A Practical Guide to Function-Valued Traits

Anika Watson

2018-03-12

Introduction

Throwing Out The Data With The Bathwater

Despite our best attempts to conduct thorough studies, many biologists are unwittingly throwing away valuable data on a regular basis. In the coming years, as technology improves and data are produced in greater and greater quantities this is likely to evolve into a much larger issue (Stinchcombe et al., 2012). One might expect that these productive technological advances will bring with them incredible computing power eliminating the need to worry about missing important features. This, however, is not the case as even the "smartest" computer needs to know what to look for. Thus, the question of how to distill informative trends out of the data without losing valuable information or making unfounded assumptions, remains. Fortunately, we biologists do not need to answer this question alone. A promising approach has emerged from the field of statistics that hinges on the fact that many biological traits are traditionally described by single measurements when they are, in fact, functions.

To explore what we mean by this, let us consider light absorbance data. Take, for example, stickleback fish who have four light-sensitive proteins, called opsins, in their eyes that allow them to see. Each of the four opsins can absorb a particular spectrum of light, and when biologists wish to compare these spectra we often distill them down to a single number, such as the peak wavelength (the wavelength of light that the opsin absorbs best) (eg. Veen et al., 2016). Figure 1 shows the absorbance spectrum of two stickleback opsins.

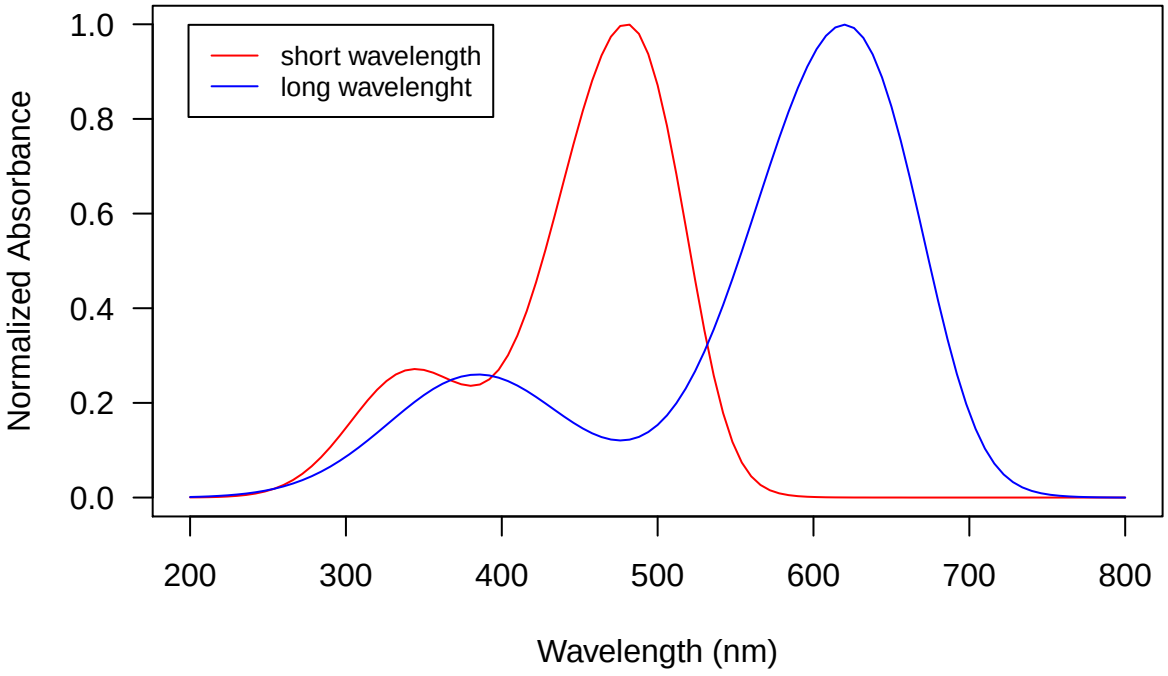

Figure 1: Spectral sensitivity curves of two of the opsins found in stickleback retina.

In the literature, these opsins are described by the wavelength at peak absorbance as λ_{max} = 434-441 nm, and λ_{max} = 566-638 nm, however they clearly absorb light well outside of these ranges. Simplifying these

spectra to their maximum frequency clearly eliminates much of the data, but this is not inherently a bad thing. The problem arises if we are throwing away *informative* data. In this case, we are. Treating these spectra as functions would allow us to compare them to other functions such as the spectrum of ambient light present in the fish's environment, and by simplifying the curves we forfeit the ability to make such comparisons.

If the idea of treating biological traits as functions, and comparing them to each other sounds daunting to you, you are not alone. The perceived mathematical sophistication and difficulty of this method is one of the main reasons that biologists have largely been hesitant to adopt it. Fortunately, even simple applications of it can have significant advantages over traditional analyses (Griswold et al., 2008; Baker et al., 2015). This entry-level guide aims to act as a starting point for biologists with some experience in R and a basic understanding of statistics, who wish to explore this functional new terrain. The paper is driven by detailed examples including thorough R code in order to be as accessible as possible. In this vain, some of the examples are clearest when the reader can interact with them in a way that PDF documents do not permit. For the interavtive version of this document please visit: https://fvt-analysis.shinyapps.io/fellowshipfinalreport/

History

How It All Started

This approach first appeared in literature as far back as 1989 when Mark Kirkpatrick and David Lofsvold noticed that many evolutionarily important traits could be described as functions and suggested that treating these traits as functions could provide biological insights (Kirkpatrick & Lofsvol, 1989). This observation caught the eyes of breeders, biologists and statisticians, gaining support from various fields as scientists realized just how widespread these "function-valued" traits are. According to a paper by the Function-valued Traits Working Group, "A function-valued trait is any trait that changes in response to another variable" (Stinchcombe et al., 2012), so given that this includes any traits that change with respect to time, gene expression, or environmental conditions there is no shortage of examples. Size can be thought of as a function of age, fur length, as a function of ambient temperature, and so on. In a 1998 article, geneticist and statistician W.G. Hill predicted that function-valued methods may eventually replace traditional methods in evolutionary biology (Griswold et al., 2008). The term "function-valued traits" was coined the very next year (Pletcher and Geyer, 1999).

Examples

Time To Venture Into The Thick Of It

As daunting as it may seem, function-valued trait analysis may be more familiar than it seems. Have you ever hit "add trendline" and "display equation on chart" in excel? If yes, then you've already done some simple function-valued analysis. Figure 2 is an example of just that, showing data from a university freshman level biology lab (designed for non-biology majors) in which students measure the volume of CO_2 produced through yeast metabolism, and use this data to calculate the rate of metabolism. The premise of this lab is, at its foundation, the simplest example of function-valued analysis.

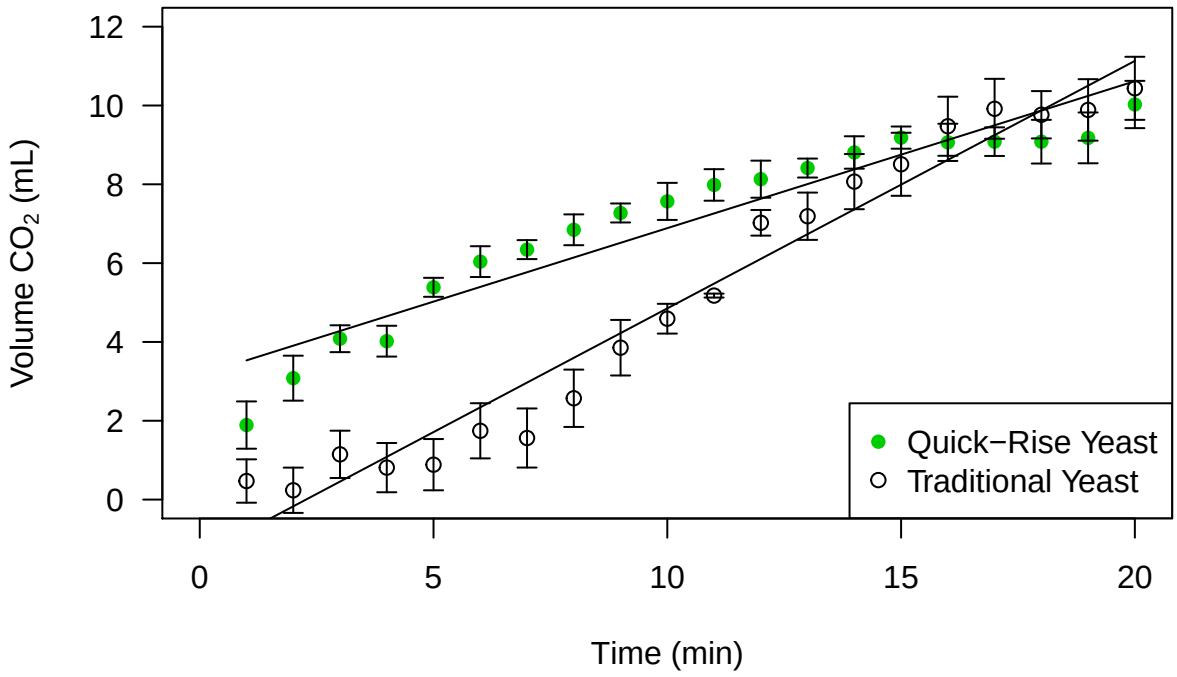

Figure 2: A plot of yeast metabolism data representing the volume of CO_2 produced over time. Error bars represent standard error from three trials.

It is clear from the graph, however, that these stright lines of best fit do not represent the data very well, but without knowledge of the underlying function (is the growth exponential? logarithmic?), we may not want to guess at a more complicated alternative. Fortunately we don't have to. We may, instead, use a non-parametric method of function-valued traits that does not assume any particular underlying function.

In order to clearly illustrate this method, we have created a sample dataset comprised of sample lengths of the mythical species of fish, the R-ctic grayling. This dataset has both random individual variance, and random measurement error and is composed of two populations, fluvial R-ctic grayling (those that live exclusively in rivers), and adfluvial R-ctic grayning (those that live in rivers and lakes). See Figure 3 for a plot of the fluvial R-ctic grayling data.

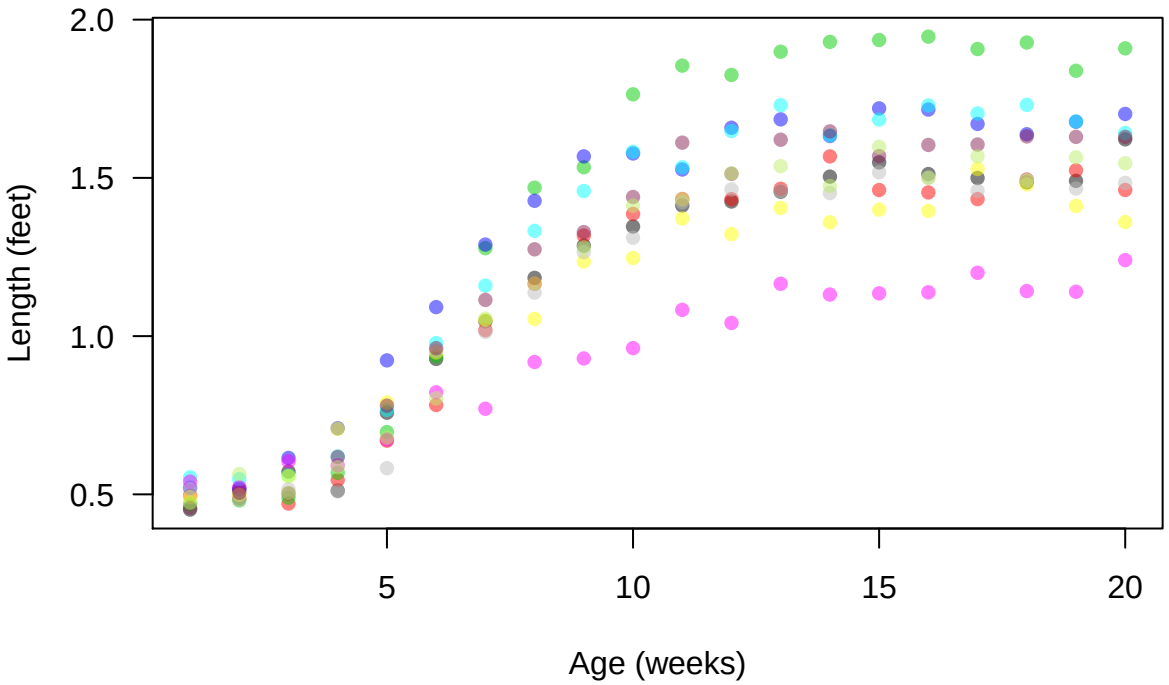

Figure 3: A plot of the ten R-ctic Grayling sampled from population 1 (fluvial).

Now that we have data we can fit various models to them. Let's begin by fitting a series of polynomials to the data in a process called polynmial regression. For simplicity, we will begin by focusing on one individual, then we will scale up to one population, and finally, we will compare two populations of R-ctic Grayling.

We can name the first individual from the first population, `RcticG1ind1` so that it will be easy to keep track of. See Figure 4 for `RcticG1ind1`'s length data.

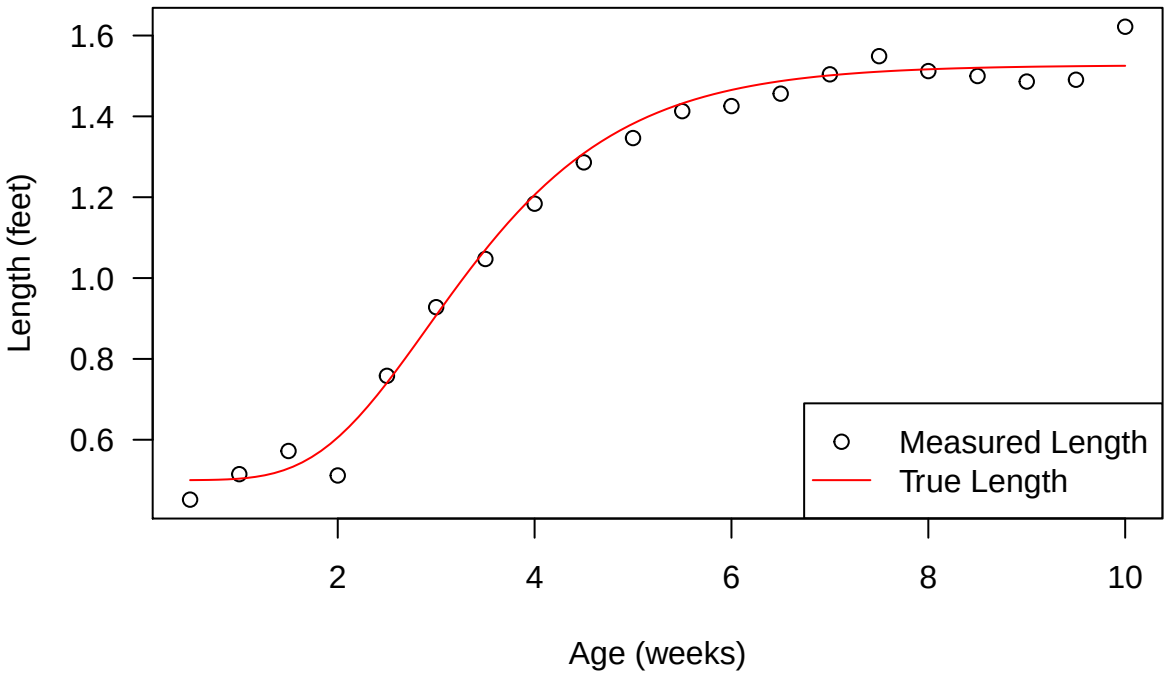

Figure 4: A plot of the true and measured length of `RcticG1ind1`.

In order to fit our polynomial model to `RcticG1ind1` we will use the linear models function in R. Unlike its name suggests, the "linear model" function in R can do far more than merely slap on the closest straight line approximation! We can get it to fit a polynomial of higher order by specifying the highest exponent when we make the model.

```
# First we need to define the x-values, In this case they
# represent the points in time when scientists measured the
# lengths of the fish.
xvalues <- seq(from = 0.5, to = 10, by = 0.5)  #This object will come up often so take note!
# Now let's create an object and assign the model to it. Note
# that we are inputting the y-values (RcticG1ind1), the
# x-values (xvalues), and the highest order to use in the
# polynomial (n = 6)
PolyModelPop1ind1 <- lm(RcticG1ind1 ~ poly(x = xvalues, n = 6))
```

Once we have a model, we can use it to calculate the predicted 99% confidence intervals.

```
# Here we input our model (PolyModelPop1ind1), the x-values
# as a data frame (data.frame(x = xvalues)), the type of
# interval we would like ('confidence'), and the level of the
# interval (99%)
predicted.intervals <- predict(PolyModelPop1ind1, data.frame(x = xvalues),
    interval = "confidence", level = 0.99)
```

As a relatively new field, function-valued analysis does not yet have a universal method of quantitatively assessing model fit, so we rely heavily on visual inspection.

Thus, it is important to plot the data (Figure 5).

```
# Start by plotting the data that the scientists cathered in
# the field.
plot.new(y = RcticG1ind1, x = xvalues, main = "Polynomial Regression",
    ylab = "Length (feet)", xlab = "Age (weeks)", las = 1)

# Then add the function of the true length of RcticG1ind1
# that was used to generate the data.
curve(RcticG(Mono1akg[1, 1], Mono1akg[1, 2], Mono1akg[1, 3],
    x), n = 101, add = TRUE, col = "red")

# And we're ready to plot the approximated polynomial
# function:

# This line will be the approximating function.
lines(inp, predicted.intervals[, 1], col = "green", lwd = 3)

# And these two will be the confidence intervals.
lines(inp, predicted.intervals[, 2], col = 4, lwd = 1, lty = 2)
lines(inp, predicted.intervals[, 3], col = 4, lwd = 1, lty = 2)

# Add a legend and we're good to go.
legend("bottomright", c("Measured Length", "True Length", "Predicted Length",
    "99% Confidence Interval"), col = c("black", "red", "green",
    4), pch = c(1, NA, NA, NA), lwd = c(NA, 1, 3, 1), lty = c(1,
    1, 1, 2))
```

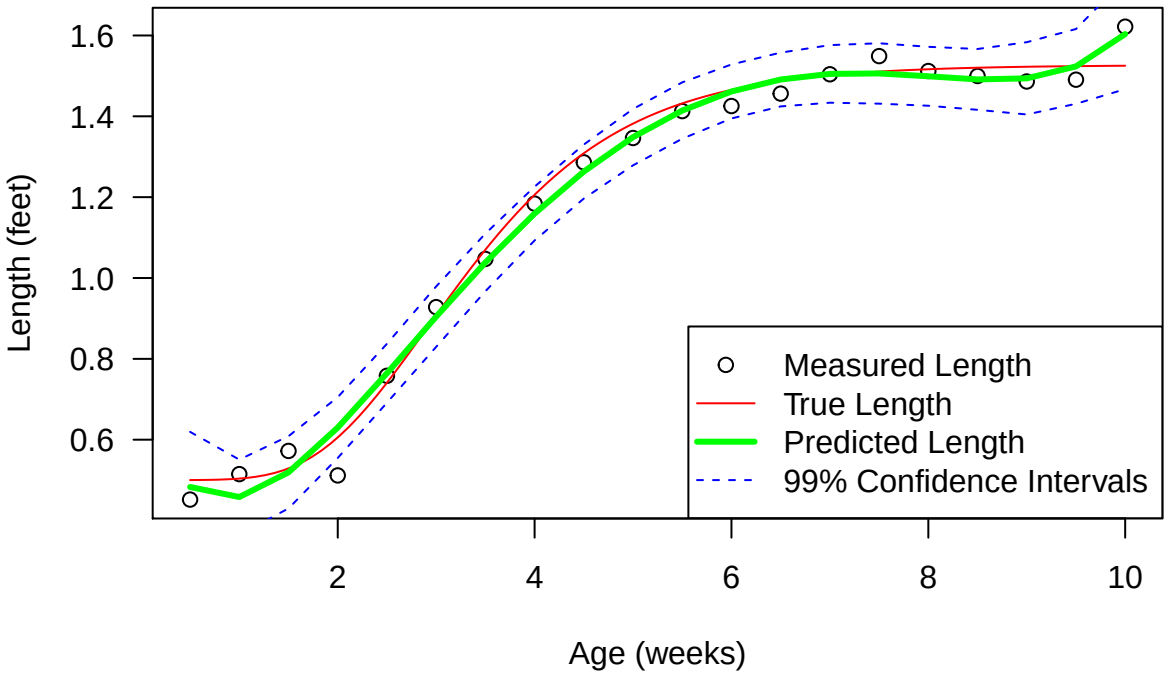

Figure 5: The polynomial fit as compares to the real function and the measured data.

This thick green line that represents our approximation seems, intuitively, to be a reasonable fit. But what if we how did we know to choose n=6? Did that arbitrary decision impact the model? Let's take it to the max, n = 19, and see what happens.

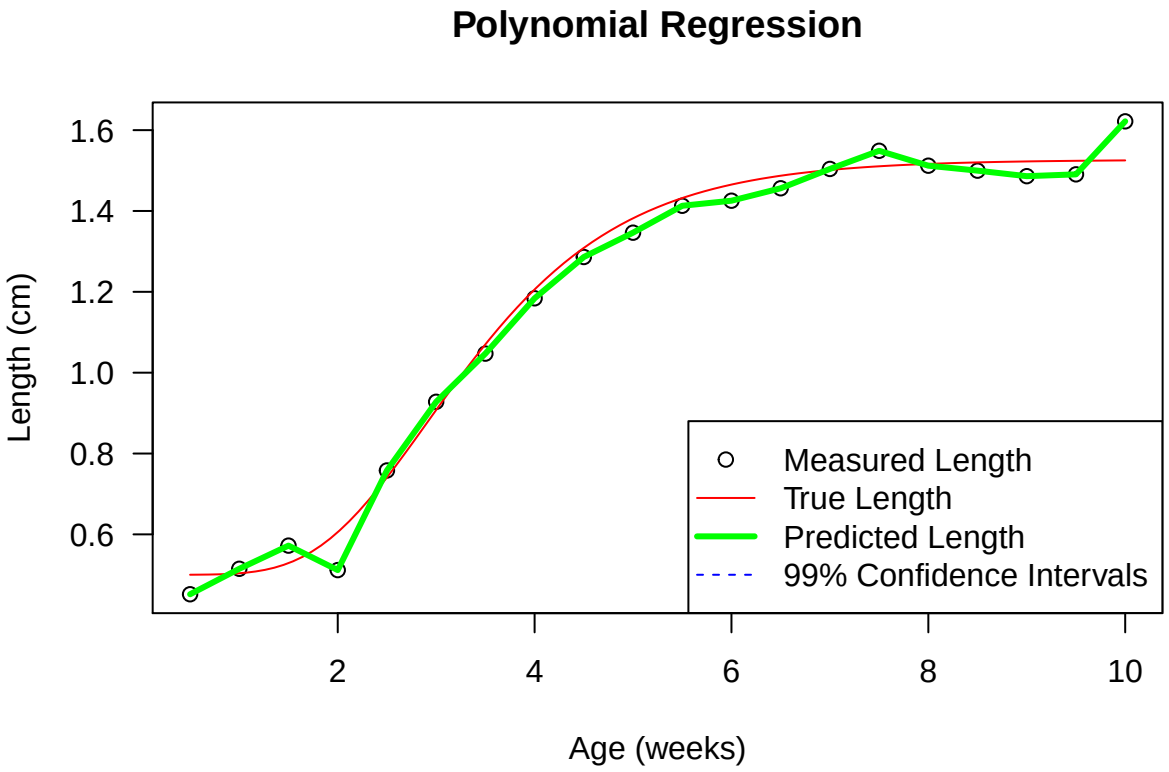

Figure 6: A plot demonstrating the shape of the polynimial fit when the number of terms in the approximating polynomial equals the number of data points.

Bigger isn't always better. Note what happpens when n = 19 (see interactive figure). Here the approximate function has been overparametrized to the extreme and has lost all predictive value. The best-looking choices for n are between 4 and 6. This is consistent with the choices made in the literature (Griswold et al., 2008).

Under The Hood

Ok so now we've got a model, but what is R actually doing? Let's see what is in this mysterious object "`PolyModelPop1ind1`".

```
# Let's call our model to see what's inside.
PolyModelPop1ind1
```

```
##
## Call:
## lm(formula = RcticG1ind1 ~ poly(x = inp, n = 19))
##
## Coefficients:
##              (Intercept)   poly(x = inp, n = 19)1   poly(x = inp, n = 19)2
##                 1.177894                 1.634426                -0.582093
##  poly(x = inp, n = 19)3   poly(x = inp, n = 19)4   poly(x = inp, n = 19)5
##                -0.073148                 0.265193                -0.028388
##  poly(x = inp, n = 19)6   poly(x = inp, n = 19)7   poly(x = inp, n = 19)8
##                 0.030072                 0.118270                -0.053117
##  poly(x = inp, n = 19)9  poly(x = inp, n = 19)10  poly(x = inp, n = 19)11
##                 0.014669                -0.003477                -0.019778
## poly(x = inp, n = 19)12  poly(x = inp, n = 19)13  poly(x = inp, n = 19)14
```

```
##                0.076191                -0.050519                 0.033801
## poly(x = inp, n = 19)15 poly(x = inp, n = 19)16 poly(x = inp, n = 19)17
##               -0.041840                -0.003852                 0.004038
## poly(x = inp, n = 19)18 poly(x = inp, n = 19)19
##               -0.006460                -0.010953
```

Evaluating `PolyModelPop1ind1` we see that it is composed of seven `Coefficients`. Under the hood, R is adding together functions we told it to combine, namely, a costant, x, x^2, x^3, etc. up to x^6. These `Coeffficients` correspond to the amplitudes of the functions making up the green curve shown in the figures above. In other words they tell us how much of each function is being added. Figure 8 is a visualization of the functions present when n = 6.

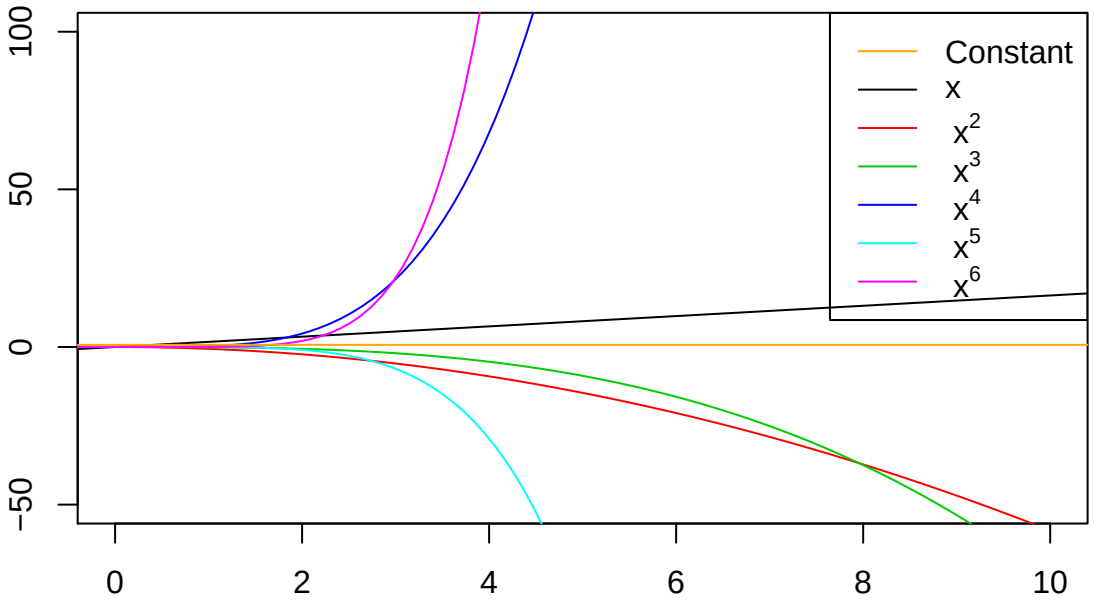

Figure 7: A plot of the functions making up the polynomial approximation of `RcticG1ind1`'s growth function.

Many of these functions get very big, very quickly, when compared to the function that they add up to (which never exceeded y = 2) but when added together they make up a fair approximation of the length of `RcticG1ind1`.

Analysis

Now that we have gotten R to approximate a model function, we are ready to broaden our scope from our friend 'RcticG1ind1' and consider the entire sample. By repeating the process outlined above for all of the samples of the lengths of fluvial R-ctic Grayling, we can collect a dataset of coefficients.

First, let's look at the data to see how it is organized.

```
# To start let's take a look at the data frame of the sample
# lengths. (The following line of code makes R report the
# structure of the object (RcticG1))
str(RcticG1)
```

```
## 'data.frame':    10 obs. of  20 variables:
##  $ X1 : num  0.452 0.496 0.468 0.521 0.554 ...
##  $ X2 : num  0.515 0.487 0.481 0.519 0.548 ...
##  $ X3 : num  0.572 0.471 0.489 0.615 0.559 ...
##  $ X4 : num  0.512 0.545 0.568 0.709 0.62 ...
```

```
##  $ X5 : num  0.758 0.67 0.697 0.923 0.767 ...
##  $ X6 : num  0.928 0.782 0.936 1.092 0.978 ...
##  $ X7 : num  1.05 1.02 1.28 1.29 1.16 ...
##  $ X8 : num  1.18 1.17 1.47 1.43 1.33 ...
##  $ X9 : num  1.29 1.32 1.53 1.57 1.46 ...
##  $ X10: num  1.35 1.39 1.76 1.58 1.58 ...
##  $ X11: num  1.41 1.43 1.85 1.53 1.53 ...
##  $ X12: num  1.43 1.43 1.83 1.66 1.65 ...
##  $ X13: num  1.46 1.47 1.9 1.68 1.73 ...
##  $ X14: num  1.5 1.57 1.93 1.63 1.63 ...
##  $ X15: num  1.55 1.46 1.94 1.72 1.68 ...
##  $ X16: num  1.51 1.45 1.95 1.72 1.73 ...
##  $ X17: num  1.5 1.43 1.91 1.67 1.7 ...
##  $ X18: num  1.49 1.49 1.93 1.64 1.73 ...
##  $ X19: num  1.49 1.52 1.84 1.68 1.68 ...
##  $ X20: num  1.62 1.46 1.91 1.7 1.64 ...
```

This tells us that we have a data frame of "10 observations of 20 variables" which in this case is "10 fish being measured 20 times each".

```
# Now, let's make an empty data frame for the coefficients.
# We know that we have measurements from 10 fish, and that
# each fish will have 7 coefficients approximating its growth
# curve (since we chose n = 6). Thus we need a 7 X 10 matrix,
# which we fill with NA's.
PolyModel1 <- data.frame(matrix(data = NA, nrow = 7, ncol = 10))

# Next, we can fill the empty data frame with the
# coefficients from the RcticG1 sample. This 'for loop'
# repeats the command 10 times, first with l = 1, then with l
# = 2, etc. Each time the loop assigns a different element of
# the model 'lm(as.matrix(RcticG1)' to the data frame
# 'PolyModel1'.
for (l in 1:10) {
    PolyModel1[l] <- coefficients(lm(as.matrix(RcticG1)[l, ] ~
        poly(x = xvalues, n = 6)))
}

# Let's take a look at what we've got by using the 'str()'
# command again.
str(PolyModel1)
```

```
## 'data.frame':    7 obs. of  10 variables:
##  $ X1 : num  1.1779 1.6344 -0.5821 -0.0731 0.2652 ...
##  $ X2 : num  1.153 1.616 -0.65 -0.187 0.341 ...
##  $ X3 : num  1.433 2.324 -0.93 -0.295 0.42 ...
##  $ X4 : num  1.34329 1.71138 -0.80547 -0.00662 0.25978 ...
##  $ X5 : num  1.314 1.829 -0.732 -0.215 0.289 ...
##  $ X6 : num  0.9376 1.0068 -0.2493 -0.0751 0.1075 ...
##  $ X7 : num  1.1307 1.3876 -0.5496 -0.0801 0.0743 ...
##  $ X8 : num  1.15 1.603 -0.611 -0.197 0.301 ...
##  $ X9 : num  1.252 1.781 -0.707 -0.114 0.273 ...
##  $ X10: num  1.188 1.68 -0.611 -0.197 0.292 ...
```

Here we see that our data frame of coefficients is much more concise than the original data frame of lengths, but this does not mean that information is lost.

(Note that the rows and colums have switched. Where X10 marked the 10th measurement in the data frame `RcticG1`, it now marks the 10th individual in `PolyModel1`).

Now let's run an analysis between this sample, and a sample of the adfluvial R-ctic Grayling.

```
# Once again let's make an empty data frame for our
# coefficients,
PolyModel2 <- data.frame(matrix(data = NA, nrow = 7, ncol = 10))

# and fill the empty data frame with the coefficients from
# the RcticG2 sample.
for (l in 1:10) {
    PolyModel2[l] <- coefficients(lm(as.matrix(RcticG2)[l, ] ~
        poly(x = xvalues, n = 6)))
}

# Let's check this to make sure it worked
str(PolyModel2)
```

```
## 'data.frame':    7 obs. of  10 variables:
##  $ X1 : num  1.5028 2.0891 -0.9193 0.0595 0.2867 ...
##  $ X2 : num  1.4788 2.0957 -1.0386 -0.0302 0.3982 ...
##  $ X3 : num  1.769 2.764 -1.314 -0.122 0.45 ...
##  $ X4 : num  1.691 2.072 -1.144 0.198 0.214 ...
##  $ X5 : num  1.6419 2.2745 -1.0742 -0.0761 0.3094 ...
##  $ X6 : num  1.2032 1.474 -0.4138 -0.0659 0.1262 ...
##  $ X7 : num  1.4415 1.8402 -0.8175 -0.0035 0.0922 ...
##  $ X8 : num  1.4719 2.0843 -0.9692 -0.0654 0.347 ...
##  $ X9 : num  1.5792 2.2306 -1.0483 0.0238 0.2943 ...
##  $ X10: num  1.5053 2.1818 -0.9753 -0.0802 0.3596 ...
```

At this stage it is difficult to see intuitively whether or not these data frames of coefficients differ significantly. Fortunately, we don't need fancy function-valued tools at this point and can simply use t-test to compare our samples. Now we are ready to run the test!

```
# t-test comparing the two data frames of polynomial model
# coefficients.
t.test(PolyModel1, PolyModel2)
```

```
##
##  Welch Two Sample t-test
##
## data:  PolyModel1 and PolyModel2
## t = -0.54188, df = 130.04, p-value = 0.5888
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3817156  0.2175695
## sample estimates:
## mean of x mean of y
## 0.3192654 0.4013385
```

According to this result, with a p-value of 0.5888 we have failed to reject the null hypothesis that the sizes of fluvial and adfluvial populations of R-ctic Grayling differ significantly.

Before we accept this result, let us plot these two samples together to see whether or not this is a reasonable outocome Figure 9.

```
matplot(y = t(RcticG1), type = 'p', lty = 1, pch = 10, ylab = "Length (feet)", xlab = "Age (weeks)", las
matplot(y = t(RcticG2), type = 'p', lty = 1, pch = 1, las = 1, col = 2, add = TRUE)
legend("bottomright",c("Males", "Females"), col = c("black", "red"), pch = c(10, 1))
```

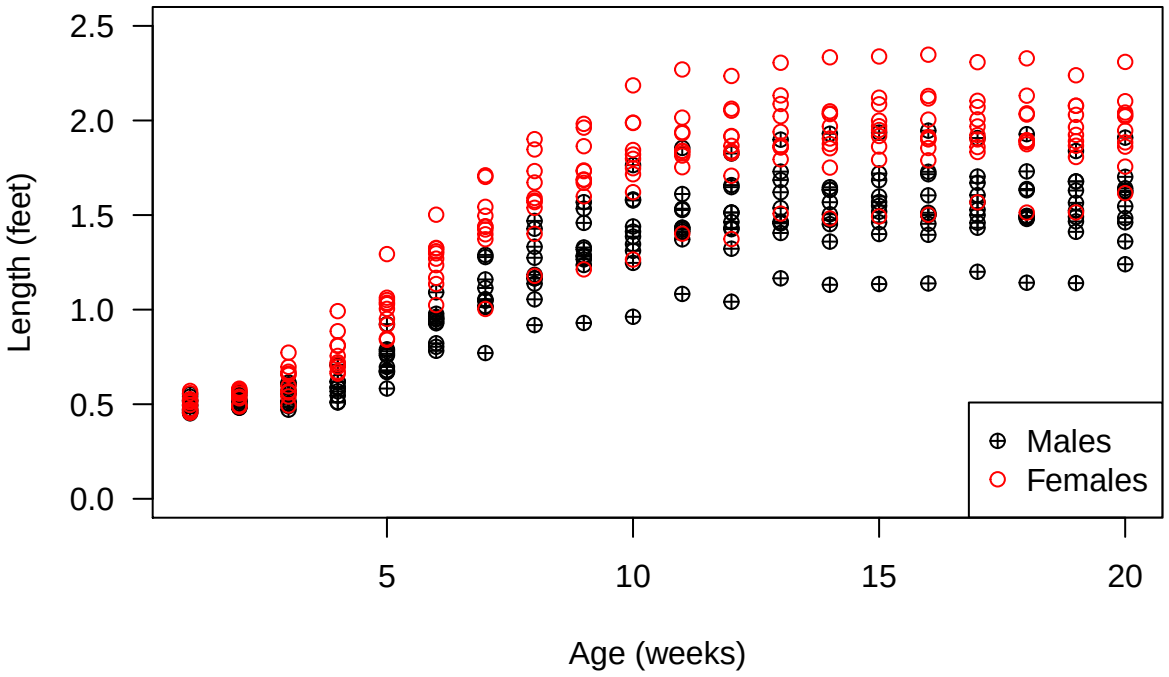

Figure 8: A comparison of male and female R-ctic Grayling lengths

There is a significant amount of overlap between these measurements of males and females which supports our calculation that we cannot reject the null hypothesis.

Fitting a Sinusoidal Function to Data

Okay so we fit a polynomial to the data, but is that really the best fit? In Excel the user can choose from a selection of fits to suit their data. Let's not be outdone by a computer program! Here we will go beyond the capabilities of Excel and fit a sinusoidal model to the data.

```
# To begin, we can use the function `spectrum` to get the
# spectral density of `RcticG1ind1`. This will tell us the
# prominence of various frequencie in the data.  If there are
# consistently peaks at a set interval, then the frequency of
# that pattern (1/(the period)) will feature prominently in
# this spectrum as an abundant frequency.  We want to know
# the distance between the peaks of the most prominent
# pattern in our data.

# To do that, we assign the 'spectrum' of our data frame to
# an object.
spctrm <- spectrum(RcticG1ind1)
```

```
# Then we take the frequency ('$freq') that corresponds to
# the maximum ('max()') spectral density ($spec). We take
# 1/(frequency corresponding to maximum spectral density) in
```

```r
# order to get the associated period.
period <- 1/spctrm$freq[spctrm$spec == max(spctrm$spec)]

# Now we're ready to make a model.  Even though we are now
# even further from our classical 'linear' model than we were
# with the polynomial approximation, we are going to use the
# linear model function once again. Here, again we are
# inputting our y-values (RcticG1ind1), but this time we also
# have to specify the basis function.
SinModelPop1ind1 <- lm(RcticG1ind1 ~ sin(2 * pi/period * xvalues) +
    cos(2 * pi/period * xvalues))

# If you find the mathematics of this basis function
# daunting, go to: https://www.desmos.com/calculator and copy
# this into the input window, adjusting the period, and
# changing between sin and cos as you see fit:
# \sin\left(\frac{\left(2\cdot\pi\right)}{period}\cdot
# x\right)

# As before, let's find the predicted intervals, and assign
# them to an object.
PredictedIntervalsSin <- predict(SinModelPop1ind1, data.frame(x = xvalues),
    interval = "confidence", level = 0.99)
```

```r
# Now we're ready to plot our model.  Once again, we can
# start by plotting the data and true function as before.
plot(y = RcticG1ind1, x = xvalues, main = "Polynomial Regression",
    ylab = "Length (feet)", xlab = "Age (weeks)", las = 1)
# Then add the true underlying function.
curve(RcticG(Mono1akg[1, 1], Mono1akg[1, 2], Mono1akg[1, 3],
    x), n = 101, add = TRUE, col = "red")

# Add the approximate function,
lines(fitted(SinModelPop1ind1) ~ xvalues, col = "green", lwd = 3)

# the 99% confidence intervals,
lines(xvalues, PredictedIntervalsSin[, 2], col = 4, lwd = 1,
    lty = 2)
lines(xvalues, PredictedIntervalsSin[, 3], col = 4, lwd = 1,
    lty = 2)

# and a legend.
legend("bottomright", c("Measured Length", "True length", "Predicted length",
    "99% Confidence Interval"), col = c("black", "red", "green",
    4), pch = c(1, NA, NA, NA), lwd = c(NA, 1, 3, 1), lty = c(1,
    1, 1, 2))
```

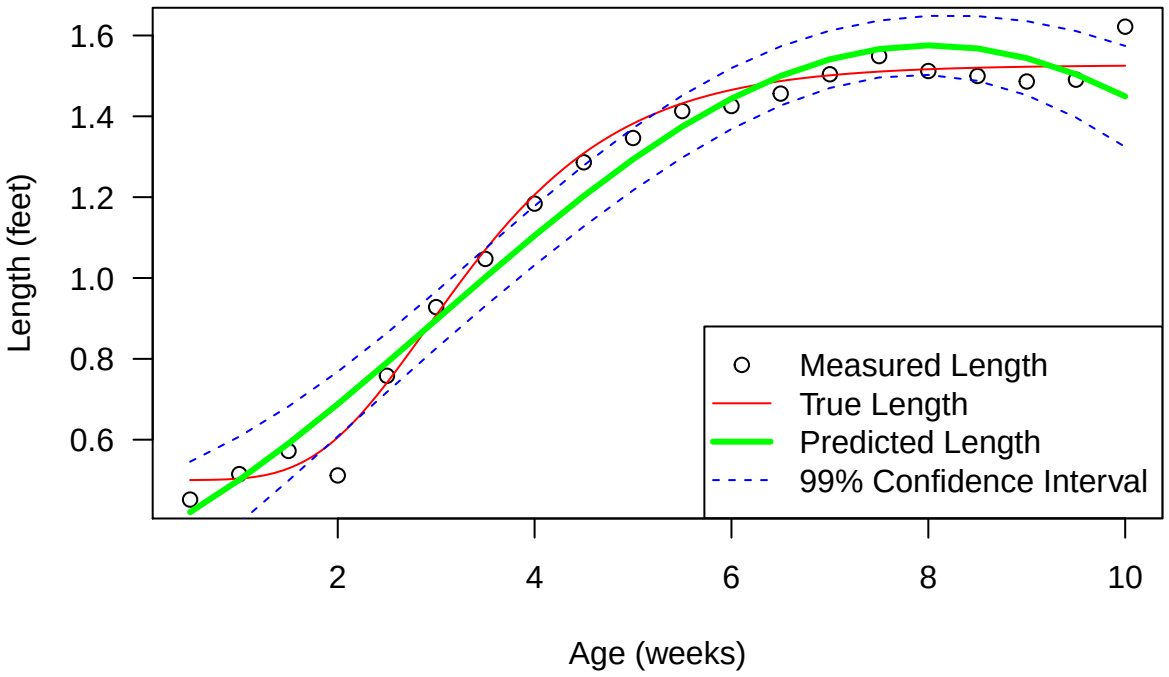

Figure 9: Cosine decomposition

In Figure 10, we see that we do not have as good of a fit as we did in Figure 6. Our sinusoidal model is comparable to about a third order polynomial (n = 3). Let's see if we can improve the fit by adding a second harmonic (i.e. adding terms with double the frequency).

```
# The only difference here is that we have added another sine
# term, and another cosine term, each of which have been
# multiplied by two inside of the brackets.
SinModel2Pop1ind1 <- lm(RcticG1ind1 ~ sin(2 * pi/period * xvalues) +
    cos(2 * pi/period * xvalues) + sin(4 * pi/period * xvalues) +
    cos(4 * pi/period * xvalues))

# To gain an intuition for what this does to the function,
# return to https://www.desmos.com/calculator and copy the
# same expression into the input window, this time, try
# multiplying the expression within the 'sin' or 'cos'
# brackets and see what happens:
# \sin\left(\frac{\left(2\cdot\pi\right)}{period}\cdot
# x\right)

# Now again we find the predicted intervals, and assign them
# to an object.
PredictedIntervalsSin2 <- predict(SinModel2Pop1ind1, data.frame(x = xvalues),
    interval = "confidence", level = 0.99)

# Now let's plot this and see how our approximation improves.
# Start with the data.
plot(y = RcticG1ind1, x = xvalues, main = NULL, ylab = "Length (feet)",
    xlab = "Age (weeks)", las = 1)
# Add the true curve.
curve(RcticG(Mono1akg[1, 1], Mono1akg[1, 2], Mono1akg[1, 3],
```

```
    x), n = 101, add = TRUE, col = "red")
# Add the old approximation for comparison.
lines(fitted(SinModelPop1ind1) ~ xvalues, col = "black", lwd = 1)

# Add the new approximation.
lines(fitted(SinModel2Pop1ind1) ~ xvalues, col = "green", lwd = 3)
# Add the new confidence intervals.
lines(xvalues, PredictedIntervalsSin2[, 2], col = 4, lwd = 1,
    lty = 2)
lines(xvalues, PredictedIntervalsSin2[, 3], col = 4, lwd = 1,
    lty = 2)
# And finally add a legend.
legend("bottomright", c("Measured Length", "True Length", "First Harmonic",
    "Second Harmonic", "99% Confidence Interval"), col = c("black",
    "red", "black", "green", 4), pch = c(1, NA, NA, NA, NA),
    lwd = c(NA, 1, 1, 3, 1), lty = c(1, 1, 1, 1, 2))
```

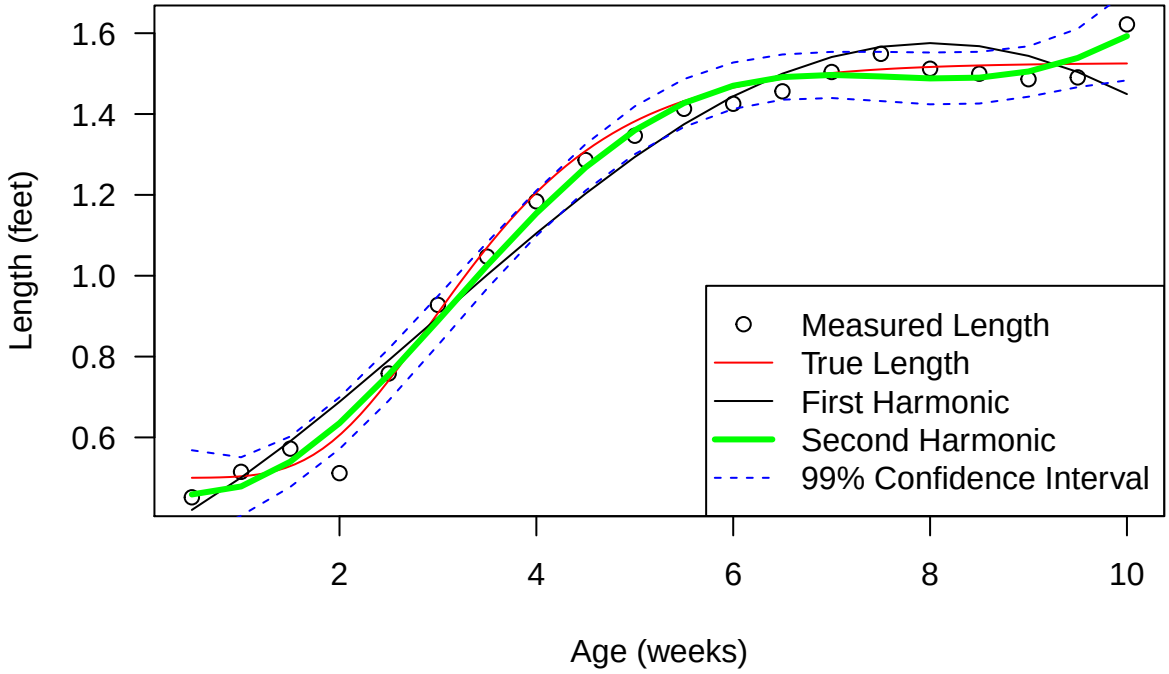

Figure 11 demonstrates that adding the second harmonic greatly improved the fit. Having said this, a rigorous assesment of which basis function is best is beyond the scope of this introductory paper.

Under the Hood

Now just as we did with the polynomial regression we can evaluate this model and take a look at the inner workings.

```
# Let's call the model to take a look what it contains.
SinModel2Pop1ind1
```

Even though we are no longer dealing with polynomials, this call still returns `Coefficients`. Just as before these are the amplitudes of various functions making up the model.

One major difference between sinusoidal functions and polynomials is that sinusoidal functions oscillate ad infinitum, whereas polynomials shoot off to infinity in one way or another (assuming non-zero coefficients).

This may be a useful point to consider when selecting a basis function. Having said that, we have just seen that both functions were able to reach visually similar levels of approximation. In fact there is little need to fret about selecting the "correct" basis function. After completing 3,000 analyses like that above, Griswolold et al. wrote, "even for basic questions about differences in mean functions, the function-valued approach never has lower, and often has substantially better, power than the multivariate approach." So don't worry about picking the perfect basis function, or exactly the right number of terms in your basis function. Treating traits as functions is just another way to ask and answer questions so don't be afraid to try it out.

Discussion

Pros And Cons Of This Method

Statistically, the function-valued perspective boasts "efficiency, flexibility, and enhanced power" (Stinchcombe et al., 2012). Of all the complements function-valued analysis receives, "enhanced power" seems to come up most. To unpack why function-valued analysis has increased power, recall that having greater "power" means that function-valued analyses are less likely to accept a false null hypothesis (Type II error) than multivariate approaches. To gain some intuition for this consider that a permutation randomly assigns each measured value to one of the subjects without knowing or caring about the order and spacing of the data points. As a result, some of the permutations will lie outside of the realm of physical possibility (an individual squirrel cannot, for instance, go from 15 cm in length, to 45 cm in length, and then down to 5 cm in length over the span of six days). Yet outlandish, impossible, permutations will still impact the analysis, perhaps making an observed variation between two populations appear negligible, when it is actually very dramatic within physical constraints. The function-valued approach *does* take into account the ordering and spacing of data because functions are sensitive to the locations of points along their curves. Another reason that function-valued analysis has increased power is that functions can pluck out trends in noisy data where traditional methods get lost in a sea of noise (Stinchcombe et al. 2012). Remarkably, Griswold et al. found that in all cases they examined, "the gain in power of the [...] function-valued approach over the multivariate approach did not appear to come at the cost of an elevated Type I error rate" (Griswold et al., 2008) meaning that function-valued analyses reduce false negatives without increasing false positives.

Another advantage of function-valued analysis is that it does not care which points the researcher selected to sample their data. One can just as easily approximate a function that fits the growth of a plant if its height is measured on odd days, as if its height were to be measured on even days. This allows for comparison between data sets with different numbers of samples, and whose samples were taken at different points, opening up worlds of possibilities to gain more insight into existing data by comparing or seemingly disparate datasets.

Despite these clear advantages, and Hill's prediction, function-valued trait analysis still hasn't taken off. A quick online search for "function-valued traits" returns 7 results between 1990 and 2000, 236 results between 2000 and 2010, and 370 since 2010 (2010-March 2018). Far from holding a monopoly over the thousands of evolutionary biology, papers published each year. W.G. Hill's prediction is far from becoming a reality. The "Function-valued Traits Working Group," listed two major limitations in 2012 that have yet to be thoroughly addressed. Firstly, the cost of additional measurements required to estimate functions presents a challenge, and secondly a general unfamiliarity with function-valued analyses is limiting its uptake by scientists (Stinchcombe et al., 2012). Other drawbacks of this method include the fact that error calculations for function-valued analyses are still poorly developed, and that there are no clear-cut rules for selecting a basis functions, or for determining how well the approximating function should fit the data (Griswold et al., 2008; Hernandez et al., 2015). Despite outstanding questions, the Function-valued Traits Working Group remains optimistic writing that, "although challenges remain, as inherently functional data become more common, functional analyses become necessary, rather than just a promising idea" (Stinchcombe et al. 2012).

Going Further

"Okay So What Now?"

In this paper, we have focused on one type of function-valued analysis, a non-parametric approach (meaning we did not assume the underlying function dictating the data) of fitting a basis function to data. We chose this approach because Griswold et al. showed that even when the correct underlying function was known, a parametric analysis fitting the curve that was used to produce the data to the data did not always have better power than the non-parametric approach (Griswold et al., 2008). Other methods include a parametric approach, which involves assuming an underlying function and estimating the parameters of that function (Hernandez et al., 2015). For example, one might assume that data fits a Gaussian function and take the mean and standard deviation as parameters (here "standard deviation" refers to the width of the bell approximating the data, not the standard deviation of the data). These parameters can then be compared to each other as data, just as the coefficients of basis functions can be compared in non-parametric function-valued analysis. The curious reader is encouraged to consult the list of references to further explore function-valued traits. This field holds many new techniques to be developed, fine-tuned, and implemented so dive in and be part of this wave of innovation.

Works Cited

Baker, R. L., Leong, W. F., Brock, M. T., Markelz, R. J. C., Covington, M. F., Devisetty, U. K., ... Weinig, C. (2015). Modeling development and quantitative trait mapping reveal independent genetic modules for leaf size and shape. The New Phytologist, 208(1), 257–268. https://doi.org/10.1111/nph.13509

Griswold, C. K., Gomulkiewicz, R., & Heckman, N. (2008). Hypothesis Testing in Comparative and Experimental Studies of Function-Valued Traits. Evolution, 62(5), 1229–1242. https://doi.org/10.1111/j.1558-5646.2008.00340.x

Hernandez, K. M. (2015), Understanding the genetic architecture of complex traits using the function-valued approach. New Phytol, 208: 1–3. doi:10.1111/nph.13607

Kirkpatrick, M., & Lofsvold, D. (1989). The evolution of growth trajectories and other complex quantitative characters. Genome, 31(2), 778–783. https://doi.org/10.1139/g89-137

Pletcher, S. D., & Geyer, C. J. (1999). The Genetic Analysis of Age-Dependent Traits: Modeling the Character Process. Genetics, 153(2), 825–835.

Stinchcombe, J. R. et al., (2012). Genetics and evolution of function-valued traits: understanding environmentally responsive phenotypes. Trends in Ecology & Evolution, 27(11), 637–647. https://doi.org/10.1016/j.tree.2012.07.002

Veen, T., Brock, C., Rennison, D., & Bolnick, D. (2017). Plasticity contributes to a fine-scale depth gradient in sticklebacks' visual system. Molecular Ecology, 26(16), 4339–4350. https://doi.org/10.1111/mec.14193