Exercise #5: Fitting a second-order polynomial to sea level data

Patrick Applegate, patrick.applegate@psu.edu; Ryan Sriver, rsriver@illinois.edu, and Klaus Keller, klaus@psu.edu

Learning Goals

After completing this exercise, you should be able to

- fit a second-order polynomial to sea level data
- write your own functions in R
- use `optim()` to perform function minimization

Introduction

In Exercise #4, we examined the question of how high to build structures for flood protection, based on the analysis of van Dantzig (1956). In this analysis, we identified the dike height that minimizes the total costs from dike construction and future flooding. We also examined how changing some of the input parameters within reasonable ranges affects the estimated optimal dike height.

This analysis is appealingly simple; however, it assumes no change in sea level, whereas tide gauge data shows that sea level is rising (see Exercise #1). Sea level has risen by a few tens of centimeters over the last two centuries (Jevrejeva et al. 2008, 2014), and could rise by a meter or more by the end of the century (e.g. Pfeffer et al., 2008, Sriver et al, 2012). If we did not take this rise into account when building flood protection structures, we would build the dikes too low and be flooded much more often than expected as a result (Sriver et al., 2012).

If we wanted to incorporate future sea-level change into the van Dantzig (1956) analysis, we would first need to characterize past sea level change. We can think of a time series, such as the sea level data, as a smooth curve with some superimposed "wiggles." For example, Jevrejeva et al. (2008) noted that the sea level data show a decline between ~1700 and ~1850, then an accelerating rise. These data may be reasonable well approximated for some purposes by a second-order polynomial (a parabola). The sea level data contains some short-period noise, though, so there will be some remaining differences between the data and any smooth curve drawn through the data. These remaining differences are called *residuals*.

Fitting a curve to data is a very common operation in risk analysis and scientific computing in general. As pointed out in the Wikipedia article, curve fitting allows us to separate the trend in a data set from its "wiggles" (smoothing), estimate a quantity between existing data points (interpolation), or estimate a quantity outside of the range of our data (extrapolation).

In this exercise, we'll fit a second-order polynomial to the sea level data from Jevrejeva et al. (2008), both manually and using automated methods. We'll also examine the residuals that are left over after performing this curve fit.

Tutorial

We'll use a somewhat older version of the sea-level data that we saw in Exercise #1, from Jevrejeva et al. (2008). The following code block creates a new directory and downloads the data into it:

```
# Create a directory called data and download a file into it.
dir.create('data')
download.file('http://www.psmsl.org/products/reconstructions/gslGRL2008.txt',
              'data/gslGRL2008.txt', method = 'curl')
```

Once the data are in hand, we can plot them with the following commands.

```
# Read in the information from the downloaded file.
sl.data <- read.table('data/gslGRL2008.txt', skip = 14, header = FALSE)

# Extract two key vectors from sl.data.
# t, time in years
# obs.sl, global mean sea level in mm
t <- sl.data[, 1]
obs.sl <- sl.data[, 2]

# Make a plot.
plot(t, obs.sl, type = 'l', xlab = 'Time (yr)', ylab = 'Sea level anomaly (mm)')
```

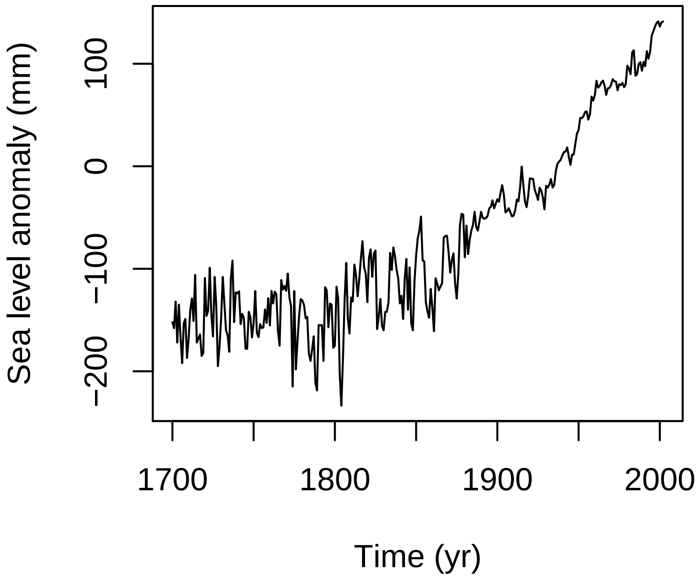

Figure 1: Global mean sea level anomaly as a function of time, from Jevrejeva et al. (2008).

How do we fit a curve to these data? Recall that a second-order polynomial has the equation

$y = ax^2 + bx + c.$

Translating this equation to the sea level context, we have

$L_S = a(t - t_0)^2 + b(t - t_0) + c,$

where L_S is the sea level anomaly in millimeters, t is time in years, and a, b, and c are coefficients of the polynomial. The subtraction $t - t_0$ shifts the lowest point of the parabola to the right (otherwise it would be near $t = 0$).

What units do each of the free parameters have? Only quantities with the same units can be added or subtracted, and therefore each term in this equation must have the same units. It follows that

- t_0 must have the same units as t (years),

- a must have units of mm/yr^2,
- b must have units of mm/yr, and
- c must have units of mm.

Note that the units of a and b correspond to acceleration and velocity, respectively.

Fitting a second-order polynomial to the sea level data involves adjusting the model parameters a, b, c, and t_0 until the curve passes through the data in some best sense. Open the R script `lab5_sample.R` and examine its contents. This script takes values of `a`, `b`, `c`, and `t.0` (set near the top of the script), plots the data and a second-order polynomial based on the free parameter values, and then displays the residuals (`resids`) between the observed and calculated sea level anomalies (`obs.sl` and `est.sl`, respectively).

The root mean squared error

The script also calculates the *root mean squared error* and prints it to the screen using the lines

```r
# Find the root mean squared error between the observed and estimated sea level
# anomaly values.
rmse <- sqrt(mean((obs.sl- est.sl)^ 2))
print(rmse)
```

The root mean squared error (RMSE) is one example metric for how well a given curve matches a particular data set. An RMSE of 0 indicates a perfect match, whereas large values of the RMSE indicate a poor match between a curve and the data. Because the errors are *squared*, the RMSE penalizes outliers heavily. That is, parameter combinations that leave large residuals will receive a high (bad) RMSE score compared to parameter combinations that give smaller residuals between the calculated and observed values. The RMSE has the same units as the differences being analyzed (in the case of the sea level data, mm). There is an interesting historical and statistical background to the choice of this metric, see Jaynes (2003) as well as Morgan and Henrion (1995)

Optimization and functions

R contains a built-in function, `optim()`, that can fit the second-order polynomial to the sea level data for us. `optim()` is a function minimization algorithm: it adjusts the parameters of a function to find lower and lower function values, reflecting increasingly better fits.

If we write a function that accepts

- the free parameters,
- a vector of time values, and
- a vector of observed sea level anomalies as inputs,

and returns the root mean squared error as an output, we can feed that function into `optim()`. An outline for this function might look like this:

```r
# Sample function for calculating the root mean squared error given a set of
# parameters, a vector of time values, and a vector of sea level values.
sl.rmse <- function(params, time, sea.levels) { # don't forget the curly braces

  # Step 1: Pick apart the vector params into its individual values.
  a <- params[1]
  b <- params[2]
  c <- params[3]
  t.0 <- params[4]

  # Step 2: Calculate the estimated sea level anomalies based on a, b, c, t.0,
  # and time.

  # Step 3: Calculate the rmse based on the estimated sea level anomalies
  # calculated in Step 2 and the observed values passed into the function
  # in sea.levels.

  # Step 4: Return the rmse value calculated in Step 3.
  return(rmse)

} # don't forget the curly braces
```

In the function outline above, Steps 1 and 4 are done for you, but you'll need to fill in Steps 2 and 3 on your own (see the Exercise, below).

Once you have your function, you can incorporate it into `optim()` with the code

```
# Optimize the sl.rmse function.
start <- c(0, 0, -100, 1800)
optim.fit <- optim(start, sl.rmse, gr = NULL, time = t, sea.levels = obs.sl)

# Extract the parameters of the best-fit polynomial, and the root mean squared
# error, from optim_fit.
best.a <- optim.fit$par[1]
best.b <- optim.fit$par[2]
best.c <- optim.fit$par[3]
best.t.0 <- optim.fit$par[4]
best.rmse <- optim.fit$value
```

This code specifies a starting group of parameter values (`start`), applies `optim()` to find the best parameter combination, and then picks apart `optim.fit` to find the best parameter combination and RMSE value.

Exercise

Part 1. In the script `lab5_sample.R`, experiment with the free parameters `a`, `b`, `c`, and `t.0` (set near the top of the script) to find a good match to the data. To do this, open `lab5_sample.R`, set the working directory to be the same as that where the script resides, and `source` the script repeatedly, changing the parameter values each time. Make a note of your best-fit parameter combination and its associated root mean squared error value, and save off the plot associated with this parameter combination.
Part 2. Save a copy of `lab5_sample.R` with a different file name. Write a function `sl.rmse()` that accepts a vector of parameter values, a vector of time values, and a vector of observed sea level values as inputs, and returns their RMSE value. The general format of this function is given above. Embed this function near the top of your R script, but *below* the line `rm(list = ls())`. If you insert the following lines near the bottom of your modified R script and `source()` the script, you should get `[1] 96.34322`.

```
# check on function sl.rmse
test.rmse <- sl.rmse(params = c(0, 0, -100, 1800), time = t,
                     sea.levels = obs.sl)
print(test.rmse)
```

Part 3. Save a copy of your R script from Part 2 under a different file name. Modify this script so that it uses `optim()` to find the best-fitting parameter values, plots the best-fit second-order polynomial on top of the data, and displays the resulting residuals. Your final script should *not* include the lines that set values of `a`, `b`, `c`, and `t.0` near the top of `lab5_sample.R`. Again, note the best-fit parameter combination, and save off the plot produced by the script.

Questions

1. In Part 1, above, how does each parameter change the appearance of the second-order polynomial and the magnitude of the residuals? Relate your answer to the general equation for a second-order polynomial in x-y space, given above.

2. Compare your results from Parts 1 and 3, above. How different are the parameter combinations that you identified by manual and automated methods? How similar are the RMSE values and the graphs?

3. What are possible other metrics than could be used to quantify how well a model fits observations? How may this relate to a decision problem?

References

Jevrejeva, S., Moore, J. C., Grinsted, A., and Woodworth, P. L., 2008. Recent global sea level acceleration started over 200 years ago? Geophysical Research Letters 35, L08715.

Jevrejeva, S., Moore, J. C., Grinsted, A., Matthews, A., and Spada, G., 2014. Trends and acceleration in global and regional sea levels since 1807. Global and Planetary Change 113, 11-22.

Morgan, M. G., and M. Henrion (1995), Uncertainty, a guide to dealing with uncertainty in quantitative policy analysis, 332 pp., Cambridge University Press.

Pfeffer, W. T., Harper, J. T., and O'Neel, S., 2008. Kinematic constraints on glacier contributions to 21st-century sea-level rise. Science 321, 1340-1343.

Jaynes, E. T. (2003), Probability Theory: The Logic of Science, 727 pp., Cambridge University Press.

Sriver, R. L., Urban, N. M., Olson, R., and Keller, K., 2012. Toward a physically plausible upper bound of sea-level rise projections. Climatic Change 115, 893-902.

van Dantzig, D., 1956. Economic decision problems for flood prevention. Econometrica 24, 276-287.