A Practical Guide to Function-Valued Traits

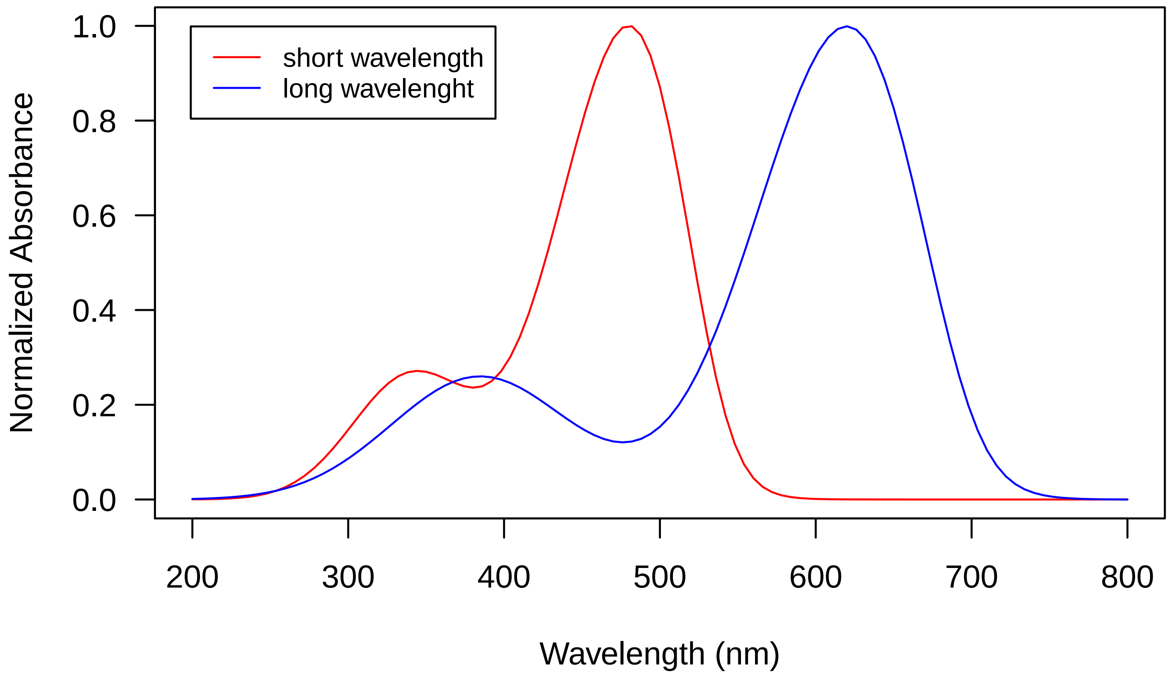
Anika Watson

2018-03-08

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. One major cause for this loss is that many biological traits are often described by single measurements when they are, in fact, too complex for such a simple framework. Consider, for instance, data for the spectrum of light absorbed by an eye (see Figure 1). These spectra are often distilled down to a single number, such as the peak wavelength (the wavelength of light that the eye is best tuned to see) (eg. Thor et al., 2016). This simplification ignores much of the data, and limits our ability to quantify the relationship between light wavelength absorbance, and, for instance, the spectrum of ambient light present. Treating these traits as functions, rather than single data points, has many statistical advantages including increased statistical power and others, which we will discuss shortly (Stinchcombe et al., 2012; Griswoldet al., 2008). Unfortunately, due to its perceived mathematical sophistication biologists have been hesitant to adopt function-valued methods. The good news is that even relatively simple function-valued analyses can provide valuable insight compared to traditional approaches (Griswold et al., 2008; Baker et al., 2015). This guide aims to serve as an introduction to function-valued trait analysis providing thorough examples including R code to dispel fear of this topic.



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 (Kirkpatrick & Lofsvol, 1989). This caught the eyes of breeders, biologists and statisticians gaining support from various fields as scientists realized just how widespread function-valued traits are. “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. In a 1998 article 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).

According to a paper by the “Function-valued Traits Working Group,” the function-valued perspective receives such praise because it “offers enhanced statistical power, greater ability to detect genetic constraints, and improved understanding of phenotypic and genetic variation in environmentally sensitive traits” compared to traditional multivariate methods (Stinchcombe & Kirkpatrick, 2012). These advantages come from the fact that fitting a function to data takes into account the ordering and spacing of the data where multivariable methods ignore this information (Griswold et al., 2008), and because fitting a curve to data smooths noise, reducing its effects on analysis (Stinchcombe et al., 2012). 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 fit a curve estimating 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.

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. 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 & Kirkpatrick, 2012). As a step towards solving the second challenge, this guide aims to serve as an introduction to function-valued traits, and function-valued trait analysis. This resource is designed for biologists with some experience in R and a basic understanding of statistics. We begin by diving headlong into examples and sorting through the theory as we go. At the end of this text there are recommended activities and example problems so that readers can acquaint themselves more fully with function-valued traits.

Examples

Fitting a Function to Data

If you’ve ever hit “add trendline” and “display equation on chart” in excel then you’ve already done this… sort of. Figure 2 shows data from a lab in a first year biology course in which students measure the volume of CO_2 produced through yeast metabolism, and use this data to calculate the rate of metabolism. This is, at its foundation, the very simplest example of function-valued analysis.

```
## Warning in as.numeric(as.vector(RcticG2[1, ])) - serr2: longer object
## length is not a multiple of shorter object length
```

```
## Warning in as.numeric(as.vector(RcticG2[1, ])) + serr2: longer object
## length is not a multiple of shorter object length
```

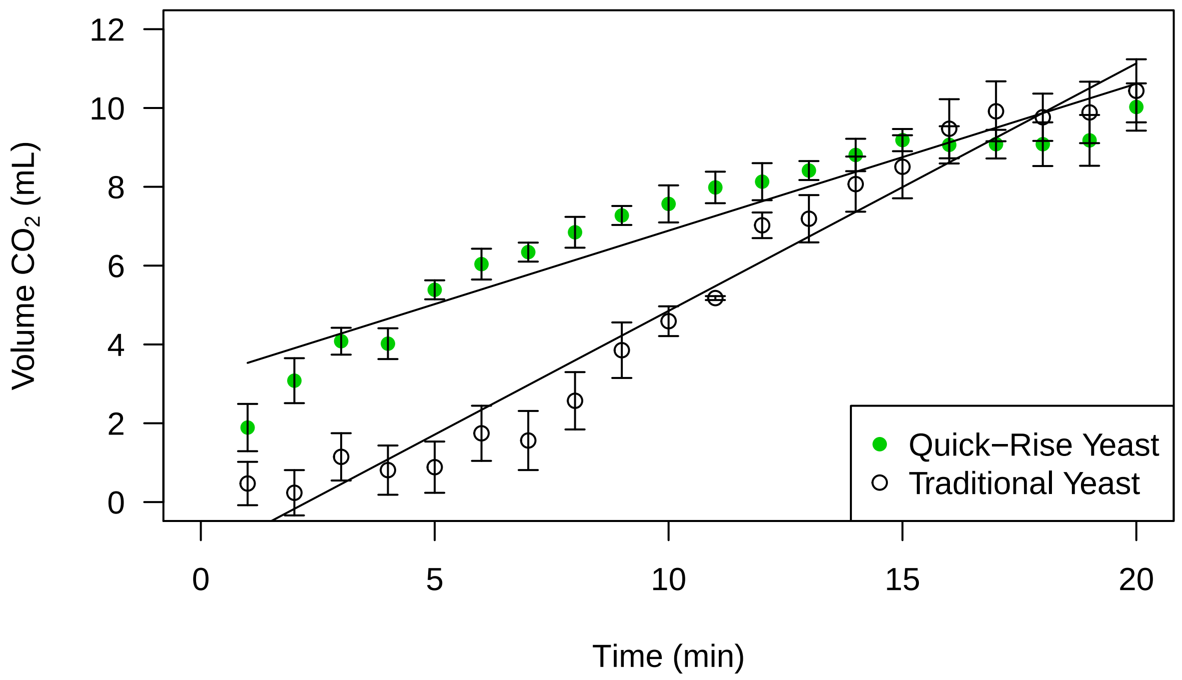


Figure 1: A plot of synthetic data representing the volume of CO_2 produced over time. Error bars represent standard error from three trials.

It is clear from the graph, however, that these stright lines of bestfit 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 better function to fit. Fortunately we don't have to. Here I will introduce a non-parametric method of function-valued traits that does not assume any particular function.

In order to do so, I have created a sample dataset comprised of the lengths of the mythical species of fish, R-ctic Grayling. This dataset has both random individual variance, and random measurement error, and details about how the data was generated have been outlined in the *Data Generation* document. Here is a plot of the data:

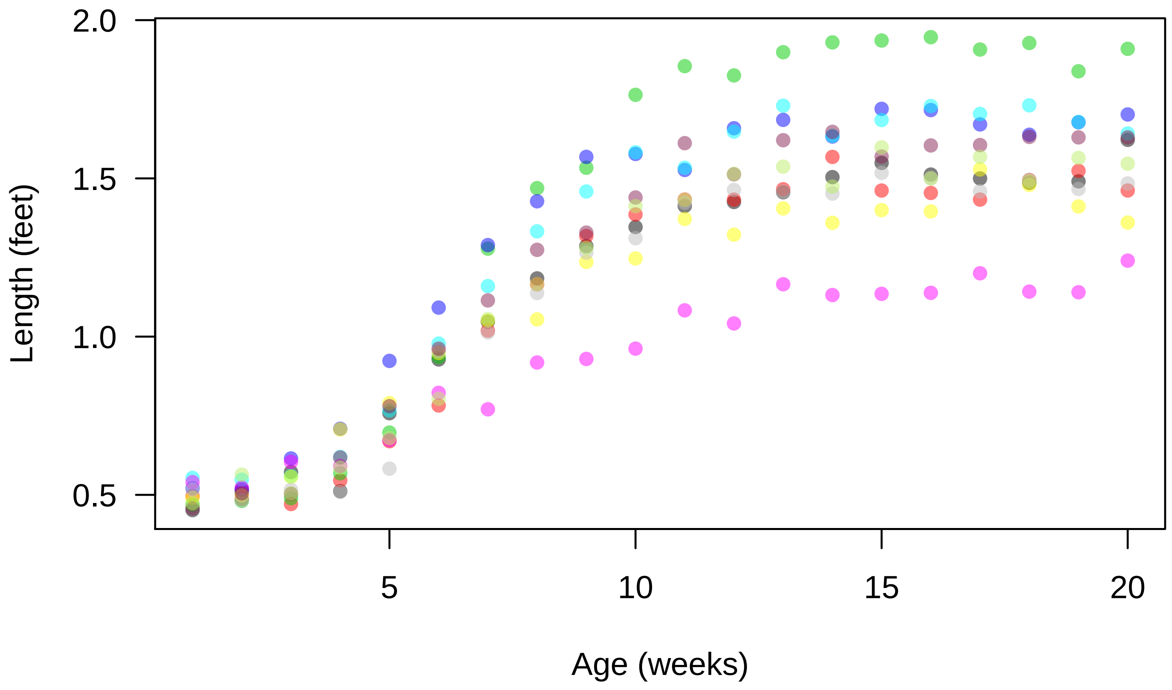


Figure 2: A plot of the ten R-ctic Grayling sampled from population 1.

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 focus on one individual, from one population of R-ctic Grayling. Let's take the first individual from the first population, `RcticG1ind1`.

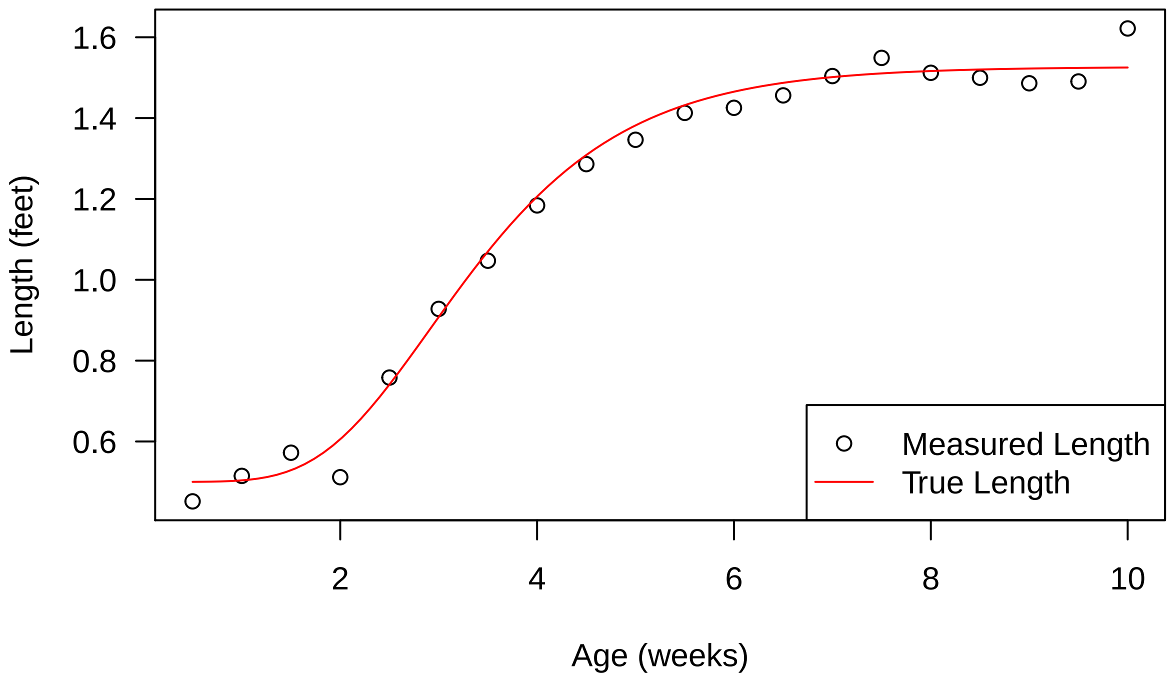


Figure 3: A plot of the true and measured length of `RcticG1ind1`.

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 whatever order we desire by specifying the highest exponent in 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
PolyModelPop1ind1 <- lm(RcticG1ind1 ~ poly(x = xvalues, n = 6))
```

From this we can calculate the predicted 99% confidence intervals.

```
predicted.intervals <- predict(PolyModelPop1ind1, data.frame(x=inp), interval='confidence', level=0.99)
```

Now that we have a model we can plot it against the data to see how it holds up.

First let's plot the data.

```
plot.new(y = RcticG1ind1, x = xvalues, main = "Polynomial Regression", ylab = "Length (feet)", xlab = "
```

Then add the secret function of the true length of RcticG1ind1 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.

```
lines(inp,predicted.intervals[,1],col='green', lwd=3)
lines(inp,predicted.intervals[,2],col=4, lwd=1, lty=2)
lines(inp,predicted.intervals[,3],col=4, lwd=1, lty=2)
```

Throw in 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,
```

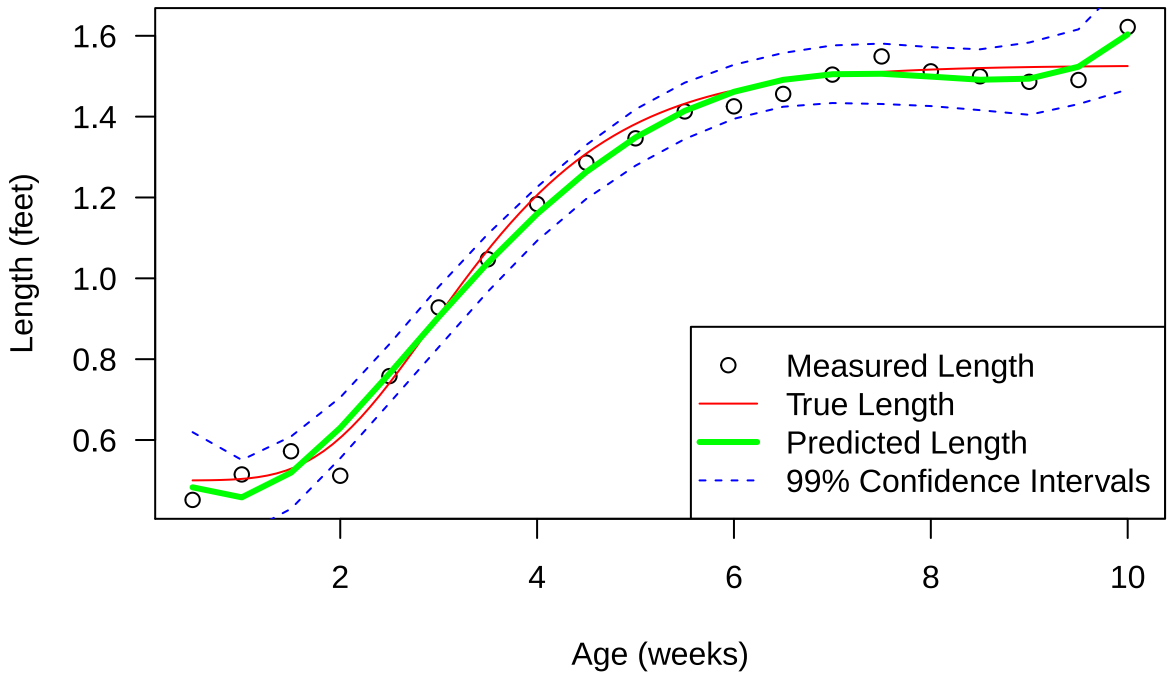


Figure 4: The polynomial fit as compares to the real function and the measured data.

This green line seems like a reasonable fit. But what if we hadn't chosen n = 6? How did that arbitrary decision impact the model? Let's take it to the max, n = 19, and see what happens.

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

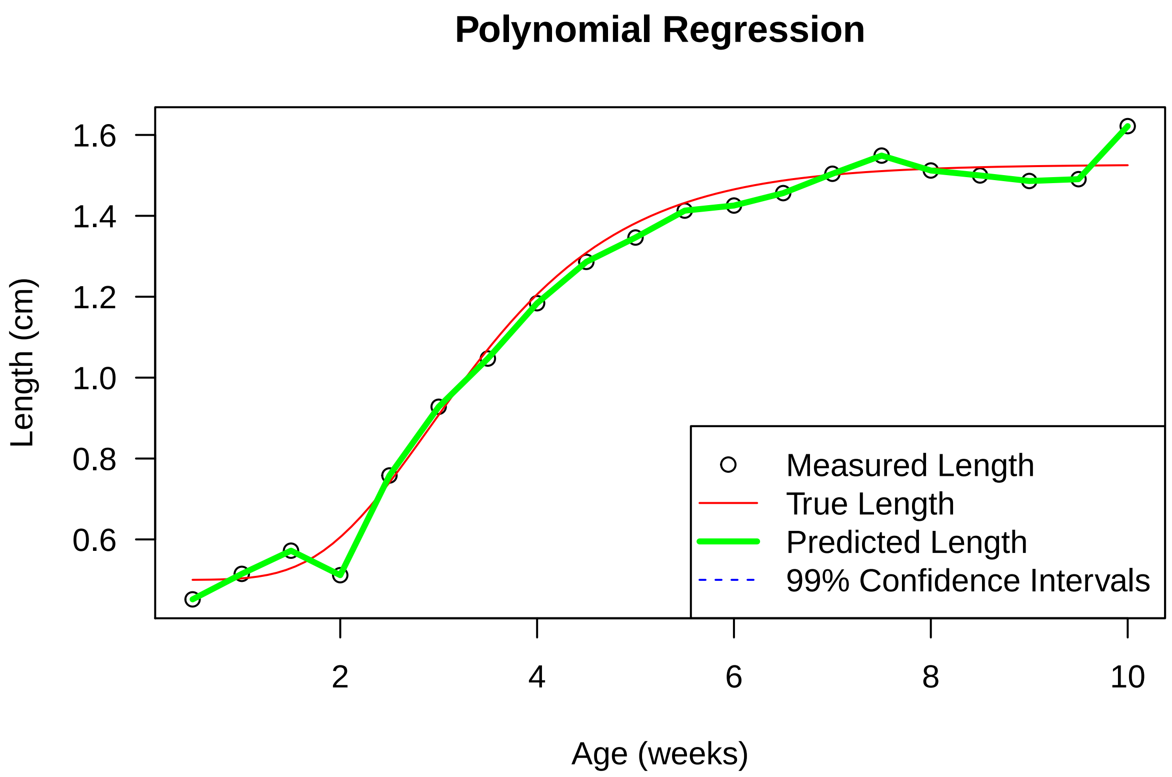


Figure 5: A plot demonstrating the shape of the polynimial fit when the number of terms in the approximating polynomial equals the number of data points.

It is important to note what happpens when n = 19. Here the approximate function has been overparametrized to the extreme and has lost all predictive value. At n = 19 the model also loses statistical power because when the leading term is x^{19}, there are 20 coefficients to analyze, the same as the number of datapoints. The best choice for n would be between 4 and 6.

Analysis

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

```
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. Below is a visualization of the functions present when n = 6.

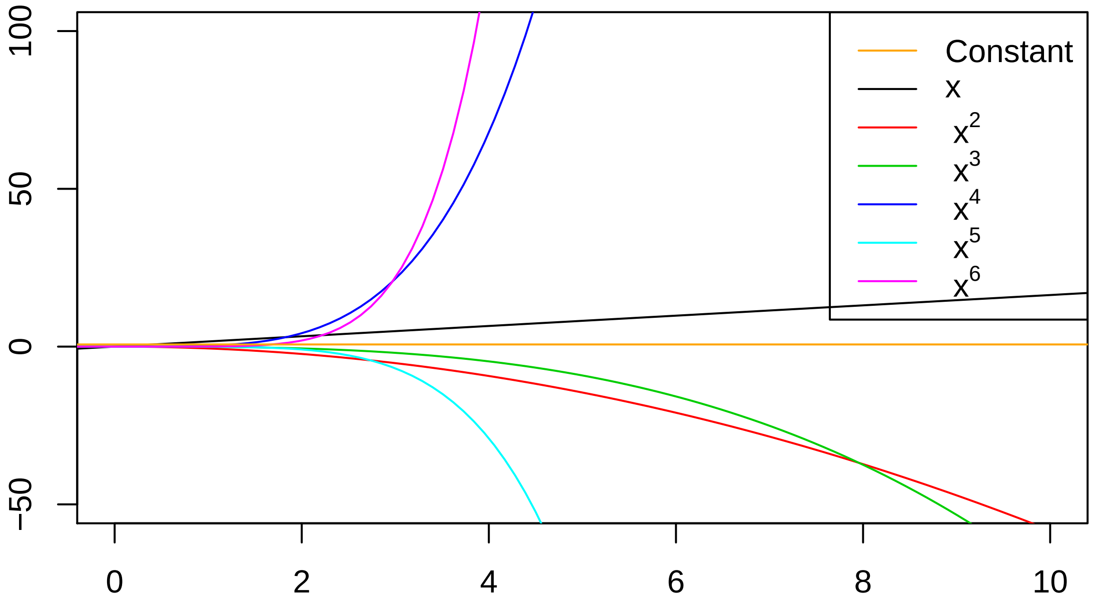


Figure 6: A plot of the functions making up the polynomial approximation of `RcticG1ind1`'s growth function.

Many of these functions get very big, very quickly, but when added together they make up a fair approximation of the length of `RcticG1ind1`.

Now that we have gotten R to describe the model function it's come up with, we are finally ready to broaden our scope from our friend 'RcticG1ind1' and consider theis entire sample. By repeating the process outlined above for all of the samples of the lengths of male R-ctic Grayling, we can collect a dataset of coefficients.

```r
#Take the data frame of the sample lengths
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 ...
```

```r
#and make an empty data frame for the coefficients
PolyModel1 <- data.frame(matrix(data=NA,nrow=7,ncol=10))

#Now we can fill the empty data frame with the coefficients from the RcticG1 sample
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
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 the data frame of coefficients is much more concise than the data frame of the data, but this does not mean that information is lost. (Note that 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 that of the female R-ctic Grayling.

```r
#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 ...
```

Before we go any further we will need to install a package in order to be able to run a Hotellings T^2 test.

```
library("ICSNP")
```

Now we are ready to run the test!

```
HotellingsT2(PolyModel1, PolyModel2)
```

```
##
##  Hotelling's two sample T2-test
##
## data:  PolyModel1 and PolyModel2
## T.2 = 0.14196, df1 = 10, df2 = 3, p-value = 0.9921
## alternative hypothesis: true location difference is not equal to c(0,0,0,0,0,0,0,0,0,0)
```

According to this result we have failed to reject the null hypothesis that the sizes of female and male populations of R-ctic Grayling differ significantly. Before we conclude let us plot these two samples together to see whether or not this is a reasonable outocome.

```
matplot(y = t(RcticG1), type = 'p', lty = 1, pch = 10, ylab = "Length (feet)", xlab = "Age (weeks)", la
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))
```

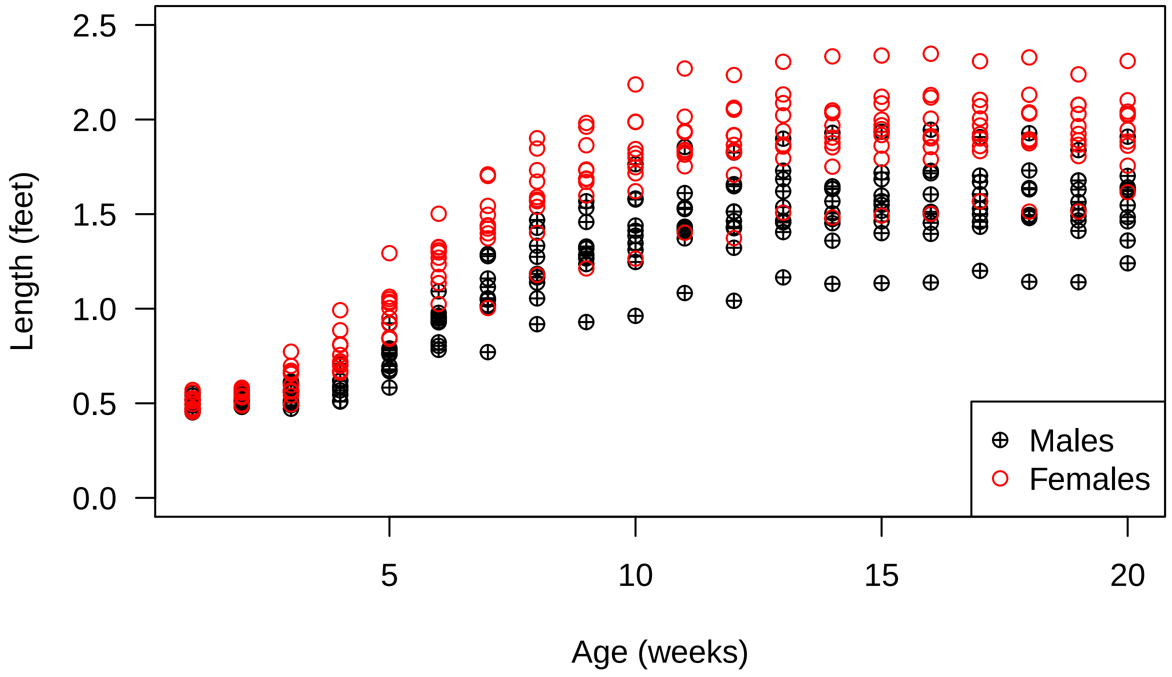


Figure 7: A comparison of male and female R-ctic Grayling lengths

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

Fitting a Sinusoidal Function to Data (still in the works)

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 comouter 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`.

```
spctrm <- spectrum(RcticG1ind1)
```

This will tell us the prominence of various frequencie in the data. If there are consistently peaks at a set interval, then that will feature prominently in this spectrum as an abundant frequency. From this information we can extract the period of the most prominent frequency by knowing that the period is the reciprocal of the frequency.

```
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 a linear model than we were with the polynomial approximation, we are going to use the linear model function once again.

```
SinModelPop1ind1 <- lm(RcticG1ind1 ~ sin(2*pi/period*xvalues)+cos(2*pi/period*xvalues))
```

Again, let's find the predicted intervals.

```
PredictedIntervalsSin <- predict(SinModelPop1ind1, data.frame(x = xvalues),interval='confidence', level=
```

Now we're ready to plot our model. 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)
curve(RcticG(Mono1akg[1, 1], Mono1akg[1, 2], Mono1akg[1, 3], x), n = 101, add = TRUE,
    col = "red")
```

And then add the approximate function and a legend.

```
lines(fitted(SinModelPop1ind1) ~ xvalues, col = "green", lwd = 3)


lines(xvalues, PredictedIntervalsSin[, 2], col = 4, lwd = 1, lty = 2)
lines(xvalues, PredictedIntervalsSin[, 3], col = 4, lwd = 1, lty = 2)

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))
```

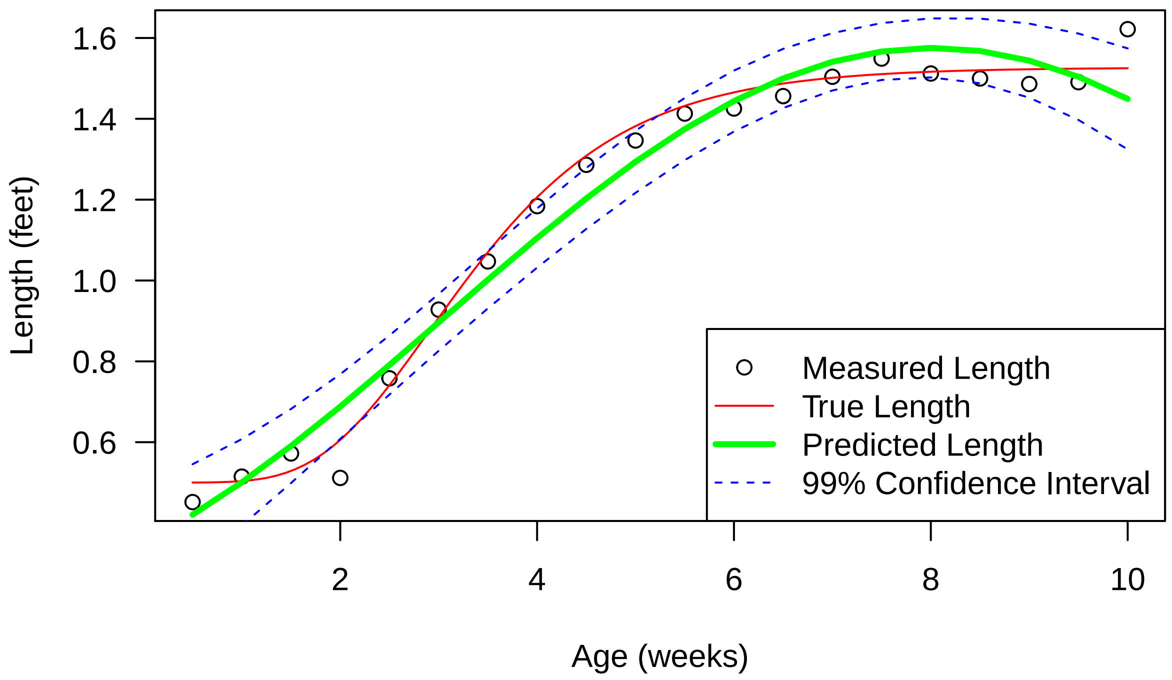
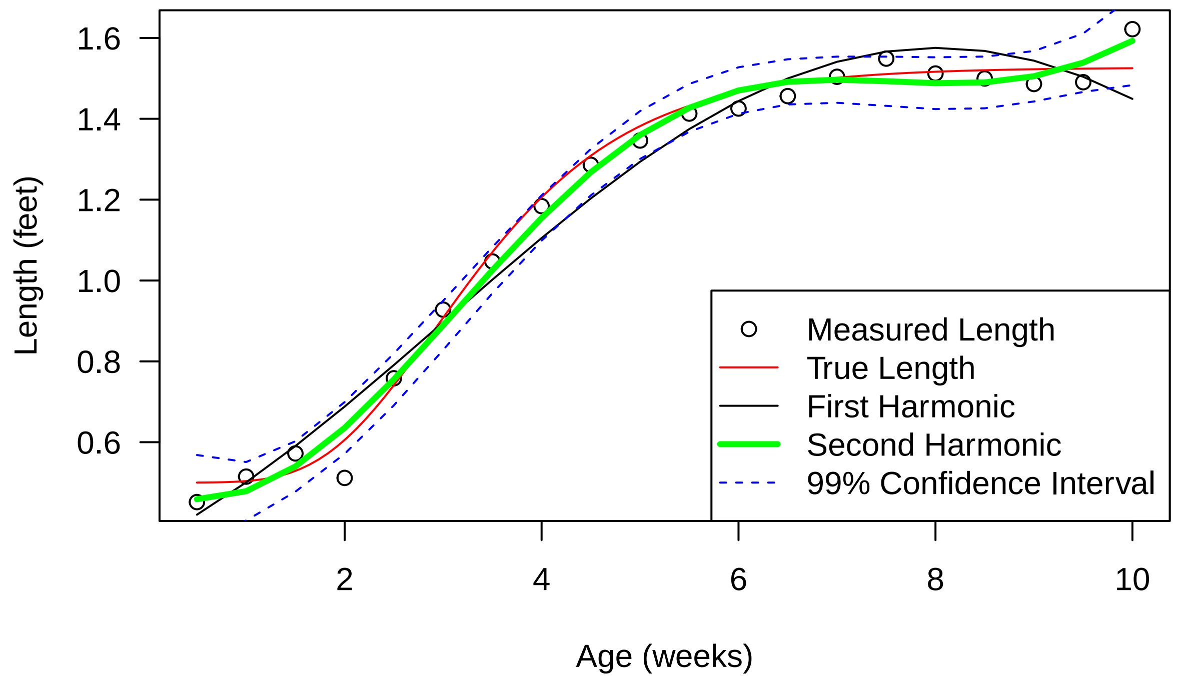


Figure 8: Cosine decomposition

This fit is comparable to a third degree polynomial which isn't so great. Let's see if we can improve the fit by adding a second harmonic.

Now let's plot this and see how our approximation improves.

```
plot(y = RcticG1ind1, x = xvalues, main = NULL, ylab = "Length (feet)", xlab = "Age (weeks)", las = 1)
curve(RcticG(Mono1akg[1,1], Mono1akg[1,2], Mono1akg[1,3], x), n = 101, add = TRUE, col = "red")
lines(fitted(SinModelPop1ind1)~xvalues, col="black", lwd=1)
lines(fitted(SinModel2Pop1ind1)~xvalues, col='green', lwd=3)
lines(xvalues, PredictedIntervalsSin2[,2],col=4, lwd=1, lty=2)
lines(xvalues, PredictedIntervalsSin2[,3],col=4, lwd=1, lty=2)
legend("bottomright",c("Measured Length", "True Length", "First Harmonic", "Second Harmonic", "99% Conf
```



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.

```
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. Below is a plot of the functions that are being added together to form the model.

Activities

Exercises:

1. Manipulate the following function, adding/removing terms, changing coefficients etc. and plot the results to gain an intuition for polynomials.

```
polynomial <- function(x) {
  0 + x + 2*x^2 + 3*x^3 +4*x^4 +5*x^5 + 6*x^6
}
```

Practice Problems:

1. a) Open the sample dataset CatTemp1.csv representing the growth rate of Freija Fritillary caterpillars as a function of temperature and write a program that fits a polynomial to the data. (Growth rate (mg day^{-1}))
   b) Use your program to fit a polynomial to the data in CatTemp2.csv representing the growth rate of the Stella Orange Tip as a function of temperature and compare these populations using multivariate and function-valued methods.

c) Plot the data of both populations, and their polynomial approximations on the same graph. Does it look like there ought to be a

d) TroubleShooting: Iport the data into Excel and use the plotting function to fit a polynomial of bestfit to the data. Do Excel's coefficients match those you found in R?

2. Determine whether or not there is a significant diference between the datasets: (Still need to get it to export nicely but I want this to be the cyclic functions that I have defned, and I want erratic measurements to make it difficult to test a significant difference between the populations without using FVT analysis).