LA-UR-16-27181

Title: Guide to Using Onionskin Analysis Code (U)

Author(s): Fugate, Michael Lynn
Morzinski, Jerome Arthur

Intended for: Report

Issued: 2016-09-20

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Guide to Using Onionskin Analysis Code (U)

Michael Fugate and Jerry Morzinski

Statistical Sciences Group, CCS-6

Los Alamos National Laboratory

September 15, 2016

LA-UR-

This document is a guide to using R-code written for the purpose of analyzing onionskin experiments. We expect the user to be very familiar with statistical methods and the R programming language. For more details about onionskin experiments and the statistical methods mentioned in this document see Storlie, Fugate, et al. (2013).

Contents

1 Background

Engineers at LANL experiment with detonators and high explosives to assess performance. The experimental unit, called an onionskin, is a hemisphere consisting of a detonator and a booster pellet surrounded by explosive material. When the detonator explodes, a streak camera mounted above the pole of the hemisphere records when the shock wave arrives at the surface. The output from the camera is a two-dimensional image that is transformed into a curve that shows the arrival time as a function of polar angle. The statistical challenge is to characterize a baseline population of arrival time curves and to compare the baseline curves to curves from a new, so-called, test series. The hope is that the new test series of curves is statistically similar to the baseline population.

1.1 Onionskin Shapes

We are most interested in comparing shapes of onionskin curves, with the overall level of a curve being something of a nuisance. For a given curve the shape is essentially the original curve adjusted so the average arrival time of the curve is 0. Note that in Storlie, Fugate, et al. (2013), we use a slightly different definition of shape but nothing is lost if the user thinks of a shape as a curve with an average arrival time of 0.

So far our analysis has been primarily concerned with comparing the mean shape of the baseline curves to the mean shape of curves from a new test series. However, we do provide many plots showing the individual baseline shapes and the shapes from the new test series.

1.2 Onionskin Names

Engineers use the naming convention “OS1” to mean onionskin tests conducted at ambient conditions; “OS2” means tests done at cold conditions; and “OS3” means tests done at hot conditions. Usually, but not always, you will have more information about exact temperature of each test.

1.3 Measurement Scale

All of the onionskin data we’ve analyzed so far have arrival times recorded in microseconds or nanoseconds. The figures produced by the R-code assume the arrival times are in nanoseconds so a transformation of the times may be necessary.

1.4 Report

We have to a large extent standardized the report format with tables and figures in pre-determined locations. To produce a report there is a Latex document which serves as a template for the report. There is a different template for each of the three shot temperatures. The templates are essentially identical except for the prefix of the pdf file names that are used in the report.

2 Comparing Curves

We were asked to address the following problem. Given a baseline set of onionskin curves and a set of curves from a new test series, does the mean curve from the new test series have the same shape as the baseline mean curve?

2.1 Two Random Vectors

If we think of the observed curves as p-dimensional vectors, where p is the number of breakout angles or arrival times, an obvious way to compare two curves is with the statistic

$$D^2 \;=\; (y_1 - y_2)^T \, \Sigma^{-1} \, (y_1 - y_2)$$

where y_1 is the mean shape of the baseline curves and y_2 is the mean shape of the new test series curves. Σ is an appropriate $p \times p$ covariance matrix. If D^2 is large this is evidence the two curves have different shapes. In the test series data we have seen so far, p ranges from about 100 to over 1000. Older tests have fewer recorded times but new tests typically have over 1000 arrival times.

Unfortunately, Σ is unknown so we can't actually compute D^2. The obvious thing to do is to estimate Σ from the data. Since p ranges from 100 to over 1000 and because at this time we have fewer than 20 tests (baseline and new tests combined) the usual estimate of Σ will not have an inverse, so again we can't actually compute D^2.

Because the total number of curves is less than the dimension p, some kind of dimension reduction technique is needed to statistically compare the mean shapes. The approach we decided on was to fit a B-spline model to each onionskin curve and then to treat the coefficients in the fitted models as our data. So we have changed the problem from comparing mean shapes to comparing the coefficients in a model for the mean shapes. To compare coefficients we use the statistic

$$T^2 = (B_1 - B_2)^T \Sigma_B^{-1} (B_1 - B_2)$$

where B_1 is the vector of B-spline coefficients that describe the baseline mean shape and B_2 is the set of coefficients that describe the new test series mean shape. Now however, the B's are 8 dimensional vectors and Σ_B is an 8×8 covariance matrix so we can actually find the inverse and compute T^2. Note that the B's may have a few more or a few less components than 8. Also, it is important to remember that B_1 and B_2 do not contain a coefficient corresponding to the intercept. The intercept tells us about the overall level of a curve and not the shape.

A multiple of T^2 is Hotelling's t-squared statistic that we calculate for the report. Again, large values of Hotelling's statistic suggests the curves have different shapes. See Storlie, Fugate, et al. (2013) for more details.

In looking at several new test series, T^2 is pretty sensitive to differences in shapes. Some differences between shapes that are statistically significant may not matter for the application of interest. This question is currently being studied by other scientists.

Calculating Hotelling's t-squared statistic, and a p-value, is just one step in the analysis. We also look at the curves and various features of the curves, like the the first breakout angles, and make our decision based on all of the evidence, not just the p-value.

For many years scientists at LANL have used so-called pointwise error bands to help decide if a new mean curve is like a baseline mean curve. If the new curve is outside the pointwise bands they decide the curves are different (well, at least in theory this is what happens). Using pointwise bands this way is equivalent to setting Σ to a diagonal matrix of positive values (the variances), which will of course have an inverse. If the arrival times of two angles are uncorrelated then setting the off diagonal elements of Σ to 0 makes sense. We have seen nothing in the data that makes us think the arrival times are uncorrelated. We do not advocate using pointwise bands to compare curves. We discuss the construction and use of pointwise bands in a little more detail below.

2.2 Control Charts

In many respects comparing a new mean shape to a baseline mean shape is really just a problem in constructing an appropriate control chart. In a typical application of control charts we usually have a lot of data from when the process is known to be in control. So, for example, if we are producing ball bearings of a fixed diameter, we will know the target diameter and we will run the process in a monitoring stage long enough to get the process variability down to an acceptable level before we start manufacturing ball bearings for sale. In process monitoring language, for an in control process we will know the center line and the control limits essentially without uncertainty.

As we produce more ball bearings periodically we will sample some small number and measure them and then plot the sample average on a control chart. If the average falls within the control limits we continue the manufacturing process.

Naturally, one might think that the baseline mean shape can play the role of the center line and the error bands around the mean shape act like the control limits. There are two reasons, perhaps more, why this thinking isn't completely correct. First, it's quite easy to fit some very strange and unacceptable shapes between error bands. So even if a new shape fit within the error bands the shape could be "obviously wrong" and unacceptable. In a typical control chart application we are monitoring a scalar and it is either between the control limits or it's not, no other options. The object we are monitoring is a function (curve, shape) and having a function within error bands is not sufficient to conclude the process is in control.

The second reason is we have only a small number of baseline shots so there is uncertainty, perhaps a lot, in the "center line" and the error bands that has to be accounted for. Over time, as we collect more baseline shots, we should be able to reduce this uncertainty and characterize the baseline population much better.

We emphasize again that we are monitoring a function and typical control chart applications monitor a scalar. Of course, there are applications that use multivariate control charts but they are still monitoring a q-dimensional point, not a function. Plus, multivariate control charts have some issues that have limited their wide spread use.

3 Steps in an Analysis

This section is a high-level summary of the steps a user needs to go through to generate tables and figures to compare onionskin curves from two populations and to generate a report. The remainder of the document provides details about each of these steps. When OSX appears in a file name X can be 1, 2, or 3 depending on the temperature of the test series.

1. For each new test series create a directory with appropriate subdirectories.

2. From the templates directory copy fReadOS-test-series-date.R and f-fit-OSX-test-series.R to the current working Rcode directory. Rename the files to reflect the current test series name.

3. From the templates directory copy osX-template.tex to the current Report directory. Rename the template.

4. From the Rcode directory read in the raw data and do some processing with fReadOS-test-series-date.R. This file saves the processed data as an R data file that can then be "loaded" into an R session. After creating this data file we move it to the appropriate Data directory.

   - If the arrival times are in microseconds this file a good place to convert them to nanoseconds.

5. From the Rcode directory generate figures and tables with f-fit-OSX-test-series.R.

6. Move pdf files generated in step 5 to the /Report/zFigures directory.

7. Step 5 produces several files with the extension "txt" or "TAB". These files are Latex formatted tables. Open these files, copy the table, and paste into the appropriate location of the Latex report document, located in the Report directory.

8. Typeset the report and open the pdf.

9. Summarize the results and typeset the document again .

Comments:

- When you copy the templates fReadOS-test-series-date.R and f-fit-OSX-test-series.R you want to rename with the current test series number (see below for more details).

- It's worth generating the plots one at a time in R and sending the plots to the screen before generating the pdf versions. This will tell you if you need to mess with any axis limits. We've tried to use defaults limits that are good for the tests we've seen, but you may still need to tweak a few of them.

- The pdf file names are set in f-fit-OSX-test-series.R. Do not alter the default file names. The Latex report template expects the specific file names generated by f-fit-OSX-test-series.R. Of course, if you want to rename everything we've automated that's really your business.

4 Users Directory Structure

To organize the analysis of what will likely be many future onionskin tests, we use the following directory structure on a Linux system. Of course, anyone analyzing onionskin data is free to use whatever system or directory structure that makes sense to them.

- /Projects/B61/Common-R-Files/

- /Projects/ B61/FY16/Data/TestSeriesName
- /Projects/ B61/FY16/OSTests/Templates
- /Projects/ B61/FY16/OSTests/TestSeriesName/Rcode/
- /Projects/ B61/FY16/OSTests/TestSeriesName/Report/zFigures/

Comments:

- **Projects:** this directory is under the users "home" directory
- **B61:** where we analyze onionskin tests for the B61 and possibly other work
- **FY16:** stands for Fiscal Year 2016
- **Common-R-Files:** a directory that holds only files used by all onionskin analysis, see below
- **Templates:** a directory that holds files used by specific tests, e.g. ambient tests
- **TestSeriesName:** name of the test series being analyzed, e.g. LT62-3 or some such thing
- **Rcode:** holds R-files specific to the test series being analyzed. For each test series there is an Rcode directory where all analysis is done.
- **Report:** directory that has Latex files for the analysis. The general form of the analysis is the same for all new test series, only the details are different. The Latex template has default file names of plots to be imported that should not be changed unless you really really want to. The Latex document is very "staple ready".

- **zFigures:** holds the pdf files generated in the Rcode directory. After generating the files you need to move them to the zFigures directory. The Latex template is expecting to read files from this location. Also, by moving them to where the report is written this helps avoid accidentally overwriting a plot.

We have also analyzed “booster” only shots in which case we use the directory “BoostTests” or some such name instead of “OSTests”. Otherwise the rest of the directory structure is identical.

5 Description of R and Latex Files

5.1 Files in Common-R-Files Directory

- **fcode-For-OS-15July2015.r**
  - This file has all the user written R-functions needed for an onionskin analysis (e.g. plotting, fitting, testing). There are over 20 functions in this R-file.
  - This file is “sourced” by the fitting functions described below so all the functions will be available in the R session.
  - If you need to change, for example, tick marks on an axis, this is where you will have to go.
  - If you want to create plots that are not currently available you will write a new function to make the plot and then put the code here after you get the bugs worked out.

  - Though rare at this point, whenever significant changes are made to one or more routines the date/name of this file should be changed.
  - If you create a new version of this file do not delete previous versions as previous onionskin analyses need this exact code (just in-case we have to go back for some reason and re-do the analysis).

5.2 Files in OS-Template Directory

The R files in the template directory need to be copied to the current working Rcode directory. You always copy the fReadOS-test-series-date.R file and one or more f-fit-OSX-test-series.R files. When the files are copied the names are changed to reflect the current test series name. For example, if we have new ambient test data from series 62-3, we would copy fReadOS-test-series-date.R to fReadOS-62-3-Date.R and f-fit-OS1-test-series.R to f-fit-OS1-62-3.R and then move these files to the current Rcode directory. Again, fReadOS-test-series.date.R and at least one of the "fit" files will always need to be copied to the appropriate working directory.

Similarly, the Latex files are copied to the Report directory with an appropriate change in name.

Files in Template directory:

- fReadOS-test-series-date.R
- f-fit-OS1-test-series.R

- f-fit-OS2-test-series.R
- f-fit-OS3-test-series.R
- os1-template.tex
- os2-template.tex
- os3-template.tex

The f-fit-OSX-test-series.R files are almost identical, the only difference is the output pdf files have slightly different names and the plot titles are slightly different. For example, f-fit-OS1-test-series.R produces output files with prefix something like “os1” and a plot title like “OS1: Something”.

The files with suffix “tex” are Latex documents. They are identical except for the file names of figures that are imported and section headings. These serve as templates for the report.

5.3 Files in RCode Directory

These files are copied from the template directory and renamed once copied.

- **fReadOS-test-series-date.R**
  - Reads in raw data, does some processing, and combines into an R list.
  - After all the processing the data is saved as an R data file that can be loaded into an R session. The R data file is moved to the appropriate Data directory.

  - Sometimes the engineers send one csv file per shot and sometimes all the data is in one csv file, which means this code always needs some editing.
  - Again, we change the file name depending on the current set of shots.
  - “test-series” is usually something like 62-3. W-division engineers will tell you the test series when they send new data.
  - “date” is something like 15April2016 and indicates the date the data was sent.

- **f-fit-OSX-test-series.R (X is 1, 2, or 3)**
  - **This file produces all the figures and tables used in the report**
  - This file is just a bunch of “if” statements.
  - Each “if” statement calls a function that will produce a plot, a table, or calculate Hotelling’s T-test .
  - To call the function within a given “if” statement you just need to set the flag for that “if” statement to T. For example, setting “c0” to T will call the function that fits the B-spline model and computes everything the other functions need. Hint, you always have to call c0 first.
  - If you set all the flags to T and the flag for a pdf file to T then this code generates all the figures and corresponding pdf files.
  - Before you can make pretty plots or anything else you will, of course, have to load the baseline data and the new test data. You do this by just setting the “paths” to where the data is stored. In the case of the new test data, you are

going to “load” the R data file that was previously saved when you ran fReadOS-test-series-date.R.

6 Reading and Processing the Raw Data

6.1 Baseline Tests

W-division engineers conducted 20 baseline onionskin tests at 3 different temperatures: 6 ambient, 8 cold, and 6 hot. Only shots at the same temperature are compared. W-division uses “OS1” to mean ambient, “OS2” for cold, and “OS3” for hot. Files they send often have this designation and we also use this convention in the summary report.

The baseline data has been processed and saved in an R data file that for purposes of this document we call “OS-baseline.dat”. This data file is a list with 3 components: “Amb”, “Cold”, and “Hot”. Each of these is also a list with three components: “shot.dat”, “shot.names”, and “shot.temps”.

- **shot.dat:** a list of length equal to the number of shots. Each element of the list is a $n_k \times 2$ data frame; one column of recorded angles and another column of arrival times. The columns are labeled “Time” and “Angle”. Times need to be in nanoseconds so you may have to transform what is sent. User sets these in fReadOS-test-series-date.R.
- **shot.names:** a vector of length equal to the number of shots. User sets these in fReadOS-test-series-date.R.

- **shot.temps:** a vector of length equal to the number of shots. User sets these in fReadOS-test-series-date.R to a numeric value if know or just a name (e.g. "Amb").

6.2 New Test Series Data

Sometime the engineer will send results for onionskin tests done at two or three temperatures, but most often they do a set of tests at just one temperature and send these data.

For purposes of this document, suppose the new test data is from series 62-3 and the tests were done at ambient conditions. Also assume the data was sent to us on 8 December 2014.

To read in and process the new test data we copy the file fReadOS-Data-Template.R to fReadOS-62-3-8Dec14.R and then move it to the appropriate Rcode directory (see above for more details). This R file reads the raw data and puts it together in a way that other functions expect. After putting things together in the right way the results are saved to an R data file with name, again for this report, "os62.3.Fake.Test.dat". After creating this R data file we move it to the appropriate Data subdirectory (see above).

For all analyses we load the R data file from the appropriate subdirectory. The raw data isn't used after the R data file is created and saved.

Because we only have new tests done at ambient conditions, os62.3.Fake.Test.dat is a list with one component, "Amb", and this is a list with 3 components: "shot.dat", "shot.names", and "shot.temps", exactly the same components and names as the baseline data.

If we had been sent new tests done at both ambient and cold conditions then os62.3.Fake.Test.dat would be a list with two components: “Amb” and “Cold”, where each of these would be a list of three components (can you guess the names?)

7 Figures and Tables Produced for the Analysis

7.1 Figures

1. A 2 × 2 figure:
   - Top left: baseline shapes along with their mean shape
   - Top right: test data shapes along with their mean shape
   - Bottom left: plot of both baseline and test shapes together
   - Bottom right: plot of both mean shapes
2. Observed baseline curves with B-spline fit
3. Observed test curves with B-spline fit
4. Baseline mean shape with pointwise error bands and the test curve shapes
5. Baseline mean shape and pointwise error bands and the test data mean shape
6. Baseline mean shape and new test data mean shape
7. B-spline coefficients for baseline and test shapes

8. B-spline coefficients for baseline and test mean shapes
9. First breakout angles for baseline and test data
10. First breakout time minus pole arrival time (so-called delta-t) for baseline and test data
11. Baseline curves with B-spline fit; a more detailed version of (2)
12. Test curves with B-spline fit; a more detailed version of (3)
13. Pointwise residual shapes for baseline data (observed shape minus baseline mean shape)
14. Pointwise residual shapes for test data (observed shape minus baseline mean shape)
15. Figure with summary results from Hotelling's T-squared test

7.2 Tables

1. Table with shot names, number of points per curve, and range of angles for baseline data
2. Table with shot names, number of points per curve, and range of angles for test data
3. Table with root mean squared error for B-spline fit to test data and some related stuff
4. Table of delta-t (first breakout time minus pole arrival time)

8 B-Spline Knots

The baseline onionskin curves have minima around ± 60 degrees and a local maximum near the pole (0 degrees). The exact knot locations are given in the "f-fitOSX-test-series.R" functions and the knots change depending on the which set of baseline data you are using (i.e. ambient, cold, or hot). The knots were chosen so the spline models would fit the baseline onionskin curves well near the minima and the local maximum.

The user should not change the knot locations as they were chosen to fit the baseline data well. If these knot locations lead to spline fits that don't fit the test data well this is evidence the test data is different from the baseline data.

9 Hotelling's T-squared Statistic

As discussed above, we use Hotelling's t-squared statistic to compare the B-spline coefficients for mean shapes. For convenience, we send the observed value of the test statistic and two p-values to a figure that can then be included easily into the report.

We report a p-value based on normal theory and one based on a permutation test. The permutation p-value is calculated from a large number of randomly chosen subsets, not necessarily all possible subsets. We also report a p-value for a test of equal variability. At this point in time this is mostly for our amusement.

The most important thing to remember is that we exclude the coefficient for the intercept when calculating the t-squared statistic. We are only interested in the shapes and the intercept tells us about overall level of the curve.

R-functions used:

- fFitBSplines
- tsquared
- fPermuteT2
- fpermuteVariances
- fPlotTestResults

10 Smoothing Splines

For some purposes it is useful to have all the onionskin curves registered at a common grid of angles. A straightforward way to register the arrival times is to fit a smoothing spline model to each curve.

For every onionskin curve we fit a smoothing spline model with 40 degrees of freedom to the arrival times. We then predict the arrival times of each curve at a grid of N evenly spaced angles from -80 to 80 degrees. N is typically 200 and is set in f-fitOSX-test-series.R. We can also go out further in angles but this range has sufficed so far. This gives smoothed arrival times for all the baseline curves and the curves from a new test series at a common grid of points.

Smoothed onionskin shapes are determined by subtracting the average of the smoothed arrival times from each smoothed curve. As before, a smoothed shape has an average arrival time of 0.

10.1 Pointwise Mean Shape and Error Bands

Using the smoothed shapes a pointwise mean shape is estimated exactly as the name implies. At each grid point we calculate the average of the smoothed shapes at that point. We estimate a mean shape for the baseline data and for the new test series data. The mean shapes can then be plotted over the common grid of angles.

Using the smoothed shapes we calculate a pointwise sample standard deviation at each grid point. Currently, we only calculate the pointwise standard deviation for the baseline data.

For the baseline smoothed shapes let m_i be the mean shape and s_i the standard deviation at grid point i. We can plot $100(1-\alpha)\%$ error bands by plotting $m_i \pm K\, s_i$ where K is an appropriate scalar. For example, we can plot error bands for a new shape or error bands for a new mean shape. The exact value of K depends on what kind of error bands we want.

The error bands are nothing more than N confidence intervals (or prediction intervals). An obvious issue is these are not simultaneous intervals. If each pointwise confidence interval is, say, a 95% interval, it's almost certainly the case that when looked at across all N grid points our confidence is something less than 95%. How much less is anybody's guess.

The most obvious way to construct simultaneous bands is to use a Bonferroni adjustment for each interval. Unfortunately a Bonferroni correction with N of 200, the number of pointwise intervals, makes the bands so wide they are essentially useless. Because we can model the onionskin curves with 8 basis functions we think of the curves as being, roughly, 8 independent observations. Thus, the pointwise error bands we construct use a Bonferroni correction factor of 8. Obviously, this is a rough correction but it's not totally stupid.

We do not advocate the use of error bands to compare onionskin curves, but since scientists at LANL have used them for many years we decided to put them in the report. The main reason we object to pointwise error bands is that curves with many unacceptable shapes can clearly fit within the bands.

R-functions:

- fFitBSplines
- fPlotPointBands

10.2 First Breakout Angles

The first breakout angle is the angle corresponding to the smallest, or first, arrival time; there is one on each side of the pole. We can determine the first breakout angle using the original curve or the spline smoothed curve (not the B-spline smoothed fit). We prefer using the smoothed arrival times because the raw data is a little noisy for finding the minimum time. The noise is introduced when the engineers take the 2-D image and produce an onionskin curve.

Scientists like to see first breakout angles about 60 degrees from the pole. First breakout angles closer than 50 degrees or further than 70 degrees from the pole are reason for concern. Possibly a lot of concern.

R-functions:

- fFitBSplines
- fSmoothBreakouts
- fPlotBreakouts

11 B-Spline Error Bands

Using the estimated B-spline coefficients we produce plots showing error bands for the difference in mean shapes, for the baseline mean shape and the new test series mean shape. The error bands are produce by first sampling a large number of possible B-spline coefficients for the mean shape and then determining a large number of possible mean shapes, or differences in mean shapes.

We are not big fans of error bands for functional data because a very strange looking curve, that is quite different from the baseline curve, can fit between the upper and lower bands. We include the bands more out of curiosity at this point rather than as a formal way to make decisions.

In classical statistical hypothesis testing there is a well know duality between confidence regions (intervals) and rejecting an hypothesis. This duality exists because one is inverting a test to get a confidence region. For our problem we are not inverting Hotelling's test to arrive at the error bands so it's entirely possible that Hotelling's test rejects the hypothesis the two curves have the same shape but the difference bands contains 0 everywhere (i.e. across all angles) which implies the two curves have the same shape.

Comments:

- Before sampling the coefficients for the mean shape we transform the estimated B-spline coefficients to their principal coordinates.
- In PC space we sample each coefficient for the mean shape uniformly from an interval of the form $m_i \pm t\ \mathrm{SE}(m_i)$ where m_i is the mean of the ith coefficient in PC space and $\mathrm{SE}(m_i)$ is the standard error of m_i.
- The critical value t is the appropriate α level value from a t-distribution with a Bonferroni adjustment equal to the number of spline coefficients.

R-functions:

- fFitBSplines
- fPlotBsplineBands
- fsampleBonferroniBoxPC

12 Delta-t

We were asked to calculate the so-called delta-t for each curve. This is defined as the pole arrival time minus first breakout time (the time corresponding to the first breakout angle).

As there are two first breakout times there are two values for delta-t. This is just another feature to look at to see if the new test series data is similar to the baseline data. We don't have much to say about this in the report beyond the obvious.

R-functions:

- fFitBSplines
- fDeltaTTexTable

13 B-Spline Residual Mean Squared Error Table

We create a table that shows the mean squared error (mse) from the B-spline fitted linear model for each new test series curve. The table also shows the ratio of the mse to the maximum and median mse from the baseline linear model fits. We look at these ratios to see if the the B-spline model fits the new data as well the baseline data.

R-functions:

- fFitBSplines
- fLmMseTexTable

14 Latex Document

We have three Latex templates that can be used to create a report. There is one document for ambient, one for cold, and one for hot tests. There is also some background and other common information in each document. The documents are saved in the templates directory.

The templates have section headings, table locations, and Latex commands for importing figures generated by the “f-fitOSX-test-series.R” functions described earlier. The templates are identical except for the pdf file names that need to be imported . For example, the ambient template imports files with names like “os1-something.pdf” and the cold template imports files with names like “os2-something.pdf”.

References

1. Storlie, C.B., Fugate, M.L., Higdon, D.M., Huzurbazar, A.V., Francois, E.G., McHugh, D.C. (2013), “Methods for Characterizing and Comparing Populations of Shock Wave Curves”, *Technometrics*, 55, 436 - 449.

2. R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.