Statistical analysis of fruit fly wing vein topology

Surabhi Beriwal

A thesis presented for the degree of
B.S. Mathematics

Department of Mathematics
Duke University
Durham, North Carolina
Class of 2018
1 Project Overview

The fruit fly *Drosophila melanogaster* is a commonly used model organism for evolution given that the species showcases interesting behaviors and is easy to modify and rear. Among other things, the *Drosophila* wings are studied because their structure is tractable, consistent, and traceable developmentally. Along with Dr. Ezra Miller and Ashleigh Thomas, I studied evolutionary changes to *Drosophila melanogaster* wings using persistent homology. To borrow from [Mil15, p. 1179], the biological hypothesis posits that selecting for continuous wing deformation leads to higher rates of topological novelty. We are interested in understanding whether selection on a continuous trait can itself cause higher rates of variation of a (separate) discrete trait. We work joint with Dr. David Houle at Florida State University.

1.1 Long-term goals

Our long-term goal is to study the aforementioned biological hypothesis by statistically comparing topologically normal and abnormal wings. We want to measure how abnormal the topologically abnormal wings are, specifically their correlation with more extreme continuous deformations. Our current focus is a dataset in which the data structures have been generated by applying a model of a topologically normal wing [Hou03, p. 2].

1.2 Short-term goals

Toward the project’s long-term goal, I worked to clarify how the data structures are encoded as splines, how the piecewise polynomial functions that make up the splines behave, and how to compute linear approximations of fly win veins and error bounds to quantify the validity of the approximation.

The code for this project can be found on GitHub [Ber18].

2 Research Program

2.1 Biological background

The model organism for the project is the *Drosophila melanogaster*, and we are specifically interested in the development of the *Drosophila* wing. Given that the veins of the *Drosophila* wing are the most prominent feature [Bla07, p. 294], we focus our attention on abnormalities in venation.
Note that here veins refer to ectodermal tubes that "serve as structural supports for the wing and as vessels for trachea, nerves, hemolymphs and blood cells" [Bla07, p. 294]. To standardize terminology refer to the following:

In the nomenclature most commonly used in Drosophila studies (Figure 1), there are five main longitudinal veins (LVs) (L1–5) that run proximodistally; two smaller abbreviated veins (L0 and L6); and three crossveins (CVs), the anterior and posterior crossveins, which bridge L3–L4 and L4–L5, respectively (ACV and PCV), and the humeral crossvein (HCV), a Late third instar Margin which runs between the anterior wing margin and L0. [Bla07, p. 295]

For our purposes, the terms longitudinal veins and cross veins will suffice without further delineation.

Our focus is evolutionary selection. To reiterate, the biological hypothesis posits that selecting for continuous wing deformation leads to higher rates of topological novelty [Mil15, p. 1179]. Such topological novelty is shown in Figure 2; examples include but are not limited to extra veins (center) or veins that terminate prematurely (right).

Figure 1: Venation of Drosophila wing [Bla07]

For our purposes, the terms longitudinal veins and cross veins will suffice without further delineation.

Our focus is evolutionary selection. To reiterate, the biological hypothesis posits that selecting for continuous wing deformation leads to higher rates of topological novelty [Mil15, p. 1179]. Such topological novelty is shown in Figure 2; examples include but are not limited to extra veins (center) or veins that terminate prematurely (right).

Figure 2: Normal wing (left) and two abnormal wings with topological differences from the first

2.2 Mathematical background

We are studying the biological hypothesis using persistent homology. Given that topology is necessarily qualitative, topological data analysis techniques are employed to understand properties of a space given a discrete structure within it and the reverse [Wei11]. Persistent homology is one such technique.

2.2.1 One-parameter persistent homology

Based on Ghrist 2007, the following explains the theory of persistent homology using a point cloud of data [Ghr07, p. 2]. Complications arise when generalizing from one parameter to more than one parameter, but the theoretical development of this technique is outside the scope of this thesis [CSZ09] [CZ09] [Mil17].

Suppose we have points in Euclidean space $\mathbb{E}^n$. We can convert these points into a graph (or network) by using the points as vertices and connecting the pairs of vertices that are within distance $\varepsilon$ of one another. There are many methods to build simplicial complexes from these graphs. Here are two such methods: The first is the Čech Complex, which is the "abstract simplicial complex whose $k$-simplices are determined by unordered $(k + 1)$-tuples of points whose
closed $\varepsilon$-ball neighborhoods have a point of common intersection" [Ghr07, p. 3]. The second is the Rips complex, which is the "abstract simplicial complex whose $k$-simplices correspond to unordered $(k+1)$-tuples of points which are pairwise within distance $\varepsilon$" [Ghr07, p. 3]. These are displayed in Figure 3.

Analysis of such constructions of simplicial complexes requires a choice of parameter $\varepsilon$ in order to extract topological features from the data. However the question arises: can an optimal $\varepsilon$ be chosen [Ghr07, p. 4]? The theory of persistence was proposed as a response to this question because it describes how the topological features in question persist over scale, i.e. a set of parameter values.

**Lemma** [Ghr07, p. 6]. For any $\varepsilon > 0$, there is a chain of inclusion maps

$$\mathcal{R}_\varepsilon \hookrightarrow \mathcal{C}_{\sqrt{2}} \hookrightarrow \mathcal{R}_{\sqrt{2}}.$$  

This implies that any topological feature which persists under the inclusion $\mathcal{R}_\varepsilon \hookrightarrow \mathcal{R}_{\varepsilon'}$ is in fact a topological feature of the Čech complex $\mathcal{C}_{\varepsilon'}$ when $\varepsilon'/\varepsilon \geq \sqrt{2}$. **Moral:** the homology of the inclusion $\iota_* : H_\ast \mathcal{R}_\varepsilon \to H_\ast \mathcal{R}_{\varepsilon'}$ reveals information that is not visible from $H_\ast \mathcal{R}_\varepsilon$ and $H_\ast \mathcal{R}_{\varepsilon'}$ unadorned.

Figure 4 displays how the zeroth homology (the vector space spanned by all connected components) $H_0$ and first homology (the vector space spanned by all loops) $H_1$ change as $\varepsilon$ increases. Each bar in the barcode plot in Figure 4 begins when a feature (connected components in zeroth homology or loops in first homology) is born and ends when the feature dies. The number of bars present at each value of the parameter $\varepsilon$ gives the dimension of the vector space at that value.
2.2.2 Two-parameter persistent homology

In this project, we use two-parameter persistence. Given our interest in the topology of the wing, we can extract the features that interest us and quantify how they persist as our parameters change. In this case, our parameters are the distance $r$ from each vertex and the distance $s$ thickened from each edge, as shown in Figure 5. In the fly wing, the edges are the biological veins and the vertices are biologically meaningful points, often the intersection of two edges, or landmarks. On our fly wing data structures, we thicken the veins and shrink the vertices; our concern is to compute the connected components of the $s$-thickened veins with the $r$-discs at the vertices removed at each measure of semi-thickness ($s$) and radius ($r$). In Figure 5, that means the connected components of the set $X_{(r,s)}$ that is the union of the red regions with the union of the blue regions deleted.

Figures 6 and 7 are visual representations of $r$ and $s$ values and connected components that have been generated using a model and not the actual data. The distance from the edge set does not change, but two different radii are shown. In Figure 6, three connected components result from the removal of the vertex. In Figure 7, the removal of the vertex does not change the number of connected components.
Figure 7: semi-thickness \( s \) bigger than radius \((-r)\)

Figure 8 plots the number of connected components for each value of \((r, s)\). The red lines on the diagram function as veins and the blue disks, vertices. Upon thickening the red lines and shrinking the disks, we take the vector space whose basis is the set of connected components. In this particular diagram depicting the \((r, s)\)-plane, light blue refers to one connected component, green refers to two, and yellow refers to three.

Figure 8: Two sample persistence diagrams

Ultimately, this project aims to compute a rank function with input parameters \((r, s)\) and \((r', s')\). The injection of spaces \(X_{(r,s)} \hookrightarrow X_{(r',s')}\) induces a (linear) map on zeroth homologies, and the rank function records the ranks of those maps for all choices of \((r, s) < (r', s')\).

3 Research Project

3.1 Generating the splines

The data files are two-fold: fly wing images and labeled wing files. The former gives biological context. The latter contains control points generated through an automated process to extract features of interest [Hou03, p. 1]. These control points are used to generate Bezier curves, which are certain curves that pass through the first and last control points of a set and lie in the convex hull of the control points. Ultimately, the Bezier curves are the data structure that we take as input to our algorithms and statistical analysis given that polynomial curves are easy to compute and, perhaps more importantly, contain biological features pinpointed by biologists.

The labeled wing files were generated by the Houle lab using an automated analysis system called WINGMACHINE that finds vein intersections through low-level processing and generates control points by fitting an a priori b-spline wing model through high-level processing [Hou03, p. 2]. The latter part of the processing is done through a program called FINDWING that ultimately generates the control points for the b-spline [Hou03, p. 3].
Although the a priori b-spline model was generated using a *Drosophila affinis* model, Figure 11 shows that vein intersection points, or landmarks, are generally conserved between different wings of the species *affinis* and, as it turns out, *melanogaster* as well. This validates our use of this *Drosophila* wing model.

Figure 9: b-spline wing model optimized for *Drosophila affinis* wing: ends of splines (circles), landmarks (large filled circles), internal control points (squares), long veins (standard notation, denoted in Figure 1), and costa (C) [Hou03]

Figure 10: WINGMACHINE image processing of *Drosophila melanogaster* wing: (a) raw image (b) reverse raw image and minimize background (c) threshold and fill holes (d) reduce features to 1 pixel width (e) remove short features (f) use line intersections to fit model to image and produce final wing image with spline model overlaid [Hou03]

Figure 11: Mean locations of landmarks for 25 species in the sub-family Drosophilinae (black circles) and positions of landmarks for each individual specimen (blue dots) [Hou03]
Using the method described by Houle [Hou03], for each wing, we generate nine b-splines (low-order Bezier curves) using Matlab. Figure 12 displays a normal wing and its corresponding spline structure. Each of the nine splines is in a different color. The following code can be found cp2fun.m, the function (provided by David Houle) that generates b-spline output from the control points.

In cp2fun.m, the code assumes that certain landmarks or vein intersections will exist in the wing. Below, the array of landmarks (LM) is being populated using the control points (cp) in the data files.

% Mark landmarks (vein intersections)
LM = [cp{2}(1,:);...
    cp{3}(1,:);...
    cp{4}(1,:);...
    cp{5}(1,:);...
    cp{6}(1,:);...
    cp{6}(end,:);...
    cp{7}(1,:);...
    cp{7}(end,:);...
    cp{8}(1,:);...
    cp{8}(end,:);...
    cp{5}(end,:);...
    cp{9}(1,:)];

Below, the segment of code plots the b-splines. For each vein (cv), the code plots the splines in their relevant regions using the Matlab function fnplt.

% Graph splines
figure; % generate figure
hold on; axis equal;
for cv = 1:length(F)
    fnplt(F{cv});
    pp = linspace(min(fnbrk(F{cv},'interval')),max(fnbrk(F{cv},
        'interval')),size(cp{cv},1));
end
hold off;

Figure 12: wing (left), corresponding spline (right)
While the visual representations help confirm the splines are biologically accurate, we are more interested in the function output of the Matlab script, which includes information essential to our computations. The main Matlab package for processing b-splines is SPMAK; using this, Matlab encodes the same information in various ways for ease of manipulation. Two of these forms are b-splines and piecewise polynomial form (or ppform). The main differences between the two forms are the way the data is encoded and the way information is outputted. Through the SPMAK functions, we converted our b-splines to ppform; this was motivated by the ease of finding documentation on ppform functions and output online. This Matlab conversion preserves all properties of the b-splines. Tables 1 and 2 display function output that corresponds to spline 6 of the wing. Table 1 shows b-spline output and Table 2 converts the information to ppform. The splines are labeled in Figure 13.

![Labeled b-spline](image)

**Figure 13: Labeled b-spline**

<table>
<thead>
<tr>
<th>form</th>
<th>'B-'</th>
</tr>
</thead>
<tbody>
<tr>
<td>knots</td>
<td>[0.3818 0.3818 0.3818 0.7189 0.7189 0.7189]</td>
</tr>
<tr>
<td>coefs</td>
<td>[2×3 double]</td>
</tr>
<tr>
<td>number</td>
<td>3</td>
</tr>
<tr>
<td>order</td>
<td>3</td>
</tr>
<tr>
<td>dim</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Sample b-spline function.

The key takeaways from Table 1 are explained below.

- **Knots** encode the locations where the polynomial pieces of a spline connect by providing parameter values along parameterized curves. The multiplicity of each knot is one more than the smoothness of the corresponding curve at the connection point.

- **Coefficients** are the control points.

- **Number and dimension** are used for Matlab bookkeeping purposes, where dimension is the target dimension.

- An **order of 3** indicates the curves are quadratic. While Houle refers to the b-splines as cubic [Hou03, p. 3], we found the b-splines to be of order 3, or quadratic curves.

<table>
<thead>
<tr>
<th>form</th>
<th>'pp'</th>
</tr>
</thead>
<tbody>
<tr>
<td>breaks</td>
<td>[0.3818 0.7189]</td>
</tr>
<tr>
<td>coefs</td>
<td>[2×3 double]</td>
</tr>
<tr>
<td>pieces</td>
<td>1</td>
</tr>
<tr>
<td>order</td>
<td>3</td>
</tr>
<tr>
<td>dim</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2: Sample ppform function.

Table 2 is the output of the form of spline that is most easily workable: ppform.
Breaks indicate where the polynomial pieces connect. The values included in the break vector are the unique values from the knot vector.

- The number of pieces is the number of quadratic curves that constitute the entire spline in question.
- Number, dimension, and order have identical meaning to the output of the b-spline.

### 3.2 Approximating the wings

#### 3.2.1 Explicit formulas for splines

To move forward, we needed explicit formulas for the splines. To generate the splines, we relied on two Matlab functions (which are displayed below in the segment of code from cp2fun.m): aptknt which computes an appropriate sequence of knots and fnplt which outputs the splines based on the knot sequence. Neither of these functions made clear how points and splines were being computed.

```matlab
% Form B-splines
for cv = 1:length(cp)
    cp(cv)(:,2) = flipy*cp(cv)(:,2);
    cp(cv)(:,1) = flipx*cp(cv)(:,1);
    F{cv,1} = spmak(aptknt(sort(cp{cv}(:,1)),3),cp{cv}');
    F0{cv,1} = spmak(aptknt(sort(cp0{cv}(:,1)),3),cp0{cv}');
end
```

The following segment of code converts the b-splines to ppform using the Matlab function fn2fm:

```matlab
% Convert B-splines to ppform
for i = 1:9
    bspline = F{i};
    ppform = fn2fm(bspline,'pp');
end
```

We focused on the array of coefficients for each of the splines in ppform, which contains \( n \) rows and 3 columns. The value of \( n \) depends on the number of polynomial pieces of each b-spline: \( n = \text{pieces} \times 2 \). The array is structured such that two rows of the array describe one polynomial piece. More concretely, the quadratic curves are parameterized, using some parameter \( t \). In the array of coefficients, the first row for each piece is required for \( x(t) \) and the second for \( y(t) \). This allowed us to compute formulas for and plot all of the polynomial pieces. Figure 14 shows all of the quadratic curves associated with one fly wing data structure.
To plot only the necessary portions of the quadratic curves, we utilized the break arrays since breaks indicate where polynomial pieces connect. Using this, we were able to plot all of the quadratic curves only in their necessary regions, which we called break regions. This is shown in Figure 15.

Currently, we have yet to determine precisely how points along the curve are being sampled; the sampling method impacts the quality of the resulting linear approximation.

3.2.2 Linear approximation

We computed piecewise linear approximations of the wings. There were two main reasons for choosing this method. First, we knew that once we had equations for each of the quadratic curves of the b-splines, linear approximations would be efficient computations, which is important given our dataset is on the order of a million wings. Second, linear approximations would be easier to “thicken” for the persistence diagrams.

To create a linear approximation, we started with one spline comprised of a single quadratic curve. This allowed us to decipher how all of the output functioned to generate the quadratic curves.
To echo a question posed in the previous section, we want to understand how points are being sampled along the quadratic curves. Specifically, we would like points to be sampled more densely at points of higher curvature. Based on observation alone, this seems to be the case, but further exploration of the parameterization of the curves is required to quantify the extent to which that is true.

3.3 Computing distance

Once we computed linear approximations with \( n \) points for the wing, we had \( n - 1 \) line segments for each polynomial piece of each spline. Our final step involved computing the Hausdorff distance between the piecewise linear approximation and the actual spline structure for the wings.

**Definition.** The Hausdorff distance between two subsets \( A \) and \( B \) of a metric space is

\[
d_H(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(a, b) \right\}.
\]

**Theorem 1.** Suppose an arc of a quadratic function of \( x \) is non-negative with endpoints on the \( x \)-axis. The height of the local maximum is the Hausdorff distance between the arc and the line segment between the two endpoints of the arc.

**Proof.** First, we need to establish that the orthogonal projection is the closest point on the line segment to each quadratic curve.

Choose a point \( p \) along the curve and call its orthogonal projection \( \pi(p) \). Suppose the closest point on the line segment to the curve is \( q \). Based on the Pythagorean Theorem, we know:

\[
d(p, q)^2 + d(q, \pi(p))^2 = d(p, \pi(p))^2.
\]

Then we know that \( d(p, q)^2 \geq d(\pi(p), p)^2 \). Because distances are positive, we can take the positive square root to get \( d(p, q) \geq d(\pi(p), p) \). Since \( q \) was defined to be the closest point, \( d(\pi(p), p) \) cannot be less than \( d(p, q) \), so we know the two distances must be equal.

Consider the curve as a function from \( x \) to \( y \). Maximizing the \( y \) value is equivalent to computing the distance between furthest point along the curve from the line and the line segment. To do this,
we find the local maximum of the curve between the endpoints of the line segment, which occurs where the derivative of the curve equals 0. Because the curves are quadratic, we know that each has only one local maximum and any other point along the curve in our region of interest would have to be closer to the line segment.

**Remark.** The theorem is true for any positive function that attains its maximum on the interval. In our case, all the curves are quadratic curves.

**Corollary 2.** Consider an arc in the plane and the line through the endpoints of the arc. Assume that the orthogonal projection of the arc onto that line lies between the two endpoints. If the arc is from a quadratic curve, there is a point at which the tangent line to the arc and the line segment between the endpoints of the arc are parallel. The distance between this point and its orthogonal projection onto the line segment is the Hausdorff distance between the arc and the line segment.

**Proof.** The arc and the line segment are in $\mathbb{R}^2$. We can rotate them so that the line segment is on the $x$-axis and the arc is above the $x$-axis. Rotation is a rigid transformation so it preserves pairwise distances, and in particular, preserves the Hausdorff distance.

**Remark.** The projection onto the line segment hypothesis of the Corollary needs to be confirmed for fly wings. We need to verify that the points are being sampled such that the hypothesis is true. If not, the algorithm needs to check for and account for the case in which the hypothesis is not satisfied by further subdividing the arc by choosing more points for the linear approximation.

The pseudo code for the final step of the distance computation is shown below.

```
% Compute distance between line segment and polynomial curve

startpt = beginning of line segment
endpt = end of line segment
point = point on polynomial curve with derivative equal
to slope of line segment

vectorA = startpt - endpt
vectorB = point - endpt
distance = norm(cross(vectorA, vectorB)) / norm(vectorA)
```

Eventually, taking the Bezier curves as the “ground truth” representation of the biology, computing distance between the curves and the linear approximation yields error bounds for our computations.

## 4 Conclusion

The overarching biological hypothesis for this project concerns topological novelty arising when directional selection pushes continuous variation in a developmental program beyond a certain threshold. Understanding this biological hypothesis will allow us to understand whether selection for continuous traits can affect the rates of variation in discrete ones.

Over the course of this project, I worked to understand the data structures and how to manipulate the given data producing two pieces of a larger data-analysis pipeline: a piecewise linear approximation of the data and a distance computation between the linear approximation and the polynomial curves. As mentioned earlier, our eventual goal is to use two-parameter persistent homology to compute a rank function that takes as input two pairs $(r, s)$ and $(r', s')$ of parameters referring to the radius assigned to vein intersections and the thickness assigned to veins. This will allow us to statistically compare wings toward greater understanding of our biological hypothesis.

## 5 Acknowledgments

I would like to acknowledge and thank Dr. Ezra Miller and Ashleigh Thomas for their mentorship on this project.
References


