Summer 2021

Nonlinear variability in human movement analysis

Scott Thatcher

Nonlinear variability in human movement analysis

by

Scott Thatcher

A Creative Component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Statistics

Program of Study Committee:
Daniel S. Nettleton, Major Professor
Dan Nordman
Stephen Vardeman

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation/thesis. The Graduate College will ensure this dissertation/thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2021

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS

Thanks to Dr. Micheal Bird and his research teams, including Katelyn Campbell, Emily Harl, Ben Seedorf, and Austin Watson, for sharing their data with me and allowing it to be used in this creative component.

Thanks also to my major professor, Dr. Daniel Nettleton, for his patience and help in preparing this paper, and to my committee, Dr. Dan Nordman and Dr. Stephen Vardeman.

ABSTRACT

This paper provides an introduction to the calculation of the Lyapunov exponent as a measure of nonlinear variability in time series data, with application to human movement data. The Lyapunov exponent is defined mathematically, and a survey of the theoretical results that underpin its computation is given. Computational algorithms for reconstructing the state space of the movement process and calculating the Lyapunov exponent are then described and translated into the R programming language when necessary. These algorithms are then applied to data on cambered running provided by Dr. Michael Bird at Truman State University. Preliminary results indicate that the Lyapunov exponent may be able to distinguish between flat-surface running and running on a camber, but uncertainties remain concerning the degree to which parameters of the algorithms must be set "by hand" for each subject and each time series.

CHAPTER 1. Introduction

1.1 Dynamical Systems and State Space

The state of a system that is described by a certain set of dynamical equations can be expressed as a point in "state space," and the evolution of that system over time can be expressed as a trajectory through state space.

Consider the simple idealized example of a mass moving without friction on a one-dimensional track under the influence of a spring. The dynamical equation describing its simple harmonic motion is

$$\frac{d^2x}{dt^2} = -kx, \tag{1.1}$$

whose solution is of the form

$$x(t) = x_0 \cos(\omega t) + \frac{v_0}{\omega} \sin(\omega t) \tag{1.2}$$

$$v(t) = -\omega x_0 \sin(\omega t) + v_0 \cos(\omega t) \tag{1.3}$$

where ω is the angular frequency of the oscillation, which is related to the constant k and the object's mass, and x_0 and v_0 represent the position and velocity of the mass at time $t = 0$. In other words, the complete state of the system can be specified by giving a point in the (x, v) plane, and the motion of the spring can be described as a trajectory through that plane. (See Figure 1.1.)

1.2 The Lyapunov Exponent

Dynamical systems that display chaotic properties often display sensitive dependence on initial conditions. In other words, small changes to the initial state of the system can have large long-term consequences on future states of the system. Given that measurements of initial states will always contain a certain amount of measurement error, sensitive dependence on initial conditions implies a long-term unpredictability, even in systems that are governed by relatively simple dynamical equations.

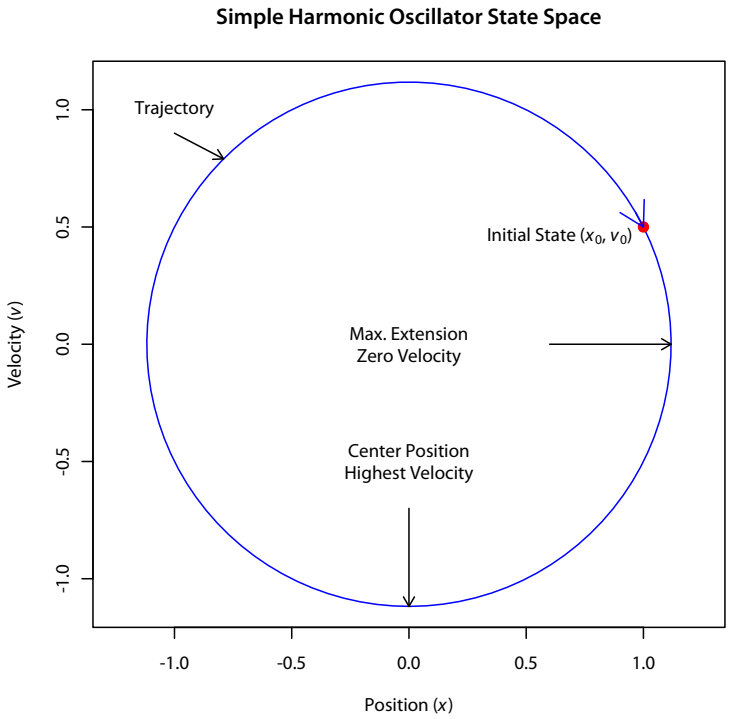

Figure 1.1 Trajectory through state space of a simple harmonic oscillator.

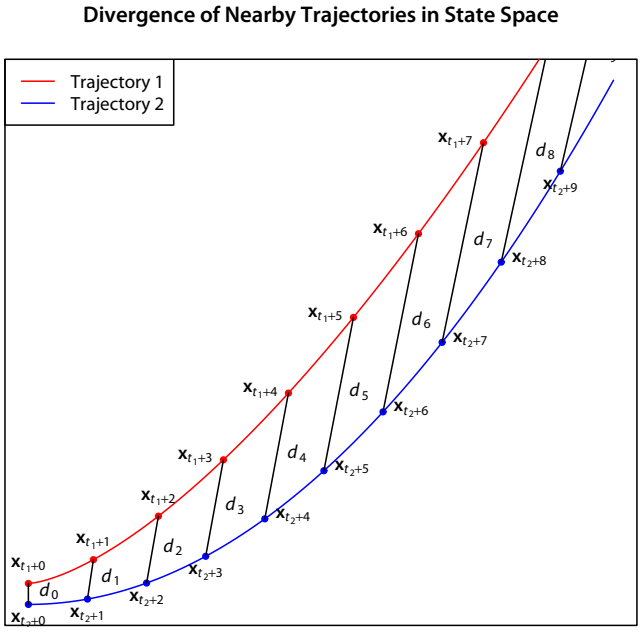

Figure 1.2 The divergence d_i of neighboring trajectories can grow over time.

The "largest Lyapunov exponent" is one measure of a system's dependence on initial conditions that measures, in some sense, how quickly two very similar states will diverge as a system evolves over time. For example, if $\mathbf{x}_{t_1}$ represents one point on the system's trajectory, and $\mathbf{x}_{t_2}$ represents another point on the trajectory that's nearby in the state space, but separated in time from the first point, we could monitor the distance $d_i = \|\mathbf{x}_{t_1+i} - \mathbf{x}_{t_2+i}\|$ between both points as they evolve in time. (See Figure 1.2.)

In a chaotic system with sensitive dependence on initial conditions, that divergence will be exponential, while in a non-chaotic system, the divergence will be much slower [8, pp. 58–67].

When analyzing one-dimensional time series data from a real-world source, information about the system's true embedding in, and trajectory through, state space is generally not known. It is possible, however, to reconstruct an approximate state space using lagged versions of the original time series variable. Suppose the time series is represented by the sequence $\{x_i\}_{i=1}^{n}$. Then the embedding would

have the form

$$\mathbf{x}_i = \left(x_i, x_{i+\tau}, x_{i+2\tau}, x_{i+3\tau}, \ldots, x_{i+(m-1)\tau}\right),$$

where m is the embedding dimension, and τ is the lag.

The embedding dimension and lag are chosen algorithmically (see Sections 2.2.2 and 2.2.3) to optimize certain characteristics of the reconstruction [8, pp. 34–35]. Assuming all this is done, an empirical maximal Lyapunov exponent can be calculated for the time series.

In the analysis of movement data, the maximal Lyapunov exponent is often taken as a measure of dynamic stability, with higher values interpreted as indicative of decreased dynamic stability. It is argued that a larger Lyapunov exponent indicates that the system less readily recovers from perturbations back to a stable state. For example, see Look, et al. [11].

1.3 A Study of Cambered Running

Dr. Michael Bird, Professor of Exercise Science at Truman State University, leads several student research teams whose broad research interests include the study of athletes' movement and its relationship to injury risk. One of his teams is currently studying the effect of ground surface camber on dynamic stability while running. They collected motion capture data from 31 experienced female runners who ran on a treadmill under three conditions: flat, low camber, and high camber (see Figure 1.3). Several measures were derived from this motion capture data, including position of the left and right anterior superior iliac spine (ASIS), left and right hip angles, left and right knee angles, and pelvis angles.

One primary question of the current study is whether camber is related to changes in the runner's dynamic stability, which is theorized to lead to increased chance of physical injury.

Initial analysis by Dr. Bird and his team [2] was conducted using standard statistical measures. For example, measurements of pelvis, hip and knee angles were made at fixed "landmarks" within the time series, such as foot strike and midstance. These measurements were then compared between left and right sides and between different camber conditions using a two-way ANOVA.

Figure 1.3 Runner on a cambered surface.

It was then decided to expand the study through the use of non-linear measures of variability, such as the Lyapunov exponent. The Center for Research in Human Movement Variability at the University of Nebraska Ohama (MOVCENTR) [15] was contracted to provide such an analysis of Dr. Bird's data. I became involved in the project when Dr. Bird asked me to help in the interpretation of that report.

CHAPTER 2. Theory and Algorithms

2.1 Stationary Time Series

Throughout the discussion that follows, it will be assumed that the observed time series $\{x_i\}$ is a sample from a *stationary* time series.

From a mathematical perspective, we can model the stationary time series as a sequence $\{X_i, i = 0, i = \pm 1, i = \pm 2, \dots\}$ of random variables such that the joint distribution of $(X_1, \dots, X_n)$ is the same as the joint distribution of $(X_{1+h}, \dots, X_{n+h})$ for any specified n and h. In other words, the statistical properites of the time series are invariant under any time shift [1].

One consequence of the stationarity assumption is the assumption that all X_i are identically distributed, along with all pairs (X_i, X_{i+h}). This assumption implies that the measured data points $\{x_i\}$ can be used to estimate marginal and joint densities for X_i and pairs (X_i, X_{i+h}).

Time series data that displays a cyclic pattern can be considered as a realization of a stationary process if that process incorporates a random phase term. For example, the process $X_t = A\cos(t + \varphi)$ is stationary if A and φ are considered to be uncorrelated random variables. A realization of the time series will show cyclic behavior, but the underlying process itself, in its full generality, is stationary.

Holger Kantz, in *Nonlinear Time Series Analysis* [8, pp. 13–15], interprets stationarity in terms of the underlying physical process and imposses additional requirements on the observed data as well, under what is perhaps an expanded definition of "stationarity." From a physical perspective, for the time series to be considered stationary, the parameters that define the dynamics of the system must remain constant throughout the measurement process. Moreover, from a practical standpoint, it is also necessary that measurements be taken over a time period long enough that the full state space trajectory of the system is explored. Measurement over a time period that is too short may produce a time series that appears to show a non-stationary trend or other time-dependent behavior, and such a time series would at best produce misleading results in the analysis of nonlinear characteristics of the process.

If the measured time period is too short, the time series may also show intermitant behavior that deviates from its "usual" pattern, while a longer measurement period might place that intermitant behavior in the context of a recurring pattern.

Finally, Kantz [8, p. 15] points out that although standard time series analysis is often content with time series that are *weakly stationary* (possessing time-invariant mean and covariance functions), non-linear time series analysis often requires the stronger assumption of stationarity.

2.2 State Space Reconstruction

2.2.1 Mathematical Underpinnings

In the example of a simple harmonic oscillator shown in Figure 1.1, the axes in state space represented one-dimensional position and velocity—two quantities that, if known, completely describe the dynamic state of the system. One could imagine that an appropriate motion-capture device could capture position and velocity in a similar physical system, thus allowing the real system's trajectory in state space to be plotted. Often, however, this complete information about every variable is not available. Instead, the available information is represented by a scalar-valued time series, with measurements taken at discrete intervals. For example, the position of the spring, measured at discrete time points.

In the case where only a scalar-valued time series is available, it is possible to reconstruct a representation of the system's trajectory through state space by using lagged versions of the time series variable. Intuitively, this makes some sense. Approximations of the velocity can be reconstructed using knowledge of the system's past position—thus time lag information should be able to substitute for velocity information. Again, intuitively, if velocity is approximated by $v_t = \frac{x_{t+\tau} - x_t}{\tau}$, then time-lagged values of x denoted by $(x_t, x_{t+\tau})$ contain similar information to the position/velocity pair (x_t, v_t). But does this work in practice?

As an example, consider a simple harmonic oscillator described by the differential equation and initial conditions

$$\frac{d^2x}{dx^2} = -x, x_0 = 1, v_0 = 0, \tag{2.1}$$

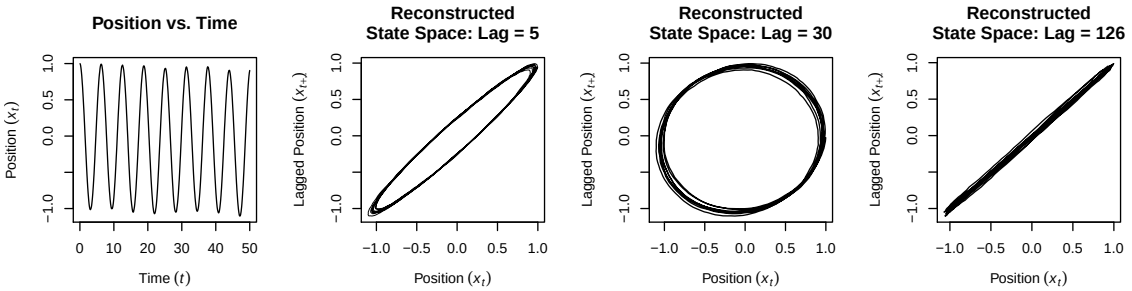

Figure 2.1 Position vs. time and reconstructed state space for a simple harmonic oscillator with random perturbations in velocity.

with theoretical solution $x(t) = \cos(t)$. Assume the system is subject to random perturbations in its velocity, and its position is sampled 1000 times from time $t = 0$ to time $t = 50$, yielding a time step of $\frac{1}{20}$ between measurements.

Figure 2.1 shows examples of attempted state-space reconstructions using various lags. Note that a lag of $\tau = 30$ gives a reconstruction that resembles the original state space, but a lag near $\tau = 126$ results in a degenerate reconstruction because $126(0.05) \approx 2\pi$, and the system's oscillation has period 2π.

The mathematical underpinnings of delay-coordinate embedding lie in embedding theorems proven by Whitney, Takens and Sauer et al., to name a few. Whitney's classic embedding theorem states that every m-dimensional smooth manifold can be embedded in a Euclidean space of high enough dimension, with $\mathbb{R}^{2m+1}$ sufficing. Although Whitney's theorem is not directly related to delay-coordinate embedding, it does illustrate a pattern that will appear in the results that are related to the case at hand.

Takens' embedding theorem [17] considers the case of an m-dimensional manifold $\mathcal{M}$, a smooth diffeomorphism $\varphi : \mathcal{M} \to \mathcal{M}$ representing the discrete evolution of the system from time t to time $t + \tau$, and a smooth map $y : \mathcal{M} \to \mathbb{R}$ representing the scalar-valued measurement of the state of the system. If φ^i represents the composition of φ with itself i times, then Takens' embedding theorem states that the map $\Phi_{\varphi,y} : \mathcal{M} \to \mathbb{R}^{2m+1}$ defined by

$$\Phi_{\varphi,y}(x) = \left(y(x), y(\varphi(x)), \ldots, y(\varphi^{2m}(x))\right) \tag{2.2}$$

is, generically, an embedding. In other words, there is an open and dense set of pairs (φ, y) for which $\Phi_{\varphi,y}$ is an embedding.

Takens' result considered the dimension m of the entire state space, which can often be quite large. Thus, a $(2m+1)$-dimensional Euclidean space would be an even higher-dimensional space. Sauer, Yorke and Casdagli [14] refined Takens' theorem, replacing the dimension m of the state space itself with the fractal dimension D_F of the system's attractor. They proved that almost every delay map into $\mathbb{R}^n$ with $n > 2D_F + 1$ is an embedding. Because the fractal dimension of the dynamical system's attractor is often much lower than the dimension of the state space itself, Sauer's theorem provides hope that a delay-coordinate embedding of relatively small dimension might suffice to measure properties such as the Lyapunov exponent.

Kantz [8, pp. 125–130] provides a non-technical discussion of conditions in which a delay embedding may reconstruct the original state space. To represent a point from an n-dimensional state space, one would need at least n coordinates, or independent pieces of information representing the same point in time. On the other hand, it could be argued that those coordinates could be reconstructed in one had n values of $y(\varphi^i(x))$, representing measurements at n time points, if the action of $y \circ \varphi$ couples the original coordinates of x in such a way that information is not lost. One obvious situation in which information could be lost, exemplified in Figure 2.1, is the situation where τ (or 2τ) is the period of a periodic orbit of the system.

In practice, of course, delay coordinates are not used to explicitly reconstruct the coordinates of the system's state in its "original" state space, but they are instead used to create a description of the system in the reconstructed state space.

2.2.2 Finding Optimal Time Lag

The mathematical results related to delay-coordinate embedding don't necessarily shed light on the practical problems of choosing an embedding dimension and choosing an appropriate lag. In theory at least, these two issues are somewhat independent. In a situation without random noise, the topological characteristics of the attractor that dictate the embedding dimension should be unrelated to the choice

of lag used to create the embedding. However, it is evident even in Figure 2.1 that the choice of lag does affect the geometry of the attractor in the reconstructed state space, and some choices may give better results than others. For example, it can become important to choose a lag that ensures that regions of "interesting" dynamics are spread over a large enough region to enable the dynamics to be distinguished from noise.

In general, it is desirable to avoid a very small lag that will cause high correlation between coordinates in the embedding. If delay coordinates are highly correlated, then neighboring points may be so close together that noise swamps dynamics in the embedding. On the other hand, if the lag is too large, then coordinates can become essentially uncorrelated, which can lead to trajectories whose attractor is unclear [4], [16, pp. 72–75].

It has been suggested that the first minimum of the time series' average *mutual information* be used as a criterion to identify an appropriate lag value. Finding this minimum seems to provide a good compromise between too little and too much correlation between coordinates in the embedding.

More specifically, for a time series $\{x_i\}$ defined for $i = 1, \ldots, n$, the average mutual information represents a measure of how much can be predicted about the value of $x_{i+\tau}$, given knowledge of the value of x_i, and is given by

$$I(\tau) = \sum_{i=1}^{n-\tau} f_{X,\tau}(x_i, x_{i+\tau}) \log\left(\frac{f_{X,\tau}(x_i, x_{i+\tau})}{f_X(x_i) f_X(x_{i+\tau})}\right), \tag{2.3}$$

where f_X represents the probability density of X_i and $f_{X,\tau}$ represents the joint probability density of X_i and $X_{i+\tau}$.

Fraser et al. [4] show an algorithm for estimating the densities f_X and $f_{X,\tau}$ based on a recursive division of the two-dimensional space into rectangles whose size is based on the level of detailed structure in each region. As a simple example of the process, however, Figure 2.2 shows the results of calculating mutual information for the simple harmonic oscillator and for movement data on pelvis angle from a single subject using the **`density`** and **`kde2d`** functions in R with default bandwidth and options.

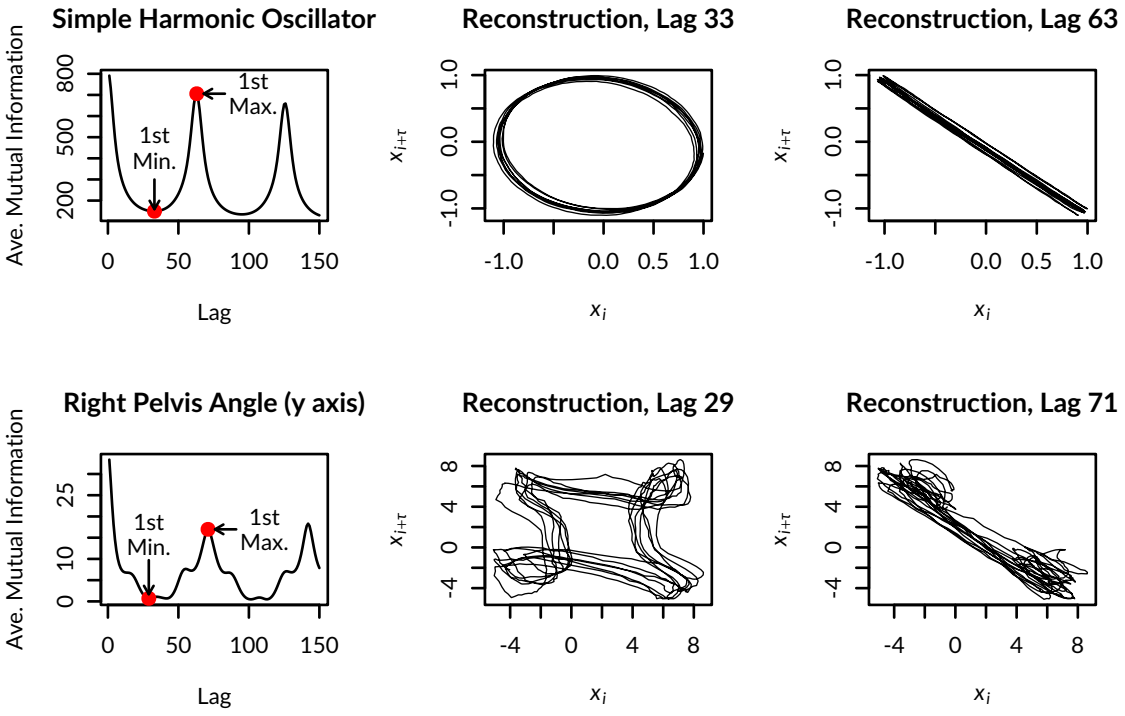

Figure 2.2 The first minimum of the average mutual information identifies a good candidate lag for reconstruction of the state space, while lag that produces a maximum yields highly-correlated coordinates.

2.2.3 Embedding Dimension

An optimal embedding dimension should be high enough that the dynamics of the attractor in phase space are completely "unfolded." From a theoretical standpoint, any dimension at or above $2D_F + 1$, where D_F refers to the fractal dimension of the system's attractor, should be sufficient, but in practice choosing an embedding dimension that is too high can cause extra computational load and can confuse the picture of the embedded attractor. Increasing the dimension increases the time delay in the original signal between the first coordinate and the last coordinate of a point in the embedding. As the last coordinate becomes less and less correlated with the first coordinate, the higher-dimensional embedding may start to look less well structured [8, p. 35][16, p. 75].

More than one method has been suggested to find a good embedding dimension. One method involves finding the minimum dimension at which an appropriate dynamic invariant ceases to change. The Lyapunov exponent itself is one such invariant. Others include correlation dimension or entropy. Stergiou [16, p. 66] mentions that the first approach to estimating embedding dimension carries a higher computational cost, and it might be desirable to use a less computationally intensive method. One such method is the method of "false nearest neighbors."

The false nearest neighbors method exploits the observation that an embedding into too few dimensions will project disparate parts of the attractor into the same region of the embedding. Therefore, the presence of these "false" neighbors is an indication that the embedding dimension is too small.

Kennel et al. [9] suggested starting with neighbors $\mathbf{x}_i$ and $\mathbf{x}_j$ in the m-dimensional embedding, then looking at the corresponding points $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ in the $(m+1)$-dimensional embedding. If $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are significantly more distant in $m+1$ dimensions, then the original neighboring points were not true neighbors.

Therefore, for each point $\mathbf{x}_i = (x_i, x_{i+\tau}, \ldots, x_{i+(m-1)\tau})$ in the embedding we can do the following:

1. In m-dimensional space, find the nearest neighbor to $\mathbf{x}_i$ by finding the point $\mathbf{x}_j$ that minimizes the Euclidean distance $||\mathbf{x}_i - \mathbf{x}_j||_m$.

2. Moving up to $(m+1)$-dimensional space, calculate the distance $||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_{m+1}$, where $\hat{\mathbf{x}}$ represents the same point in the $(m+1)$-dimensional embedding. In other words, $\hat{\mathbf{x}}_i = (x_i, x_{i+\tau}, \ldots, x_{i+m\tau})$.

3. If the distance between the points expands by a substantial margin, as measured by the ratio of the change in squared distance to the original squared distance:

$$\sqrt{\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||^2_{m+1} - ||\mathbf{x}_i - \mathbf{x}_j||^2_m}{||\mathbf{x}_i - \mathbf{x}_j||^2_m}} = \frac{|x_{i+m\tau} - x_{j+m\tau}|}{||\mathbf{x}_i - \mathbf{x}_j||_m} > R_{\text{tol}},$$

   count the nearest neighbor as a "false" nearest neighbor. Here R_{tol} is arbitrary, but a value of 10 is suggested in Kennell et al. [9].

   Stergiou [16, pp. 66–69], on the other hand proposes calculating the ratio of change in distance to original distance, and using a tolerance $R_{\text{tol}} = 15$:

$$\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_{m+1} - ||\mathbf{x}_i - \mathbf{x}_j||_m}{||\mathbf{x}_i - \mathbf{x}_j||_m} > R_{\text{tol}}.$$

4. The smallest dimension at which the percentage of false nearest neighbors drops to near zero is considered an appropriate embedding dimension.

There are computational issues with applying this algorithm indiscriminately. White noise, which Kennel characterized as a high-dimensional attractor produced by a computer's pseudo-random number generator, could be reported as being embeddable in a low-dimensional space. This seems related to the "curse of dimensionality," where even close points in a high-dimensional space may not be that close. Presumably then, those false neighbors start far enough apart that they don't move significantly farther apart, and are not identified as "false." Kennel suggests a second condition that labels points as false neighbors simply if

$$\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_{m+1}}{s_x} > A_{\text{tol}},$$

where $s_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$.

Short-term noise can also be an issue, leading dimension-setting algorithms to confuse this noise with complex dynamics. In these cases, the percentage of points identified as false nearest neighbors may decrease, then increase as dimension is increased, or may simply decrease very slowly, implying a

high embedding dimension. One way to control for these effects is to impose a rule that points too close in time cannot be considered nearest neighbors by the algorithm [5]. The time separation w, such that points $\mathbf{x}_i$ and $\mathbf{x}_j$ are only considered to be neighbors when $|i - j| \geq w$, is referred to as the Theiler window.

Finally, the dimension estimate can change depending on the size of the initial neighborhood considered when looking for a nearest neighbor. It seems that the size of the neighborhood given to the algorithm needs to take into consideration the amount of noise present in the data. If a neighborhood is defined too narrowly, there may not be enough nearest neighbors found at all. If a neighborhood is defined to be too wide, issues with non-close nearest neighbors may crop up.

Heggar, Kantz and Schreiber [5, 6] have published a set of computational algorithms for nonlinear time series analysis in their TISEAN package. These algorithms have been ported into the R package **`tseriesChaos`** [12], and include the **`false.nearest`** command for estimating percentage of false nearest neighbors.

Figures 2.3 and 2.4 show the results of running the **`false.nearest`** command on both the simple harmonic oscillator and pelvis angle examples. The simulated simple harmonic oscillator appears to be fairly well-behaved with respect to the Theiler window, although the estimated dimension is sensitive to the neighborhood size, decreasing from 3 to 2 as neighborhood size is increased.

For the pelvis angle data, the false nearest neighbor percent seems to encounter more computational variability if a small Theiler window is chosen, and the estimated embedding dimension seems to decrease from 5 to 4 as the neighborhood size is increased. From an inspection of the attractor in the reconstructed phase space in Figure 2.2, it appears that the choice of neighborhood size equal to $\frac{1}{2}s_x$ might be reasonable, since the "false" crossings of the trajectory in two dimensions are fairly spread out, compared to the overall range of x.

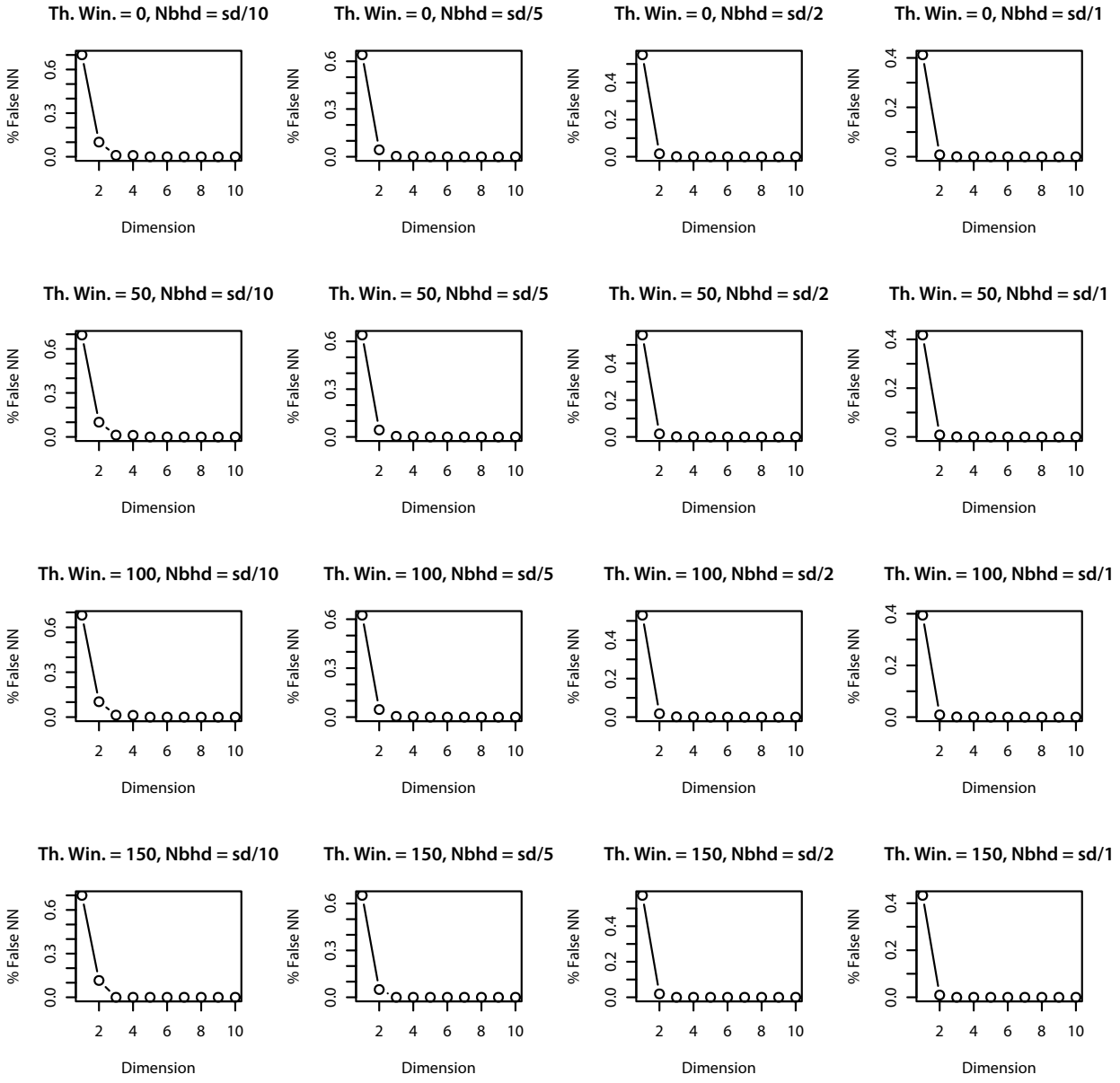

Figure 2.3 Embedding dimension estimation for the simple harmonic oscillator using the false nearest neighbors method. Both Theiler window and neighborhood size (as a fraction of standard deviation of the series) are varied.

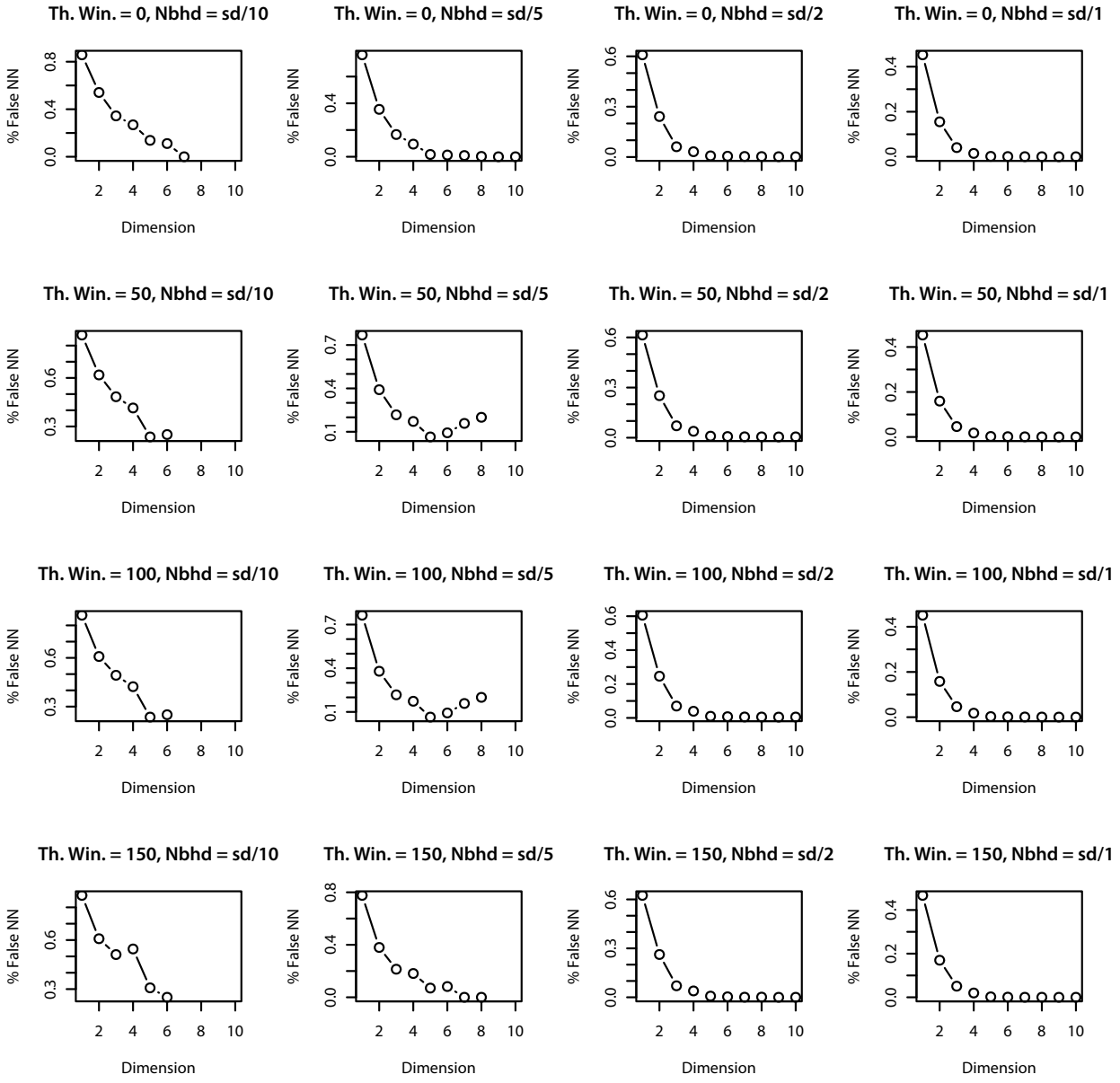

Figure 2.4 Embedding dimension estimation for the Right Pelvis Angle data using the false nearest neighbors method. Both Theiler window and neighborhood size (as a fraction of standard deviation of the series) are varied.

2.3 Lyapunov Exponents

2.3.1 Mathematical Definition of the Lyapunov Exponent

Returning to the idea of quantifying the divergence of nearby trajectories in the state space of the system, it is possible to do more than simply quantify the change in distance between trajectories. It is possible to quantify how each dimension of a neighborhood in the state space is compressed or stretched in the time evolution of the system. Figure 2.5 shows an example. Let F represent the function that maps the state space at time i to the state space at time $i+1$. A spherical region around point $\mathbf{x}_i$ with radius $d_{1,i} = d_{2,i}$ can be imagined to be stretched into an elliptical region with largest semi-axis of length $d_{1,i+1}$ and smaller semi-axis of length $d_{2,i+1}$.

This stretching can be quantified by considering the function D defined on this neighborhood, which gives the difference between the time evolutions of two points $\mathbf{x}_i$ and $\mathbf{y}_i$:

$$D(\mathbf{y}_i, \mathbf{x}_i) = F(\mathbf{y}_i) - F(\mathbf{x}_i) = \mathbf{y}_{i+1} - \mathbf{x}_{i+1}.$$

The function D can be linearized by taking $\mathbf{J}_i$ to be the Jacobian of F at the point $\mathbf{x}_i$:

$$D(\mathbf{y}_i, \mathbf{x}_i) \approx \mathbf{J}_i(\mathbf{y}_i - \mathbf{x}_i).$$

The eigenvectors of $\mathbf{J}_i$ give local directions in which the state space is either compressed or expanded by the action of F, and the eigenvalues of $\mathbf{J}_i$ give the magnitude of the compression or expansion [8, pp. 174–175].

If we desire to follow the time evolution of the system over multiple steps, a product of Jacobians $\prod_{i=1}^{N} \mathbf{J}_i$ can represent the combined compression/expansion behavior of the mapping, and we can ask about the eigenvectors and eigenvalues of this product. Define $\Lambda_j^{(N)}$ to be the jth eigenvalue of $\prod_{i=1}^{N} \mathbf{J}_i$.

As previously mentioned, in a system displaying sensitive dependence on initial conditions, it is expected that we will see an exponential divergence of points on neighboring trajectories. Therefore, it makes sense to look at the logorithms of the $\Lambda_i^{(N)}$, and we would expect those logarithms to grow linearly in N.

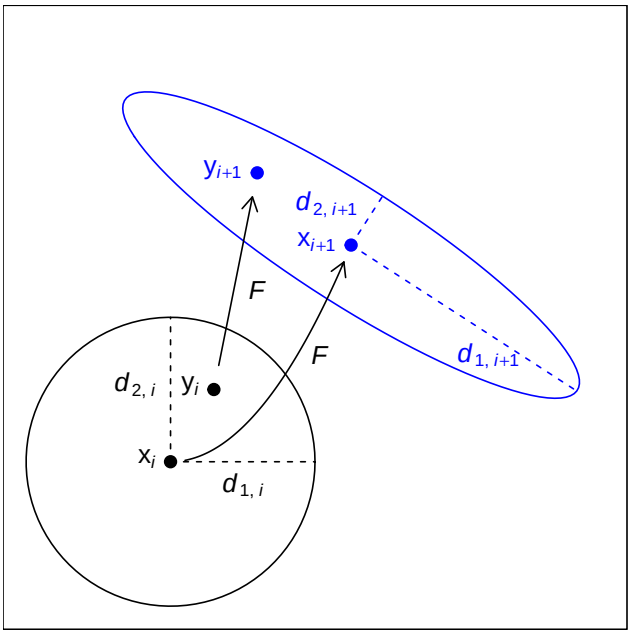

Figure 2.5 A spherical neighborhood in the state space is stretched in the time evolution of the system.

With that background, a *Lyapunov exponent* λ_i is defined as a limit on an infinite trajectory of the form

$$\lambda_i = \lim_{N\to\infty} \frac{1}{N} \ln \left| \Lambda_i^{(N)} \right|.$$

The *Lyapunov spectrum* is the collection of these exponents.

The existence and uniqueness of this limit is characterized by Kantz as "highly nontrivial" [8, p. 175], but we could say that it would be expected that this limit would be finite and positive for at least certain directions in which the system displayed sensitive dependence on initial conditions. If all Lyapunov exponents were zero, we would expect a linear time evolution, and all negative Lyapunov exponents would suggest a fixed point in the dynamics.

Kantz [8, p. 61] also points out that a deterministic system perturbed by random noise would theoretically have an infinite maximal Lyapunov exponent, no matter the magnitude of the noise. This can be justified by the fact that noise remains constant in magnitude, even as two trajectories are taken closer and closer together. Kantz also observes, however, that it is possible to estimate Lyapunov exponents in this context, as long as the error is assumed to be additive (and presumably not too large).

2.3.2 Computational Algorithms for the Lyapunov Exponent

There seem to be at least two widely-referenced algorithms for estimating the maximal Lyapunov exponent from time series data: one due to Wolf et al. [19], and another credited to Rosenstein, Collins, de Luca[13] and (independently) Kantz [7].

Wolf's algorithm follows a single reference trajectory $\{\mathbf{x}_i\}$ through the state space. At each step, it selects a nearest neighbor, subject to some conditions, and computes the distance L_i between that neighbor and the reference trajectory. After one time step of chosen width Δt, the distance L'_{i+1} between the evolved reference trajectory and that nearest neighbor is measured again. From those two measurements the log of the ratio of distances is calculated and normalized:

$$Z_i = \frac{\log_2(L'_{i+1}/L_i)}{\Delta t}.$$

At the next time point $i+1$, a new nearest neighbor is selected, and the process is repeated. At the end of the process, the largest Lyapunov exponent is estimated by the average of the Z_i's:

$$\lambda_1 \approx \frac{1}{M}\sum_{i=1}^{M} Z_i.$$

The implication with regard to this algorithm seems to be that the time step Δt and the embedding dimension m are chosen before the algorithm is run, and the result is a single Lyapunov exponent [16, pp. 94–99].

Rosenstein's algorithm, on the other hand, seems to emphasize more thorough "averaging" of expansion rates, a more extensive set of outputs, and human interpretation of those outputs.

In Rosenstein's algorithm, to obtain an estimate of the divergence of neighboring trajectories, a base point $\boldsymbol{x_i}$ in the state space is first selected, followed by a neighboring point $\boldsymbol{x_j}$ that is within a small radius ε of $\mathbf{x}_i$. The distance between the two trajectories after Δt time steps is calculated: $|\mathbf{x}_{i+\Delta t} - \mathbf{x}_{j+\Delta t}|$ This process is then repeated over *all* points within the ε-neighborhood of $\boldsymbol{x_i}$, and with other choices of base point. Results are averaged to obtain the final estimate:

$$S(\Delta t) = \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x_i}\in\mathcal{B}} \ln\left(\frac{1}{|N_\varepsilon(\boldsymbol{x_i})|}\sum_{\mathbf{x}_j\in N_\varepsilon(\mathbf{x}_i)} |\mathbf{x}_{i+\Delta t} - \mathbf{x}_{j+\Delta t}|\right)$$

where $\mathcal{B}$ represents the set of all possible base points such that $|N_\varepsilon(\mathbf{x}_0)| \geq N_{\min}$. In other words, $\mathcal{B}$ contains all base points that meet the requirement of having a certain number of close neighbors. This last requirement helps to limit the effect of unusual random variations in the data that lead to outlying points in state space.

In this algorithm, important parameters include the choice of ε, and the choice of embedding dimension m. Kantz recommends running the algorithm using several values for these parameters.

Once this is done, a plot of $S(\Delta t)$ against Δt is created. If exponential divergence is present, an initial linear portion of that graph should be observed. (Eventually, there will be a maximum distance that any points can travel from one another, and the linear pattern won't hold up after a certain value of Δt.) Linear regression is then used to estimate the slope of the linear portion of the graph, which provides the estimate of the largest Lyapunov exponent. Again, Kantz cautions that this slope should be robust against changes in m and ε. If not, there may not be strong evidence for exponential divergence of points. Kantz [8, pp. 62–67]. also observes that particular small-scale aspects of these graphs can be used to gain some insight into the scale of noise in the system

Rosenstein's algorithm seems to be often credited with being more robust, especially when used with smaller data sets, although Stergiou argues that those qualities are sometimes misunderstood and that Wolf's algorithm often performs better with real-world data sets [16, pp. 104–107] [8, p. 62].

Stergiou [16, p. 106]. also makes the point that the authors of both algorithms suggest a minimum sample size of 10^D, where D is the dimension of the attractor.

2.3.3 Examples of Lyapunov Exponent Estimation

A Matlab implementation of Wolf's algorithm was provided by MOVCENTR as part of the analysis they provided for Dr. Bird's movement data. That code was, in turn, based on Matlab code provided by Wolf [18] in the Matlab file exchange site in 2016. MOVCENTR indicated that Wolf's Matlab code had been patterned after the Fortran syntax of earlier versions of the code, and theirs had been modified to make better use of Matlab's capabilities. For use in this paper, I translated the MOVCENTR code into R code, which is listed in Appendix A.

Wolf's algorithm requires several parameters in order to run:

- **`Fs`**: The sampling frequency. What fraction of a time unit does each data point in the time series represent? Changing **`Fs`** merely results in a multiplicative change in the final estimated Lyapunov exponent.
- **`tau`**: The time lag used in reconstructing the state space.
- **`dim`**: The embedding dimension m used in reconstructing the state space.
- **`evolve`**: The number of forward time steps at which divergence of neighboring points is measured. (Δt, as mentioned in the previous section.)
- **`SCALEMX`**: The maximum distance to which a neighboring point will be followed before choosing a new "nearest neighbor" to follow. By default, this value is chosen to be $\frac{1}{10}$ of the time series' range.
- **`SCALEMN`**: The length below which it is thought that noise dominates the behavior of the attractor. Nearest neighbors are not selected below this distance threshold. By default, this value is chosen to be 10^{-4}.
- **`ANGLMX`**: The maximum angle difference between the evolved neighboring point and its replacement, relative to the evolved reference point. The algorithm appears to try to find a neighboring point with angle difference less than **`ANGLMX`**, and if it fails, it simply finds the closest neighboring point with distance greater than **`SCALEMN`**. By default, this value is taken to be $\frac{\pi}{6}$.

If **`tau`** and **`dim`** are taken as fixed, then two parameters that have a clear effect on the resulting estimate of the Lyapunov exponent are the choice of **`evolve`** and **`SCALEMX`**.

The graph in Figure 2.6 shows the various estimates of the Lyapunov exponent for the pelvis angle data set, generated with values of **`evolve`** running from 2 to 20 and values of **`SCALEMX`** chosen to be between $\frac{1}{5}$ and $\frac{1}{100}$ of the range of the time series. Note that there is significant variation in the estimated Lyapunov exponents, with a decreasing trend as **`evolve`** increases, and an increasing trend as

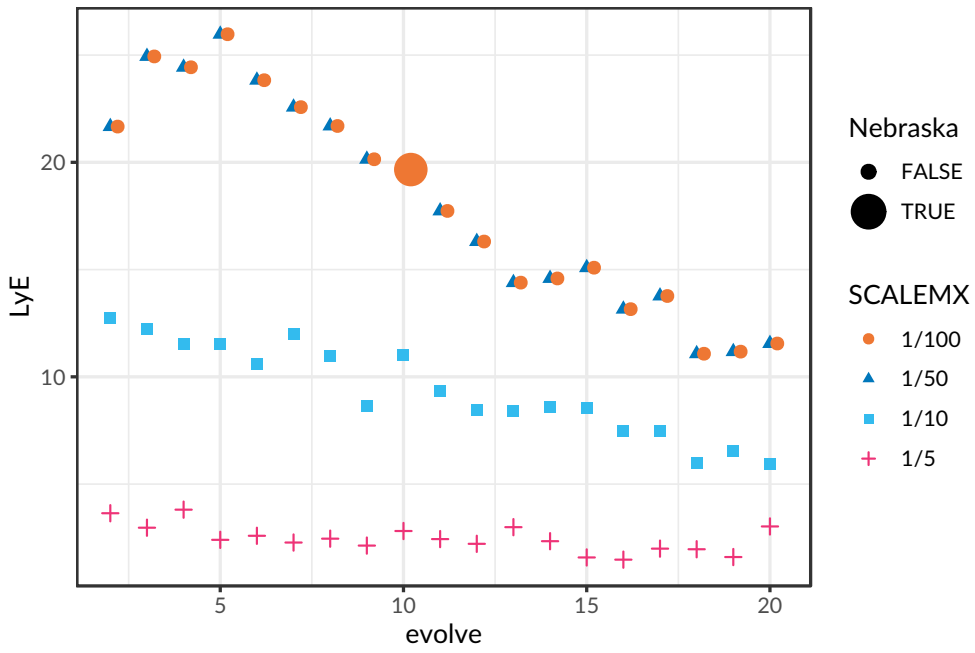

Figure 2.6 Estimates of the Lyapunov exponent from Wolf's algorithm for various values of **evolve** and **SCALEMX**. The MOVCENTR value is indicated by the large dot. Note that **SCALEMX** is expressed as a fraction of the time series' range.

SCALEMX decreases. Values obtained from a choice of **SCALEMX** to be $\frac{1}{50}$ or $\frac{1}{100}$ of the range were indistinguishable. Jitter is used in the graph in order to show both series.

MOVCENTR set **SCALEMX** equal to $\frac{1}{100}$ of the range of the time series and set **evolve** equal to 5% of the sampling frequency (i.e., **evolve** = 10.) With those values, the R implementation returned a value of the Lyapunov exponent comparable to the MOVCENTR results.

It should be noted, however, that Stergiou's description of the algorithm suggests inspection of the calculated values of the Lyapunov exponent, with the final selection of **evolve** to be with a region where there is a plateau in the graph—in other words, a region where the estimated Lyapunov exponent is relatively constant. This region is expected to correspond to the time scale in which exponential divergence occurs [16, pp. 96–97].

A value of **evolve** equal to 10 may lie at the right-most edge of a plateau when **SCALEMX** is chosen to be $\frac{1}{10}$ of the time series range, but does not seem to correspond to a plateau for smaller values of **SCALEMX**.

Application of Rosenstein's algorithm also requires several parameters, with labels matching those used in the **tseriesChaos** package.

- **m**: the embedding dimension m.
- **d**: The time lag used in reconstructing the state space.
- **k**: The number of neighbors considered around each base point.
- **eps**: The radius of the neighborhood considered when finding neighbors.
- **ref**: The number of base points used in the algorithm.
- **t**: The Theiler window used to exclude points that are too close in time.

Figure 2.7 shows results from applying Rosenstein's algorithm to the pelvis angle data set. Graphs on the top row show the effects of various choices of embedding dimension and neighborhood radius. At the smallest dimension $m = 3$, all choices of neighborhood radius produce very similar results. However, as dimension increases, the smaller neighborhood radius choices tend to produce more uneven results. Further analysis indicated that smaller radius choices at higher dimensions led to fewer acceptable base points being identified by the algorithm. At the extreme, when the algorithm was told to find 200 base points with four neighbors within a radius of 1 in dimension 6, only 6 base points were found that met those criteria. This is consistent with the fact that distances between points tend to grow as dimension is increased. A radius of 2 or 2.5 worked well in all dimensions tested.

It should also be noted that embedding dimension 3 produced a graph of log dispersion distance verses time that did not resemble the others, while dimensions 4 through 6 were similar. This outcome is consistent with the fact that embedding dimension 4 was identified by the false nearest neighbors method as the lowest acceptable embedding dimension.

The graphs of log dispersion distance verses time step seemed to display a linear region between time steps 2 and 8, and the bottom left graph shows the fitted lines whose slope estimates the Lyapunov exponent in Rosenstein's algorithm. Numeric values for LyE have been converted to match those from Wolf's algorithm by multiplying by $\frac{200}{\log 2}$ to account for the sampling frequency of 200 input

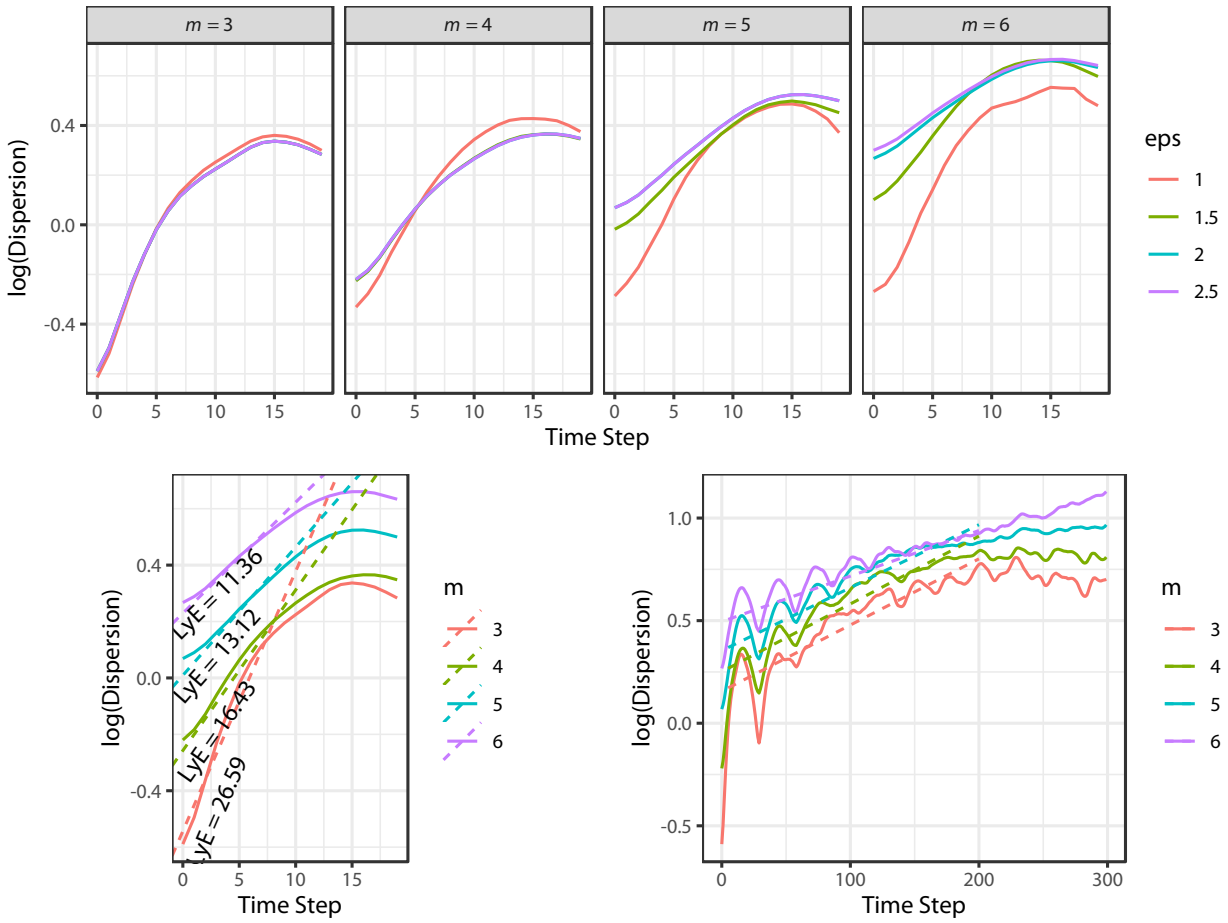

Figure 2.7 Results from Rosenstein's algorithm. The top row of graphs show the effects of various choices of embedding dimension and neighborhood radius. The bottom left graph shows estimates of Lyapunov exponent for each embedding dimension, with line fit between time steps 2 and 8. Numeric values for LyE have been converted to match those from Wolf's algorithm by multiplying by $\frac{200}{\log 2}$ to account for the sampling frequency and the use of $\log_2$ rather than natural logarithm in Wolf's algorithm.

to Wolf's algorithm and the use of $\log_2$ rather than natural logarithm in Wolf's algorithm. For embedding dimensions between 4 and 6, estimates seem relatively stable, with dimensions 5 and 6 agreeing most closely. This appears to be consistent with the fact that an embedding dimension of 5 was the first to reduce the percentage of false nearest neighbors to nearly zero, while an embedding dimension of 4 produced just slightly higher false nearest neighbor percentage.

Estimates of Lyapunov exponents between 16.43 (m = 4) and 11.36 (m = 6) appear to be most consistent with those generated by Wolf's method when using **SCALEMX** equal to $\frac{1}{10}$ of the time series range, which is equal to approximately 1.4. Since **SCALEMX** and **eps** play a similar role—determining the radius around a base point at which we pay attention to dispersion rates, it is perhaps not surprising that similar results would be obtained for similar values.

Rosenstein's requirement of robustness at several neighborhood radii and several embedding dimensions might then call into question the use of the smaller value of **SCALEMX** by MOVCENTR, although, in fairness, a rationale for their choice was not provided with their code, and there may be a systematic way in which **SCALEMX** was chosen.

The bottom right graph in Figure 2.7 shows a graph of log dispersion distance over 300 time steps. Note that there is evidence of a periodic pattern in the dispersion distances, whose period is roughly equal to the time lag used in the delay embedding that reconstructed the phase space. Use of other time lags in the embedding creates dispersion distance graphs with the same periodicity as the embedding time lag, thus that periodicity can be considered an artifact of the embedding.

Disregarding periodicity, a much longer interval of linear growth may be identified in the last plot, between the initial time step and, roughly, time step 200. This raises the question of the time scale on which we expect to see "chaotic" behavior in the time series. The entire time series consists of 1394 measurements, which at 200 frames per second corresponds to approximately 7 seconds. An open question might be whether we expect to see chaotic behavior on the scale of a fraction of a stride ($\Delta t \approx 20$), which might, for example, correspond to small, continuous corrections, or whether we expect to see chaotic behavior on the order of one complete stride ($\Delta t \approx 200$), which might correspond to uneven foot placement on the uneven surface.

Some authors [11] choose to set Δt by examining the plot of log dispersion distance for linear regions, as previously described. On the other hand, Stergiou [16, p. 106] observes that it has also become standard in gait-related literature, when using Rosenstein's algorithm, to choose a time interval related to a half or whole stride.

Finally, it may be important to mention that this analysis depends on the assumption that there is non-linear behavior to be identified in this time series. One approach to that question seems to be a subjective evaluation of graphs like those produced by Rosenstein's algorithm. Surrogation tests are also used to compare the actual time series to surrogate time series created to have similar properties but no nonlinear dynamics. At present, that analysis is beyond the scope of material presented in this paper, but it is true that MOVCENTR reported evidence of non-linear dynamics in most time series examined, including this pelvis angle data. However, there remain some unanswered questions as to how those test were performed.

Other authors [10, 3], especially when using the half or whole-stride "standard" when applying Rosenstein's approach, forego concern about evaluating the presence of non-linear dynamics, and simply look at group-wise comparisons of Lyapunov exponents measured at these standard times.

CHAPTER 3. Application to Movement Data

This final section will briefly discuss results seen in the cambered running data gathered by Dr. Bird's lab. It should be noted that the data set was not originally collected with an eye to analysis of nonlinear measures of variability. If data were to be collected again for that express purpose, a longer time period would likely be collected.

3.1 The Data

In each treatment condition, study participants were given a one-minute familiarization period where they ran at the set camber. After that familiarization period, motion-capture data collection began, continuing for approximately eight seconds at a rate of 200 frames per second, usually encompassing between 7 and 10 strides.

Positions and angles were measured relative to three axes:

- the x axis, in the direction of the treadmill track,
- the y axis, parallel to the ground and transverse to the treadmill direction, and
- the z axis, perpendicular to the ground.

Angles represent rotation in the plane perpendicular to the given axis.

Lyapunov exponents were calculated by MOVCENTR for

- ASIS (x, y and z),
- left and right hip angles (x and y),
- left and right knee angles (x), and
- pelvis angles (y).

3.2 Validity of the Lyapunov Estimate

As noted previously, in order for the Lyapunov exponent to be meaningful, it is assumed that measured values are the realization of a stationary process and that the system's trajectory through the reconstructed state space follows an *attractor*. It is not assumed that the trajectory is exactly periodic in nature, but it is assumed that the trajectory displays no non-stationary trend. This is a reasonable assumption for many, but not all, variables associated to a repetitive motion such as running.

Because the motion capture data gathered from each subject covered a time period that encompassed between seven and ten strides, we might then expect to see a state-space trajectory with a cyclic structure that "repeats" between 7 and 10 times, depending on subject. In that case, it would be reasonable to assume that a point on the trajectory would have at most between 7 and 10 close neighbors. Although Lyapunov exponents can be calculated with such a data set (1400 data points $> 10^3$, and we assume the dimension of the attractor is less than or equal to three), the short time period covered by the data means that there is not much leeway in dealing with variables that do not exhibit a roughly cyclic trajectory. In other words, if a variable's trajectory does not create many "close neighbors," or if those close neighbors happen by chance, rather than because of the cyclic nature of the trajectory, the estimation of the Lyapunov exponent will be, at best, suspect, and at worst unusable.

The analysis performed by MOVCENTR [15] found that most variables had an optimal embedding dimension greater than $n = 3$. For that reason, trajectories are visualized here in two ways: first by projecting onto the first three dimensions of the reconstructed state space, and second by using principal components analysis to project the state space into three dimensions.

It was found that the position coordinates of ASIS were subject to non-cyclic behavior—possibly because of the runner's drift on the treadmill over the relatively short recording period. On the other hand, angular measurements exhibited trajectories that did appear to move about an attractor. Examples from several subjects are shown in Figures 3.1 and 3.2.

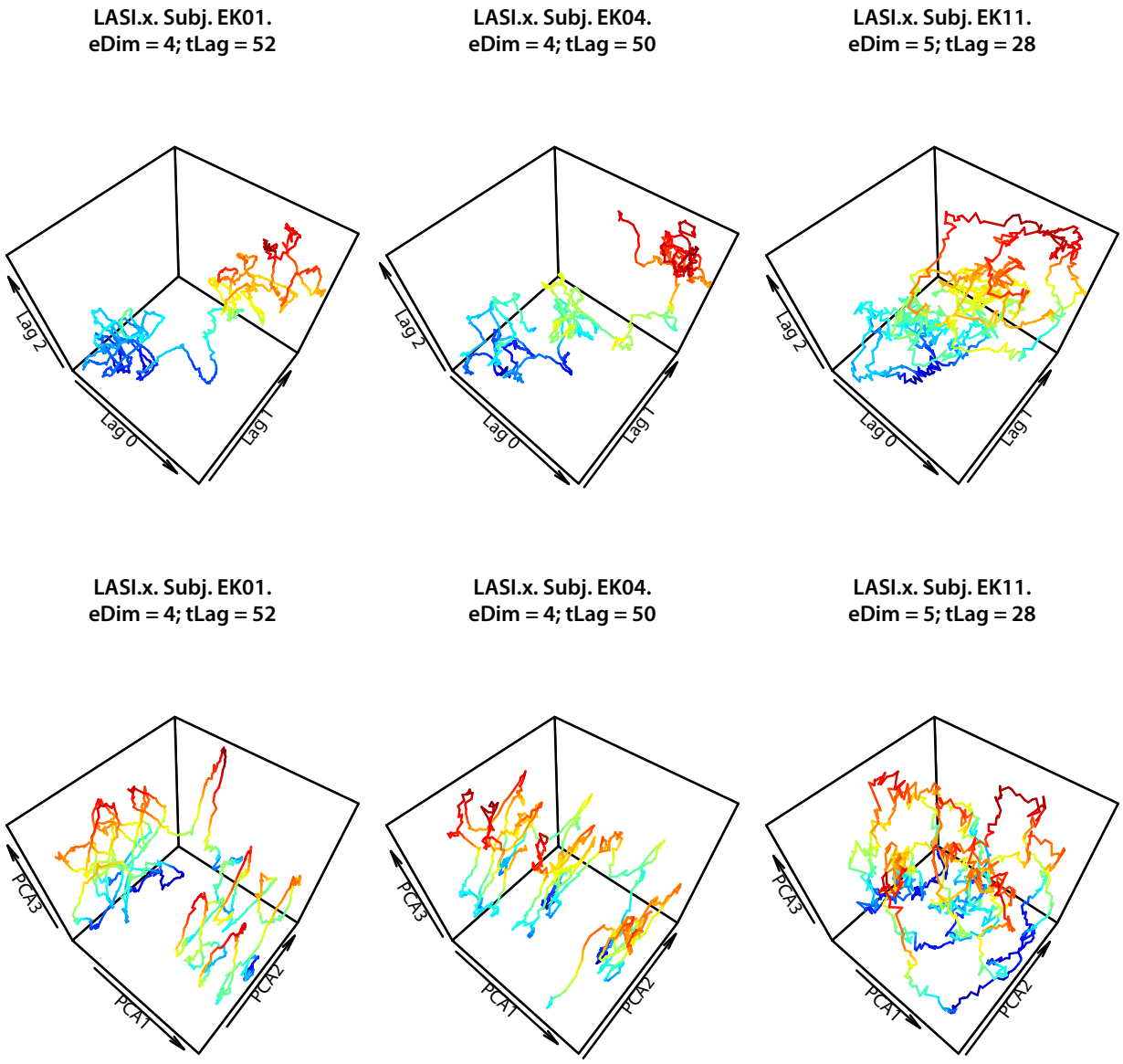

Figure 3.1 A collection of subjects for whom the LASI variable did not appear to be conducive to use of the Lyapunov Exponent.

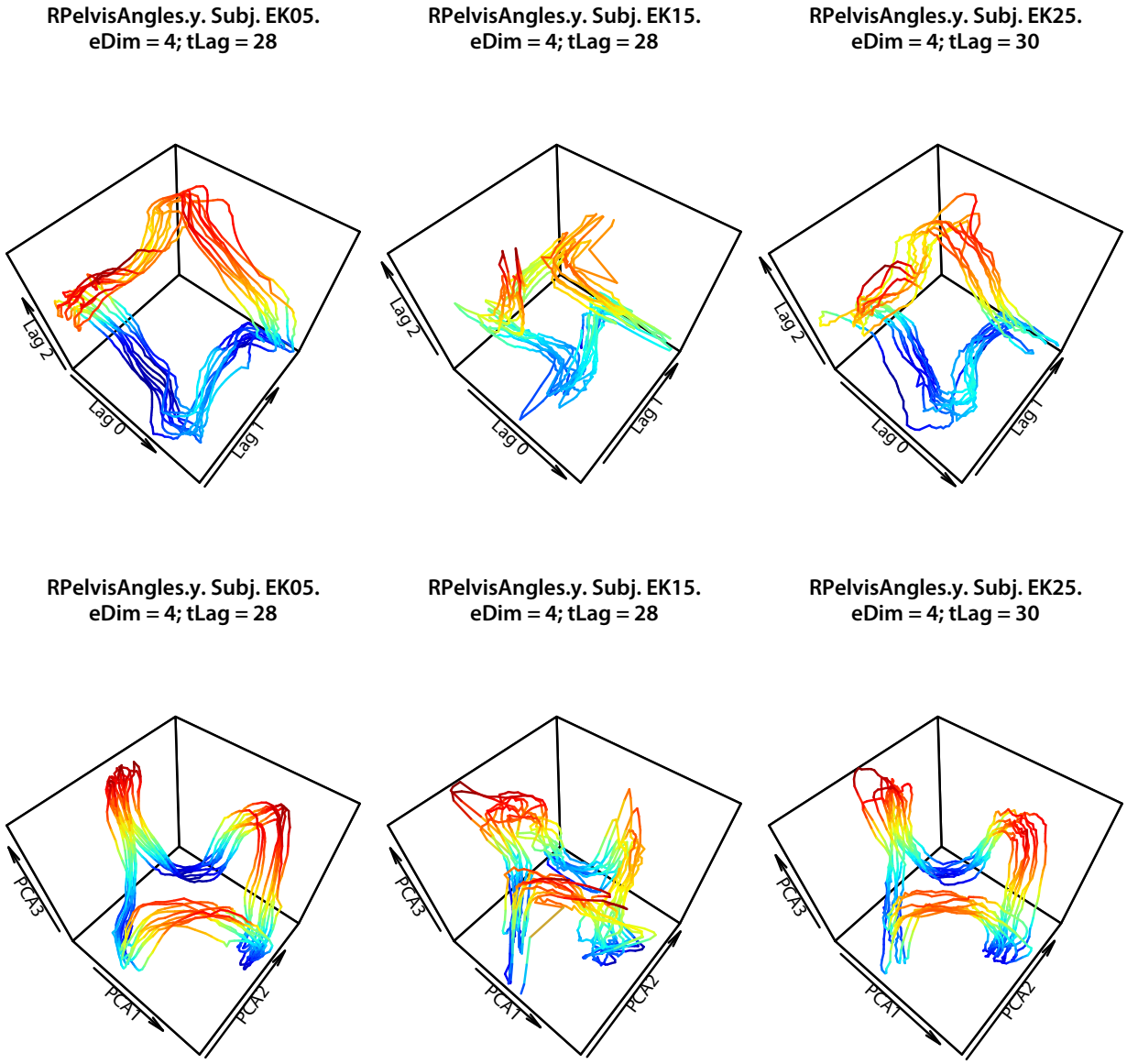

Figure 3.2 Use of the Lyapunov exponent appears to be reasonable with the pelvic angle.

3.3 Differences in Lyapunov Exponents

3.3.1 MOVCENTR Report

Among the angular measures, Dr. Bird's lab decided that "Pelvis Angle" was of most interest. Figure 3.3 shows a boxplot of Lyapunov exponents, calculated by MOVCENTR, under each camber condition. Note that pelvis angle does not have a separate measurement for left and right side.

Although it may appear that low and high camber have larger Lyapunov exponents, on average, than flat camber, this effect is not statistically significant. A linear mixed model with camber as fixed effect and subject as random effect finds no statistically significant difference between cambers ($p = 0.775$).

The plot of individual values in Figure 3.3 indicates that there does not seem to be a consistent increasing or decreasing trend in Lyapunov exponent, and in fact only 51.7% of subjects showed an increase in Lyapunov exponent between flat and high cambers.

Similar analysis of other measures that appeared amenable to Lyapunov exponent calculation showed similar null results.

3.3.2 Reanalysis with Larger `SCALEMX`

As noted in Section 2.3.3, Rosenstein's method gave some indication that a larger distance scale might be appropriate for this particular data set. Rerunning Wolf's algorithm on the pelvis angle data from all subjects, using a value of **`SCALEMX`** equal to $\frac{1}{10}$ of each time series range, rather than $\frac{1}{100}$ of the range, yielded results as shown in Figure 3.4.

Although variability in Lyapunov exponents was still large, variability was decreased using the larger value of **`SCALEMX`**. In the boxplot of individual values, now 69% of subjects saw an increase in Lyapunov exponent between flat and high camber, and the difference in means between the camber groups was now statistically significant at the 5% level ($p = 0.046$).

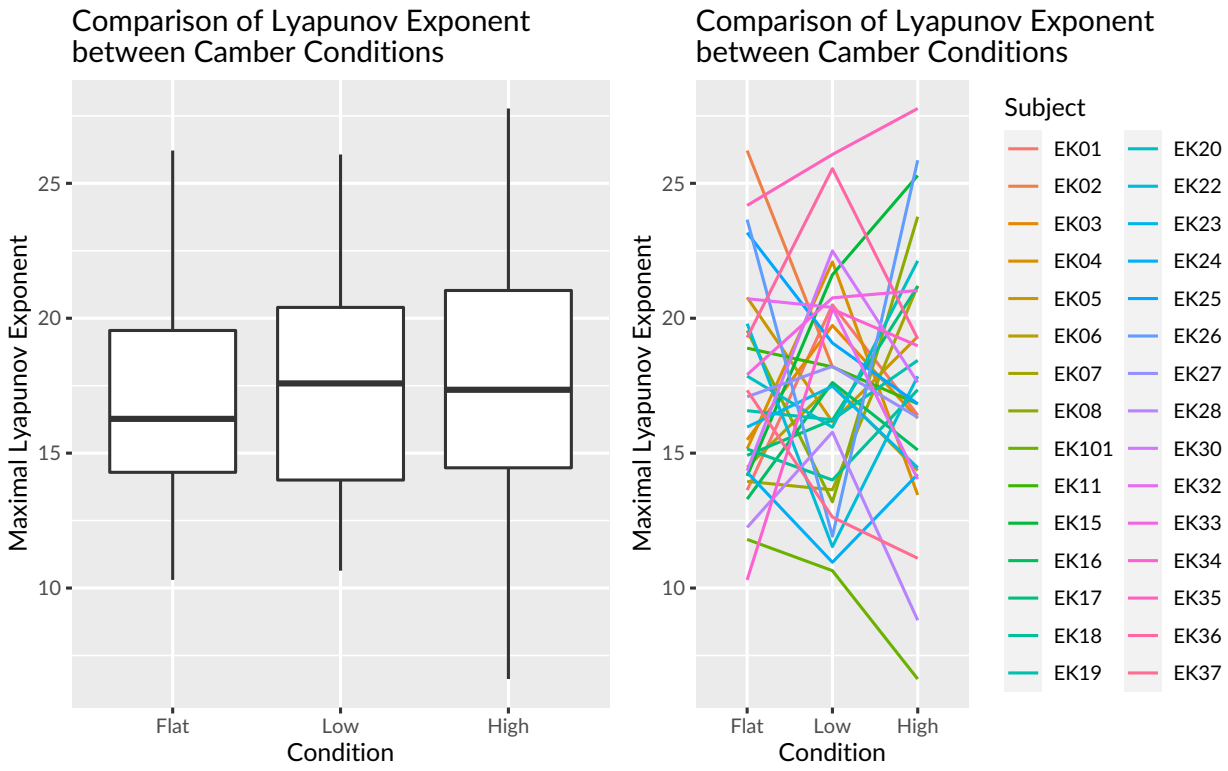

Figure 3.3 At **SCALEMX** equal to $\frac{1}{100}$, median Lyapunov exponent is larger in the low and high camber treatments, although there is a great deal of variation within individual subjects, as shown in the second graph.

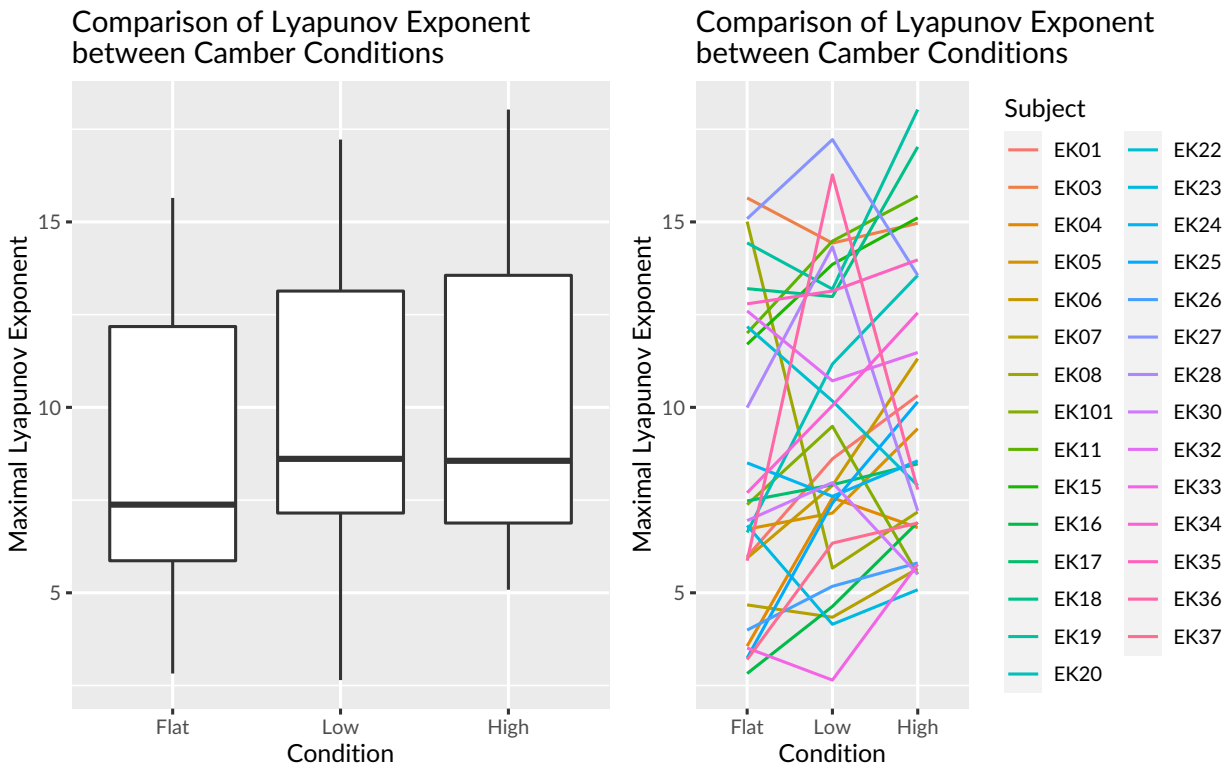

Figure 3.4 At **SCALEMX** equal to $\frac{1}{10}$, median Lyapunov exponent is larger in the low and high camber treatments, and there is less variation within individual subjects, compared to that seen in Figure 3.3. More subjects showed an increasing trend in Lyapunov exponent as camber increased.

3.4 Further Applications

The actual data collected by Dr. Bird's lab may not cover a long enough time period/enough cycles to allow this sort of analysis. There is interest in evaluating this technique on data that was collected, from the start, with these non-linear methods in mind. Both time span and frame rate of the data collected can be optimized for use with these nonlinear algorithms.

There is also interest in other methods, beyond the Lyapunov exponent, that may be used to evaluate these reconstructed embeddings in a holistic way.

CHAPTER 4. Discussion and Personal Reflection

Perhaps because of my mathematical background, I find these nonlinear techniques interesting, and would love to see them provide meaningful insight into data sets like those studied in Dr. Bird's lab here at Truman. It's certainly the case that others are using them for movement analysis and publishing results.

Initial results using the Lyapunov statistics reported by MOVCENTR were disappointing, however. Initially, I was tempted to attribute the large variability in Lyapunov exponent to the short time frame under which Dr. Bird's data was collected, and the subsequent small number of repetitions defining the attractor in the phase space reconstruction.

As I looked more closely at the number of parameters required by both algorithms, and the difference they made in the final calculated estimate for the Lyapunov exponent, I became less certain that there is a single Lyapunov exponent that *can* be used for statistical analysis. The author of one study we reviewed wrote, "All of the nonlinear time-series analysis algorithms mentioned in this section are known to be sensitive to data and parameter effects. These systematic uncertainties preclude the use of traditional statistics to assess or compare their results…" [11]. The author of that paper seemed to view purely descriptive use of those statistics as the solution to the problem of parameter choice effects.

On the other hand, I was heartened to see that a more careful look at the dispersion distance graphs from Rosenstein's algorithm led to a choice of distance scale that also led to more consistent results from Wolf's algorithm, and a more statistically significant difference between camber conditions.

It would of course be wrong to choose parameters in order to obtain a more statistically significant result, but in this case there seemed to be some genuine basis for the choice, and the choice was made before the results were known. It's hard not to view them as "better."

I have not looked deeply enough at the literature that tests the performance of these and other Lyapunov algorithms, and I don't know whether there are canonical choices for algorithm

parameters—both Kantz and Stergiou give general guidance, but few particulars in what I've read. I don't know how much trust can be placed in statistics calculated through a process whose parameters require a substantial degree of tweaking for each individual time series.

On the other hand, I can imagine automating a look at the data that can identify the range of distances of nearest neighbors and set the neighborhood distance limits of both algorithms in a semi-intelligent way. If that part of the algorithm can be automated, there may be hope that Lyapunov exponents might be consistent enough to allow further analysis. The next step in my work with Dr. Bird will likely be a more thorough evaluation of the literature in this area.

BIBLIOGRAPHY

[1] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Third Edition. Switzerland: Springer, 2016.

[2] Katelyn Campbell et al. "The Effect of Camber on Women's Running Kinematics". 2019.

[3] Scott A. England and Kevin P. Granata. "The influence of gait speed on local dynamic stability of walking." In: *Gait Posture* 25.2 (2007), pp. 172–178.

[4] Andrew M. Fraser and Harry L. Swinney. "Independent coordinates for strange attractors from mutual information". In: *Phys. Rev. A* 33 (2 Feb. 1986), pp. 1134–1140. DOI: 10.1103/PhysRevA.33.1134. URL: https://link.aps.org/doi/10.1103/PhysRevA.33.1134.

[5] Rainer Hegger, Holger Kantz, and Thomas Schreiber. "Practical implementation of nonlinear time series methods: The TISEAN package". In: *Chaos* 9 (1999), p. 413. DOI: 10.1063/1.4837095.

[6] Rainer Hegger, Holger Kantz, and Thomas Schreiber. *TISEAN 3.0.1: Nonlinear Time Series Analysis*. 2007. URL: https://www.pks.mpg.de/~tisean/Tisean_3.0.1/index.html (visited on 10/16/2020).

[7] Holger Kantz. "A Robust Method to Estimate the Maximal Lyapunov Exponent of a Time Series". In: *Physics Letters A* 185.1 (1994), pp. 77–87.

[8] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge: Cambridge University Press, 1997.

[9] Matthew B. Kennel, Reggie Brown, and Henry D. I. Abarbanel. "Determining embedding dimension for phase-space reconstruction using a geometrical construction". In: *Phys. Rev. A* 45 (6 Mar. 1992), pp. 3403–3411. DOI: 10.1103/PhysRevA.45.3403. URL: https://link.aps.org/doi/10.1103/PhysRevA.45.3403.

[10] Thurmon E. Lockhart and Jian Liu. "Differentiating fall-prone and healthy adults using local dynamic stability." In: *Ergonomics* 51.12 (2008), pp. 1860–1872. DOI: 10.1080/00140130802567079.

[11] Nicole Look et al. "Dynamic stability of running: The effects of speed and leg amputations on the maximal Lyapunov exponent". In: *Chaos* 23 (2013). DOI: 10.1063/1.4837095.

[12] Antonio Fabio Di Narzo. *tseriesChaos: Analysis of Nonlinear Time Series*. 2019. URL: https://cran.r-project.org/web/packages/tseriesChaos/index.html (visited on 10/16/2020).

[13] Michael T. Rosenstein, James J. Collins, and Carlo J. De Luca. "A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets". In: *Physica. D* 65.1-2 (1993), pp. 117–134.

[14] Tim Sauer, James A. York, and Martin Casdagli. "Embedology". In: *Journal of Statistical Physics* 65 (3-4 1991), pp. 579–616. DOI: 10.1007/BF01053745.

[15] Ben Senderling. *Nonlinear Analysis for Truman State U*. University of Nebraska Omaha Department of Biomechanics Center for Research in Human Movement Variability (MOVCENTR), Nonlinear Analysis Core, 2019.

[16] Nicholas Stergiou. *Nonlinear Analysis for Human Movement Variability*. Boca Raton, FL: CRC Press, 2016.

[17] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by D.A. Rand and L.-S. Young. Vol. 898. Lecture Notes in Mathematics. Berlin, Heidelberg, New York: Springer Verlag, 1981, pp. 366–381.

[18] Alan Wolf. *Wolf Lyapunov exponent estimation from a time series*. 2020. URL: https://www.mathworks.com/matlabcentral/fileexchange/48084-wolf-lyapunov-exponent-estimation-from-a-time-series (visited on 10/17/2020).

[19] Alan Wolf et al. "Determining Lyapunov exponents from a time series". In: *Physica D: Nonlinear Phenomena* 16 (3 1985), pp. 285–317. DOI: 10.1016/0167-2789(85)90011-9.

APPENDIX A. MOVCENTR CODE, TRANSLATED TO R

```
# This code is a translation to R of the Matlab function LyE_Knarr_Mod which
# was provided by the Stergiou lab in the resutls they sent to us. I have
# made a few comments and fixed a few of what appear to be bugs in the code,
# marked with "SLT." Otherwise, further comments below taken fromt the
# original code:
#
# inputs  - X, time series
#         - Fs, sampling frequency
#         - tau, time lag
#         - dim, embedding dimension
#         - evolve, parameter of the same name from Wolf's 1985 paper
#         - SCALEMX, length of which the local structure of the attractor
#           is no longer being probed
#         - SCALEMN, length below which noise predominates the attractors
#           behavior
#         - ANGLMX, maximum angle used to constrain replacements
#         - ZMULT, multiplier used to increase SCALEMX, unused in the
#           current version of the code
# outputs - out, matrix detailing variables at each iteration
#         - LyE, largest lyapunov exponent
# Remarks
# - This code calculates the largest lyapunov exponent of a time series
#   according to the algorithm detailed in Wolf's 1985 paper. This code has
#   been aligned with his code published on the Matlab file exchange in
#   2016. It will largely find the same replacement points, the remaining
#   difference being in the replacement algorithm.
# - The varargin can be used to specify some of the secondary parameters in
#   the algorithm. All of the extra arguements must be specified if any are
#   to be specified at all. Otherwise defaults are used.
# - ZMULT is not currently used in the code but was in a previous version.
#   Its place in the subroutine inputs and outputs was kept in case it is
#   put back in.
# - It should be noted that the process in the searching algorithm has a
#   significant impact on the resulting LyE.
# Prior - Created by Shane Wurdeman, unonbcf@unomaha.edu
#       - Adapted by Brian Knarr, unonbcf@unomaha.edu
#       - The code previously was influenced heavily by the FORTRAN syntax
#         published in Wolf's 1985 paper. These were modified to better
```

```
#         take advantage of MATLAB and speed up the code.
# Mar 2017 - Modified by Ben Senderling, unonbcf@unomaha.edu
#          - Changed parameter "n" to "evolve."
#          - Changed "ZMULT" back to 1.
#          - Aligned the code with Wolf's Matlab File Exchange submission
#            to find the same replacement points. This is now essential his
#            algorithm but retains the speed of previous versions.

psr_deneme <- function(x, m, tau, npoint=NULL){
  # Phase space reconstruction
  # x : time series
  # m : embedding dimension
  # tao : time delay
  # npoint : total number of reconstructed vectors
  # Y : M x m matrix
  # author:"Merve Kizilkaya"

  N <- length(x)
  if(!is.null(npoint)){
    M <- npoint
  } else {
    M <- N-(m-1)*tau
  }

  Y <- matrix(numeric(M*m), nrow=M)

  for(i in 1:m){
    Y[, i] <- x[(1:M)+(i-1)*tau]
  }
  Y
}

find_next_point <- function(flag,theta,Ydisti,SCALEMX,SCALEMN,ZMULT,ANGLMX){

  # Restrict search based on distance and angle
  PotenDisti <- Ydisti
  PotenDisti[Ydisti<SCALEMN | theta>=ANGLMX] <- NaN
  # SLT: Debug
  if(all(is.na(PotenDisti))){warning("No points left inside angle!")}

  next_point <- 0
  if(flag==0){
    # find min distance after excluding points
    next_point <- which.min(PotenDisti)
```

```
    # If closest angle point is within angle range -> point found and reset
    # search space
    # SLT: Added check on length of next_point since it throws an error if
    # there is no next point that satisfies distance and angle conditions.
    if((length(next_point) > 0) && PotenDisti[next_point] <= SCALEMX){
        # SLT: I feel that there should be something more with ANGLMX. Why is
        # it being reset if it's never changed?
        ANGLMX <- 30*pi/180
        thbest <- abs(theta[next_point])
        #return; #SLT: Don't think this is needed. And need to set up list in
                 # R, so do it at the end.
    }
    else{
        next_point <- 0
        flag <- 1
    }
  }
  if(flag==1){
    PotenDisti <- Ydisti
    PotenDisti[Ydisti<SCALEMN] <- NaN
    # SLT: Debug
    if(all(is.na(PotenDisti))){error("No points left without angle!")}
    next_point <- which.min(PotenDisti)
    thbest <- ANGLMX
  }
  list(
    next_point=next_point,
    ZMULT=ZMULT,
    ANGLMX=ANGLMX,
    thbest=thbest,
    SCALEMX=SCALEMX
  )
}


GetNextPoint <- function(flag, Y, current_point, current_point_pair, NPT,
                         evolve, SCALEMX, SCALEMN, ZMULT, ANGLMX){

  # Distance from evolved point to all other points
  Yinit <- matrix(rep(Y[current_point+evolve,], NPT), byrow=TRUE, nrow=NPT)
  Ydiff <- (Yinit - Y[1:NPT,])^2
  Ydisti <- sqrt(rowSums(Ydiff))

  # Exclude points too close on path and closer in distance than noise (SCALEMN)
```

```
  range_exclude <- ((current_point+evolve)-10):((current_point+evolve)+10)
  range_exclude <- range_exclude[range_exclude >= 1 & range_exclude <= NPT]
  Ydisti[range_exclude] <- NaN

  end_dist <- sqrt(sum((Y[current_point+evolve,] -
                        Y[current_point_pair+evolve,])^2));

  # Vector from evolved point to all other points
  Vnew <- matrix(rep(Y[current_point+evolve,], NPT), byrow=TRUE, nrow=NPT) -
          Y[1:NPT,]

  # Vector from evolved point to evolved point pair
  PT1 <- Y[current_point+evolve,]
  PT2 <- Y[current_point_pair+evolve,]
  Vcurr <- PT1 - PT2

  # Angle between evolved pair vector and all other vectors
  # SLT: Added as.numeric to get rid of matrix structure. Good idea?
  cosTheta <- abs(as.numeric((Vcurr * t(Vnew))) / (Ydisti*end_dist))
  theta <- acos(cosTheta)

  # Search for next point
  # SLT: Not sure why this is broken into a separate function, but keeping
  # their structure for now.
  next_point <- 0
  while(next_point==0){
    list.output <-
      find_next_point(flag,theta,Ydisti,SCALEMX,SCALEMN,ZMULT,ANGLMX)
    next_point <- list.output$next_point
  }

  list.output
}


LyE_Knarr_Mod <- function(X, Fs, tau, dim, evolve,
                          # SLT: Matlab needs min(min()), but R is just min()
                          SCALEMX = (max(X)-min(X))/10,
                          SCALEMN = 10^(-4),
                          ANGLMX = 30*pi/180,
                          ZMULT = 1){
  # Setup

  DT <- 1/Fs
```

```
ITS <- 0
distSUM <- 0

# Create embedding if it hasn't been passed in already

if(ncol(X) == 1 || is.null(ncol(X))){
  Y <- psr_deneme(X, dim, tau)
  NPT <- length(X) - (dim-1)*tau - evolve # Size of useable data
  Y <- Y[1:(NPT+evolve),]
} else {
  Y <- X
  NPT <- length(Y) - evolve
}

# Start analysis

#out <- zeros(floor(NPT/evolve),9);
# SLT: Original code doesn't seem to always work (index sometimes exceeds
# size of out, as defined), but I think it's saved by
# the fact that Matlab will add a row if need be. So, instead we'll define
# the looping sequence first, make a matrix that matches that, then go from
# there.
loop.seq <- seq(1, NPT, by=evolve)
out <- matrix(rep(0, length(loop.seq) * 9), ncol=9)
colnames(out) <- c(
  "ITS", "current_point", "current_point_pair", "start_dist", "end_dist",
  "LyE", "OUTMX", "thbest*180/pi", "ANGLMX*180/pi"
  )
thbest <- 0
OUTMX <- SCALEMX

for(i in loop.seq){
  current_point <- i

  # Find first pair
  if(current_point==1){
      # Distance from current point to all other points
      Yinit = matrix(rep(Y[current_point,], NPT), byrow=TRUE, nrow=NPT)
      Ydiff = (Yinit - Y[1:NPT,])^2
      Ydisti = sqrt(rowSums(Ydiff))

      # Exclude points too close on path and closer in distance than noise
      # (SCALEMN)
      range_exclude = (current_point-10):(current_point+10)
```

```
    range_exclude = range_exclude[range_exclude > 1 & range_exclude <= NPT]
    Ydisti[Ydisti<SCALEMN] = 1e5; # SLT: This is meant to make dist big to
    Ydisti[range_exclude] = 1e5;  #      exclude pts. Better way?

    # find minimum distance point for first pair
    current_point_pair = which.min(Ydisti);
}

# Calculate starting and evolved distance

start_dist <- sqrt(sum((Y[current_point,] - Y[current_point_pair,])^2))
end_dist <- sqrt(sum((Y[current_point+evolve,] -
                    Y[current_point_pair+evolve,])^2))

if(end_dist != 0 && start_dist != 0){
    # Calculate total distance so far
    distSUM <- distSUM +
               # SLT: Don't know why log2 is used here.
               #
               log2(end_dist/start_dist)/(evolve*DT) # DT is sampling rate?!
               #log(end_dist/start_dist)/(evolve*DT) # DT is sampling rate?!
    ITS <- ITS + 1  # count iterations
    LyE <- distSUM/ITS; # max Lyapunov exponent
}

out[floor(i/evolve)+1,] <- c(
  ITS, current_point, current_point_pair, start_dist, end_dist, LyE,
  OUTMX, thbest*180/pi, ANGLMX*180/pi
)

ZMULT <- 1

if(end_dist < SCALEMX){
  current_point_pair <- current_point_pair+evolve
  if(current_point_pair > NPT){
    current_point_pair <- current_point_pair-evolve
    flag <- 1
    list.output <- GetNextPoint(flag, Y, current_point, current_point_pair,
      NPT, evolve, SCALEMX, SCALEMN, ZMULT, ANGLMX)
    current_point_pair <- list.output$next_point
    ZMULT <- list.output$ZMULT
    ANGLMX <- list.output$ANGLMX
    thbest <- list.output$thbest
    OUTMX <- list.output$SCALEMX
```

```
      }
      next
    }

    # find point pairing for next iteration
    flag <- 0;
    list.output <- GetNextPoint(flag, Y, current_point, current_point_pair,
      NPT, evolve, SCALEMX, SCALEMN, ZMULT, ANGLMX)
    current_point_pair <- list.output$next_point
    ZMULT <- list.output$ZMULT
    ANGLMX <- list.output$ANGLMX
    thbest <- list.output$thbest
    OUTMX <- list.output$SCALEMX
  }
list(out=out, LyE=LyE)
}
```