "domain" branching is named for the pattern of trunk and offshoot branches seen in the lung (Metzger, Klein et al. 2008) but it is not capable of modelling ureteric tree structures.

By Lemma 3, we can assume there is one bifurcating and one non-branching state. Up to isomorphism we have cases

1. $a \mapsto a$ and $b \mapsto (a, a)$
2. $a \mapsto a$ and $b \mapsto (b, b)$
3. $a \mapsto a$ and $b \mapsto (a, b)$
4. $a \mapsto b$ and $b \mapsto (a, a)$
5. $a \mapsto b$ and $b \mapsto (b, b)$
6. $a \mapsto b$ and $b \mapsto (a, b)$

In theory each of these cases corresponds to two models, depending on the choice of start state. But if the start state is *a* then it will either be mapped to itself, giving a trivial model, or it will be mapped to *b*, producing the same sequence of trees with a delay of one time step as the model with start state *b*, and is thus equivalent to that model. So we need only consider start state *b*. In fact, we can assume for any model (not restricted to bifurcating models) that the start state is a branching state. Although in general it may take more than one iteration for an initial non- branching state to be mapped to a branching state, it will always be equivalent to the model starting at that branching state.

Lemma 4: A tip state model with a non-branching initial state is either trivial or is equivalent to a model which has a branching initial state and is otherwise the same.

Thus when enumerating distinct models, we assume that the start state is branching.

Returning to two-state models and the six cases above, case 1 is trivial, while cases 2, 4, and 5 are degenerate (in cases 2 and 5 $f^n(b) \equiv P_T(n)$, while in case 4 $f^n(b) \equiv P_T(\lfloor (n+2)/2 \rfloor)$). Case 3 produces a new family with trees of the form $(a, \big(a, (\ldots (a, (a, b)) \ldots)\big))$. This pattern of repeated offshoots from a trunk is seen in the branching of organs such as the lung, and we refer to it as *domain* branching. But our 2-state system cannot model further development of the offshoots. We call this minimal model *singleton* domain branching.

This leaves Case 6, which we call the Fibonacci model.

Definition: The *n*th *Fibonacci* tree, denoted $F_T(n)$, is defined recursively by $F_T(1) = F_T(2) = ()$ and $F_T(n+2) = (F_T(n), F_T(n+1))$.

Case 6 produces exactly the set of Fibonacci trees: $f^n(b) \equiv F_T(n+1)$. From the recursive definition of the Fibonacci numbers, the following follows immediately:

Lemma 5: $F_T(n)$ has $F(n)$ tips, where $F(n)$ is the *n*th Fibonacci number.

We have completed the 2 state classification:

Theorem 1: The only non-trivial and non-degenerate 2-state binary tip state models are the Fibonacci and singleton domain branching models.

Generalised delay models - single branching state

Before we examine three-state models in detail we first consider a class which can be regarded as a generalization of the Fibonacci model. The Fibonacci model has a single branching state, with one child tip that branches at the next model iteration and the other child branching with the delay of one additional model iteration. Similar models with additional states allow variable delays before bifurcation.

Suppose there is a single branching state c, and $f(c) = (a, b)$, where a, b, c are not necessarily distinct. Then we have 3 cases up to generality:

1. $f^p(a) = f^q(b) = c$, for some $p \geq q \geq 0$.
2. $f^q(b) = c$ for some $q \geq 0$ but there is no p such that $f^p(a) = c$
3. There is no p or q such that $f^p(a) = c$ or $f^q(b) = c$.

Case 2 will produce the singleton domain branching trees (branching will be delayed but the tree structures produced are the same), and hence will be degenerate for more than 2 states, while case 3 will produce only the trees () and $(,)$. We are left with case 1, which we define as the generalized (p, q) delay model.

Definition: For integers $p \geq q \geq 0$, the (p, q) delay model is a model with a single bifurcating state c, with $f(c) = (a, b)$ and $f^p(a) = f^q(b) = c$. The generalized delay tree $D_T^{p,q}(n)$ is the tree structure $f^n(c)$, produced by the (p, q) delay model.

Lemma 6: The (non-degenerate) (p, q) delay model will have $p + 1$ states $a, f(a), f^2(a), \ldots, f^p(a) = c$, with $b = f^{p-q}(a)$.
Proof: The listed states must exist and be distinct. Since this is sufficient to produce the delay trees, a model with more states is degenerate.

Note that $D_T^{1,0}(n) = F_T(n+1)$; the difference by one in the index is because the Fibonacci trees were defined to match the Fibonacci numbers (Lemma 5). The recursive definition of $F_T(n)$ generalises as follows:

Lemma 7: $D_T^{p,q}(m+p+1) = (D_T^{p,q}(m), D_T^{p,q}(m+p-q))$ for $m \geq 0$

Proof: From the definition and Lemma 2,

$$D_T^{p,q}(m+p+1) = f^{m+p+1}(c) = f^{m+p}\big((a,b)\big) = \big(f^{m+p}(a), f^{m+p}(b)\big) = \big(f^m(c), f^{m+p-q}(c)\big)$$

The result follows. □

The perfect trees can also be regarded as the base case of the generalized delay model: $D_T^{0,0}(n) = P_T(n)$. Thus we have categorised all models with a single branching state:

Theorem 2: Any non-trivial binary tip state model with exactly one branching state will produce either singleton domain trees or a family of generalized delay trees.

We now examine balance in the delay model.

Theorem 3: The asymptotic balance at the branching points of $D_T^{p,q}(n)$ as the weight tends to infinity is λ^{p-q}, where λ is the (unique) positive real solution of $\lambda^{p+1} - \lambda^{p-q} - 1 = 0$.

> Proof: We consider the root node of $D_T^{p,q}(n)$ as $n \to \infty$. By Lemma 7, the balance at this node is the ratio between the weight of two smaller $D_T^{p,q}$ trees, one of which is $(p-q)$ iterations advanced from the other. Consider the $(p+1) \times (p+1)$ transition matrix M for the (p,q) delay model. Assuming states are ordered $a, f(a), f^2(a), \dots, f^p(a)$, M has ones in position $(i+1, i)$, $i = 1,2, \dots, p$, plus positions $(1, p+1)$ and $(p-q, p+1)$ in the last column representing the bifurcating state. This matrix has characteristic equation $\lambda^{p+1} - \lambda^{p-q} - 1 = 0$. Since $p+1 > p-q \geq 0$, M will thus have exactly one positive real eigenvalue λ, which will be greater than 1. Therefore in the asymptote, the number of tips increases by a factor of λ in each model iteration (the corresponding eigenvector gives the asymptotic tip state proportions). Thus by Lemma 7, the asymptotic root balance is λ^{p-q}.

The subtree associated with each branch point is itself a $D_T^{p,q}$ trees, so this asymptotic property generalizes from the root to each branch point.

Since $F_T(n) = D_T^{1,0}(n)$, we have the following result:

Corollary 1: The Fibonacci trees have asymptotic balance $(\sqrt{5}+1)/2 \cong 1.61$

Three-state models

In this section we show that there are 34 distinct three-state bifurcating tip models, including one which is potentially useful for modelling ureteric trees, which we call the half-delay model. By Lemma 3, the non-degenerate 3 state models have exactly one or two bifurcating states, so we consider these two cases in turn.

We have shown that a single branching state implies that the only non-degenerate models are (p,q) delay models. By Lemma 6 we have $p = 2$, and hence $q = 0,1,2$. The $(2,2)$ delay model will give perfect trees so is degenerate. Hence we have two three-state models with a single branching state, the $(2,0)$ and $(2,1)$ delay models.

By Theorem 2, the $(2,1)$ delay model will have asymptotic balance $\cong 1.325$, while the $(2,0)$ delay model will have asymptotic balance $\cong 2.148$ (the square of the asymptotic growth rate $\cong 1.466$). Hence the $(2,1)$ delay model is potentially useful for modelling ureteric trees, but the $(2,0)$ delay model is not since it gives balance values outside the observed range.

We call the $(2,1)$ delay model the *half-delay*; see Fig. 2B.

In the remainder of this section we consider models with 2 branching states. Assume that states b and c branch, a does not. The cases are listed below up to generality, with each distinct, non-trivial and non-degenerate model noted in bold; we confirmed that these 32 models are distinct by generating all trees with up to 200 tips for each model, and comparing the tip numbers and in some cases tree structures until we were able to distinguish all models. Two main groups are domain structures, where one branching state is strictly downstream from the other, and nested structures, where each branching state will give rise to the other (possibly with delay). Note that by Lemma 4, in each case listed below we have two starting starts to consider, b and c; we use ss to refer to the initial state.

1. Both b and c branch symmetrically: this will give a perfect tree (note that all tips will have the same state at each step). Thus this case is degenerate.
2. Exactly one of b and c branch symmetrically. Without loss of generality assume it is c.
    a. $c \mapsto (c, c)$; we consider only initial state b since c gives perfect trees (degenerate).
        i. $b \mapsto (b, c)$: Degenerate, produces perfect trees.
        ii. $b \mapsto (a, b)$:
            1. $a \mapsto a$: Degenerate, produces singleton domain trees.
            2. $a \mapsto b$: Degenerate, produces Fibonacci trees.
            3. **$a \mapsto c$: Domain branching with a perfect tree at offshoots after single step delay; equivalent to a perfect tree with one pair of tips advanced beyond the rest.**
        iii. $b \mapsto (a, c)$:
            1. **$a \mapsto a$: Produces trees of the form $(a, P_T(n-1))$, with a single tip plus a perfect tree arising directly from the root node.**
            2. **$a \mapsto b$: Produces a family trees similar to domain branching except that the offshoots are more advanced than the trunk. We can write this family recursively as $T_1 = ()$, $T_2 = (,)$, $T_{n+2} = (T_n, P_T(n+1))$**
            3. **$a \mapsto c$: Produces trees of the form $(P_T(n), P_T(n+1))$; both sides are perfect trees, but one is one step more advanced.**
    b. $c \mapsto (b, b)$
        i. $b \mapsto (b, c)$: Degenerate, produces perfect trees.
        ii. $b \mapsto (a, b)$:
            1. **$a \mapsto a$: Gives trees consisting of two equal singleton domain trees descending from the root node, if ss=c.** Degenerate if ss=b (produces singleton domain trees).
            2. **$a \mapsto b$: Gives trees consisting of two equal Fibonacci trees descending from the root node, if ss=c.** Degenerate if ss=b (produces Fibonacci trees).
            3. **$a \mapsto c$: 2 Nested models**
        iii. $b \mapsto (a, c)$:
            1. **$a \mapsto a$: 2 distinct models: Resemble perfect trees with the addition of singleton offshoots off every node between each main branching generation; models vary on whether this occurs at odd or even generations.**
            2. **$a \mapsto b$: 2 distinct models: ss=b gives nested "3-cherries"; recursively, $T(n) = \big(T(n-2), \big(T(n-2), T(n-2)\big)\big)$; ss=c gives pairs of these trees descending from the root.**
            3. **$a \mapsto c$: 2 Nested models.**

c. $c \mapsto (a,a)$

i. $b \mapsto (b,c)$:

1. $\boldsymbol{a} \mapsto \boldsymbol{a}$: ss=b gives domain branching with $\boldsymbol{P}(\mathbf{2})$ domains; ss=c trivial

2. $\boldsymbol{a} \mapsto \boldsymbol{b}$: 2 Nested models.

3. $\boldsymbol{a} \mapsto \boldsymbol{c}$: ss=b gives domain branching with perfect domain subtrees; compared to 2a(ii)3 ($\boldsymbol{c} \mapsto (\boldsymbol{c},\boldsymbol{c})$, $\boldsymbol{b} \mapsto (\boldsymbol{a},\boldsymbol{b})$, $\boldsymbol{a} \mapsto \boldsymbol{c}$) there is extra delay step, so more asymmetric; ss=c degenerate, gives perfect trees.

ii. $b \mapsto (a,b)$:

1. $a \mapsto a$: Degenerate (domain branching).

2. $a \mapsto b$: Equivalent to 2b(ii)2: Fibonacci, or each half is Fibonacci.

3. $\boldsymbol{a} \mapsto \boldsymbol{c}$: ss=b gives domain model with “slow growing” perfect tree offshoots; ss=c degenerate, gives perfect trees.

iii. $b \mapsto (a,c)$:

1. $a \mapsto a$: Trivial, produces $\{(\ ),(,\ ),(,(\ ))\ \}$

2. $\boldsymbol{a} \mapsto \boldsymbol{b}$: 2 Nested models.

3. $\boldsymbol{a} \mapsto \boldsymbol{c}$: ss=b gives both perfect trees and trees of the form $(\boldsymbol{P_T}(\boldsymbol{n}), \boldsymbol{P_T}(\boldsymbol{n}+\mathbf{1}))$; ss=c degenerate, gives perfect trees only

3. Both b and c branch asymmetrically.

a. $b \mapsto (a,x)$ and $c \mapsto (a,y)$, for $x,y \in \{b,c\}$: Degenerate; gives singleton domain branching (states b and c are redundant).

b. $b \mapsto (w,x)$ and $c \mapsto (y,z)$, for $w,x,y,z \in \{b,c\}$: Degenerate; gives perfect trees.

c. $c \mapsto (b,c)$

i. $b \mapsto (a,b)$:

1. $\boldsymbol{a} \mapsto \boldsymbol{a}$: ss=c gives domain branching where every domain is singleton domain tree; ss=b degenerate (singleton domain trees).

2. $\boldsymbol{a} \mapsto \boldsymbol{b}$: ss=c gives domain branching where each domain is Fibonacci; ss=b degenerate (Fibonacci trees).

3. $\boldsymbol{a} \mapsto \boldsymbol{c}$: 2 Nested models.

ii. $b \mapsto (a,c)$:

1. $\boldsymbol{a} \mapsto \boldsymbol{a}$: 2 Nested structures with repeated singleton offshoots; for ss=b tip numbers equal Fibonacci, but structure is distinct.

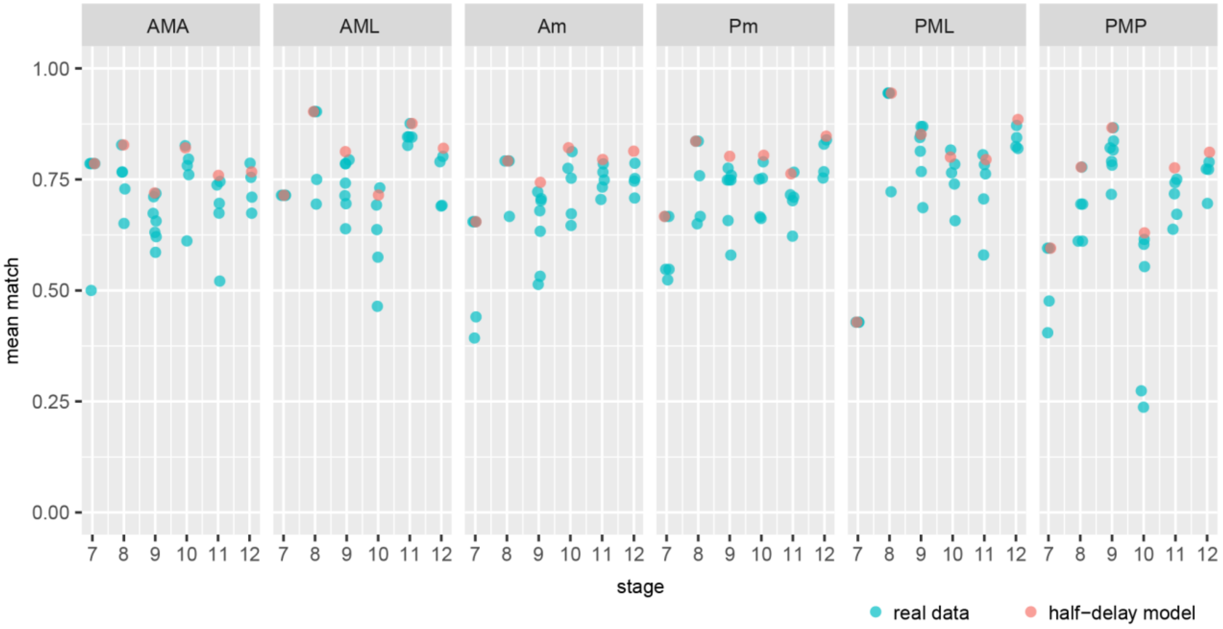
2. $\boldsymbol{a} \mapsto \boldsymbol{b}$: 2 Nested models.

3. $\boldsymbol{a} \mapsto \boldsymbol{c}$: 2 Nested models.

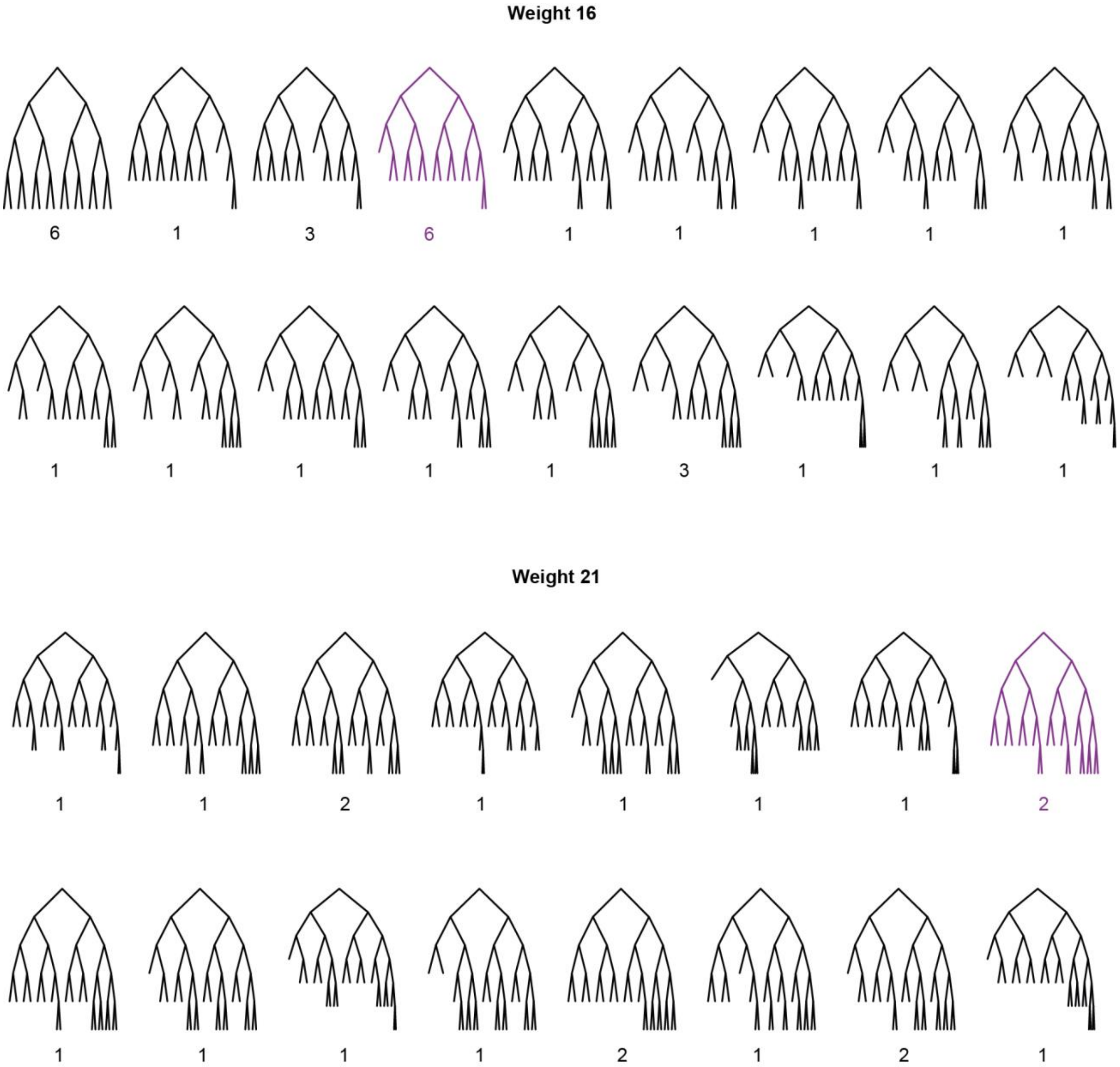
In total there are 32 distinct, non-degenerate 3 state models with 2 branching states. Of these, 20 are nested (these models are in pairs, with distinct models being produced depending on the initial state) and 7 are domain branching structures with infinite or finite (but non-trivial) offshoots. The remaining 5 trees are equivalent to 2 state models with additional (finite) structure around the root.

None of the models with two branching states had features that were observed in the ureteric tree data, although domain branching models may be of interest for modelling other structures such as the lung.

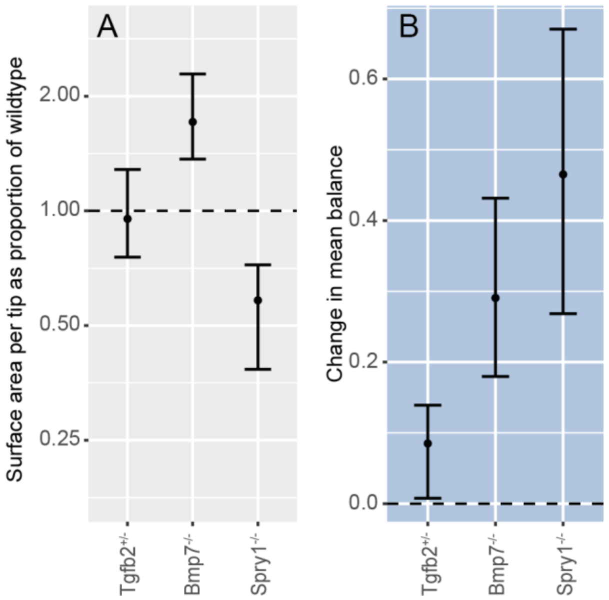
SUPPLEMENTARY FIGURES



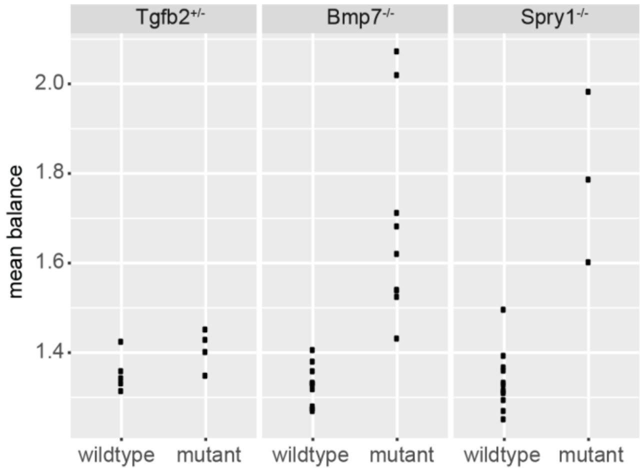
Supplementary Figure 1: Comparison of the best half delay model fit against variation between samples, within each stage and clade, showing that the model lies at the center of the natural variation. For each of the six developmental stages and six clade types, the corresponding set of clades was isolated from the wild type ureteric tree data. Clades which had not developed beyond a single tip were discarded (10 of 42 at stage 7 and 1 of 36 at stage 8). For each such set of clades, the half-delay model tree with the best average matching to the set was added (see Fig. 3A for definition of matching score). Then for each real or model tree, the average matching score with the other members of its set was calculated, and this metric was used to compare the model (red) against the real samples (blue). In 29 of 36 cases the half delay model has highest or equal highest average matching score, indicating that the model is closer to the sample clades on average than are any of the individual samples to the others; in the remaining 7 cases the model is surpassed by only small margins. A small amount of random horizontal permutation has been added to points in the plot to enable points with equal matching score to be distinguished.



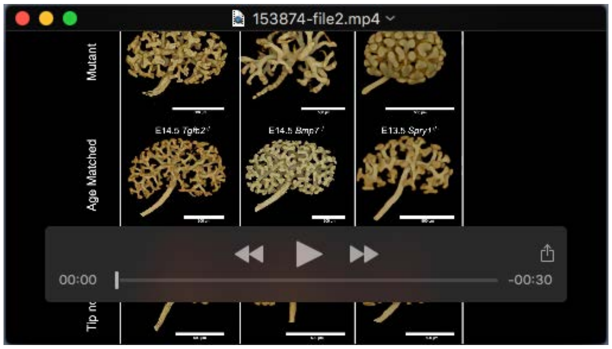
Supplementary Figure 2: Weight 16 and 21 subtree frequency in wild type time series dataset. In each case the highlighted tree is the centroid of the set (most representative structure, determined by maximum mean matching score with the other members of the set), and is also the half-delay tree of the given weight.



Supplementary Figure 3: Mutant phenotypes compared to tip-number matched wild type controls in terms of (A) surface area per tip; (B) balance, averaged over the binary branch points in each tree. Compare to Fig. 6F, Fig. 7A. Alternative control groups are to show that mutant differences are not an artefact of generally delayed development. Each plot shows estimated median change from wild type (dashed line) with 95% confidence interval, using a Wilcoxon rank sum test on the kidney level metrics. For A, values were first log transformed, and the estimates presented as proportional change. Control / test group sizes are: *Tgfb2*$^{+/-}$ 12/4; *Bmp7*$^{-/-}$ 12/9; *Spry1*$^{-/-}$ 12/3.



Supplementary Figure 4: **Mean balance of kidneys in mutant and control groups.** Control / test group sizes are: *Tgfb2*$^{+/-}$ (E14.5) 5/4; *Bmp7*$^{-/-}$ (E14.5) 12/9; *Spry1*$^{-/-}$ (E13.5) 12/3. Comparison is to same-age wildtype controls.



Movie 1: Ureteric trees from mutant *Tgfb2*$^{+/-}$ (A), *Bmp7*$^{-/-}$ (B) and *Spry1*$^{-/-}$ (C) embryos (top row) compared with wild type controls (middle row) and alternative wild type comparison kidneys matched by tip number (bottom row).

References

Barber, C. B., D. P. Dobkin and H. Huhdanpaa (1996). "The quickhull algorithm for convex hulls." ACM Trans. Math. Software **22**(4): 469--483.

Epp, S. S. (2004). Discrete Mathematics with Applications, Thomson Brooks/Cole.

Krane, D. E. (2003). Fundamental concepts of bioinformatics, Pearson Education India.

Metzger, R. J., O. D. Klein, G. R. Martin and M. A. Krasnow (2008). "The branching programme of mouse lung development." Nature **453**(7196): 745-750.