SS ANOVA

Josef Fruehwald

March 22, 2010

1 The Data

Here's a little demonstration of how to use and analyze Smoothing Spline ANOVAs. Davidson (2006) has a good overview of what SS-ANOVAs are, and an overview of how to analyze them. I was able to follow her executive summary, but can't say I control the math. Here's an example with formant data from /ay/.

I have 21 tokens of /ay/ here, with pre-voiceless tokens excluded. I extracted the formant data from each vowel at 2 ms intervals from Praat. I then scaled the times for each word proportionally, from 0 to 1. I called this variable `RTime`. Then, I classified the vowels into two groups based on whether their log(Duration) was greater or less than the mean(log(Duration)).

	Count
Long	12
Short	9

Table 1: Long/Short Breakdown

The tracked formants, colored by Long or Short group membership, are plotted in Figure 1[1]. As can be seen, there are regions with significant tracking errors. I've measured all of these vowels by hand also, and I recall having a lot of issues with pseudo-formants with high-front vowels, and it looks like the worst tracking errors are also in the glides. The long vowels also seem to have disproportionately bad tracking, and this may be due to the presence of nasal formants. Processing the lpc analysis via Keelan's method would probably improve this kind of analysis.

```
> library(ggplot2)
> theme_set(theme_bw())
> formants <- ggplot(ay, aes(x = RTime, groups = Word, colour = LDur))
> formants <- formants + geom_line(aes(y = F1), alpha = 0.8)
> formants <- formants + geom_line(aes(y = F2), alpha = 0.8)
> formants <- formants + ylab("Hz")
```

2 Smoothing Spline Anovas

To fit SS-ANOVAs to this data, I'm using the `ssanova` function from the `gss` package (R Development Core Team, 2009; Gu, 2009). I believe that Davidson (2006) and Nycz and De Decker (2006) used the `ssr` function from the `assist` package (Wang and Ke, 2009), and I know for a fact that Baker (2006) did. I found `ssr::assist` to be seriously inefficient, and would choke on datasets of about the same size as this one. `ssanova::gss` seems to work more efficiently, is more straightforward to interact with, and was developed

[1] I've included the code to produce this plot here. I'm using the `ggplot2` package (Wickham, 2009)

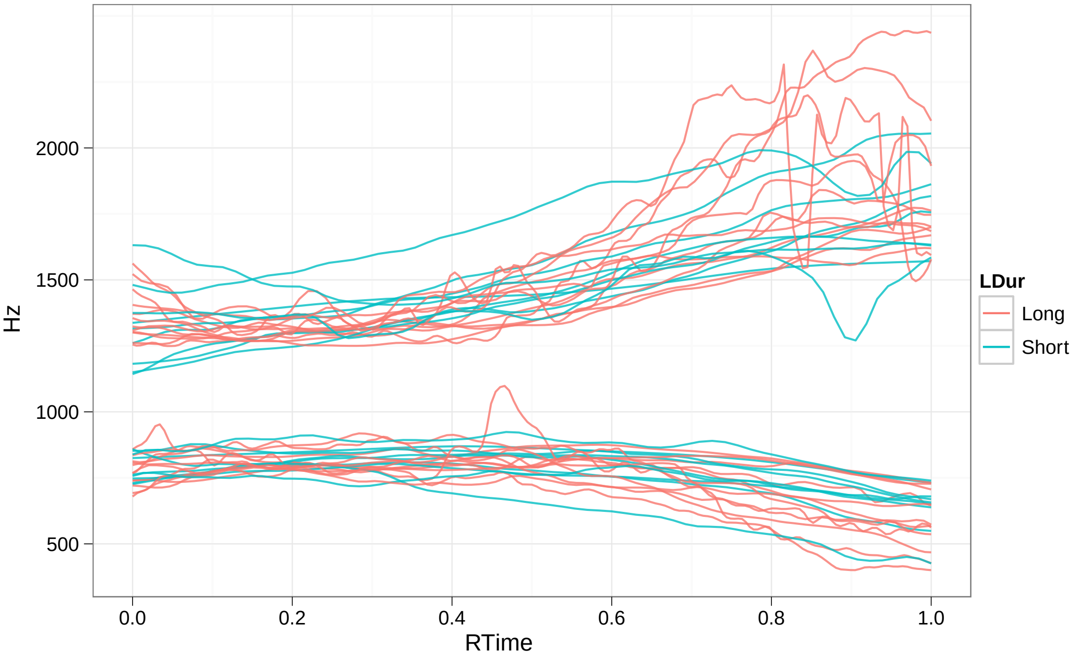

Figure 1: Formant Tracks for 21 tokens of /ay/

by the author of a few pieces of the primary literature cited by Davidson (2006) (Gu, 2002). Putting aside that appeal to ease and authority, I do not control the mathematics here well enough to judge one package over the other, and I feel as if it won't really matter too much for our purposes.

Here is the R code to fit the ANOVA models for F1 and F2.

```
> library(gss)
> f1.model <- ssanova(F1 ~ LDur + RTime + LDur:RTime, data = ay)
> f2.model <- ssanova(F2 ~ LDur + RTime + LDur:RTime, data = ay)
```

There are a variety of summaries and diagnostics to judge the model fits that I don't know or understand right now. Gu (2002) is very hard to read to a non-statistician. However, I have figured out how to look at the model fits graphically by utilizing the `predict` function.

The way `predict` works is it takes a model (linear regression, logistic regression, ssanova), and gives you the response values predicted by that model for some other (usually simulated) set of data. It can also return standard errors you can use to produce Bayesian 95% confidence intervals.

First, we need to generate a dummy dataframe to predict fitted values against. I'll use `expand.grid` to generate a dataframe with 100 points evenly spaced between 0 and 1 for the `RTime` term, for each level of the `LDur` term.

```
> grid <- expand.grid(RTime = seq(0, 1, length = 100), LDur = c("Long",
+     "Short"))
```

You can print `grid` to see exactly what's been done here (it's 200 rows long).

Next, we use `predict` using `grid` as the new data for each model to get the fitted values for F1 and F2, plus the standard error.

```
> grid$F1.Fit <- predict(f1.model, newdata = grid, se = T)$fit
> grid$F1.SE <- predict(f1.model, newdata = grid, se = T)$se.fit
> grid$F2.Fit <- predict(f2.model, newdata = grid, se = T)$fit
> grid$F2.SE <- predict(f2.model, newdata = grid, se = T)$se.fit
```

Now, you have to plot the data you've stored in `grid` somehow. There are tons of ways to skin this cat, and I'll use `ggplot2` again (see ssanova.rnw for plotting code).

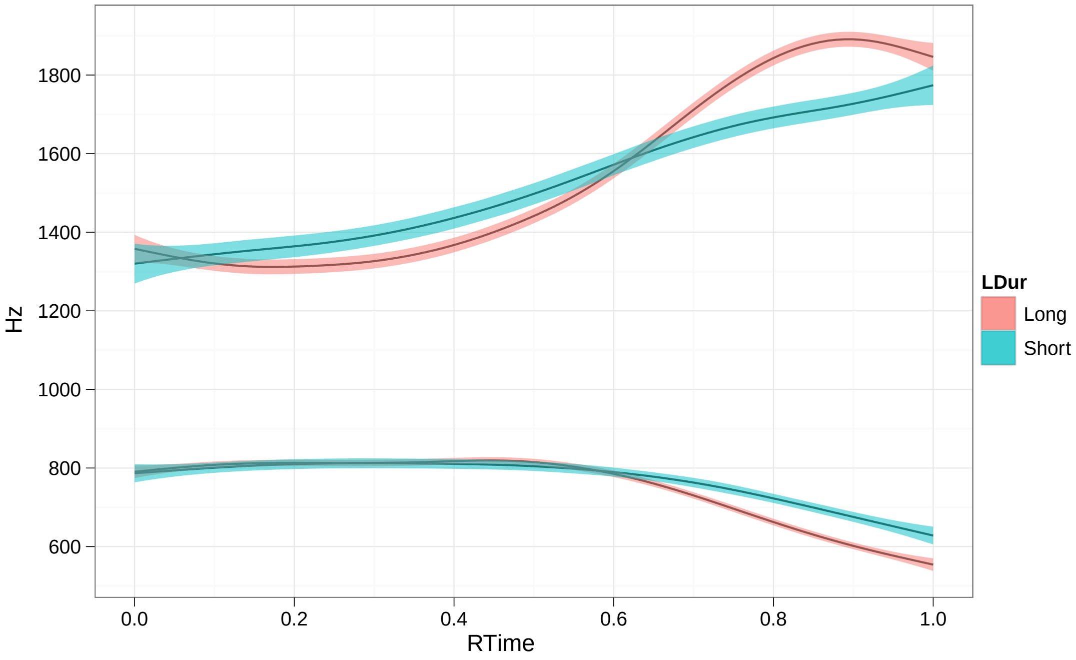

Figure 2: Smoothing Spline ANOVAs with 95% Confidence intervals

Figure 2 is nicely informative. I think we can tell the following things from this plot:

- Long [ay]'s have significantly more peripheral offglides. This isn't too surprising, since we would probably expect short [ay]'s to experience significant undershoot.
- Duration differences seem to have a more extreme effect on F2 dynamics than F1. Both short and long [ay]'s F1s seem to have a steady state, but short [ay]'s F2s seem to be all transition. The consequences for measurement would be that short [ay]'s would appear to be, and maybe are, significantly fronter.

2.1 Analysis of the terms

It is also possible to inspect the individual contributions each term makes to the shape of the spline. I don't exactly control this part either, but as best as I understand it, every term contributes something to the overall shape of the spline:

1 This is the intercept, just an additive constant.

RTime An abstract shape of the formants, without any group effects.

LDur This will also be a constant, additive effect of group.

LDur:RTime This is the really interesting effect that makes each group's spline different, beyond simple upward or downward shifts of the same shape.

Let's look at the abstract formant shapes first. We can think of this as the underlying shapes of the formants. Group membership (here, Long or Short) essentially does transformations on these shapes. We'll tell `predict` that we want it to predict fitted values and standard errors for the Intercept and RTime terms[2].

```
> ## create prediction grid
> grid <- expand.grid(RTime = seq(0,1,length = 100),LDur = c("Long","Short"))
> ## Predict for certain terms
> ## The "inc" argument to predict is key here
> grid$F1.Fit <- predict(f1.model,grid,se = T,inc = c("1","RTime"))$fit
> grid$F1.SE <- predict(f1.model,grid,se = T,inc = c("1","RTime"))$se.fit
> grid$F2.Fit <- predict(f2.model,grid,se = T,inc = c("1","RTime"))$fit
> grid$F2.SE <- predict(f2.model,grid,se = T,inc = c("1","RTime"))$se.fit
```

The resulting plot is in Figure 3.

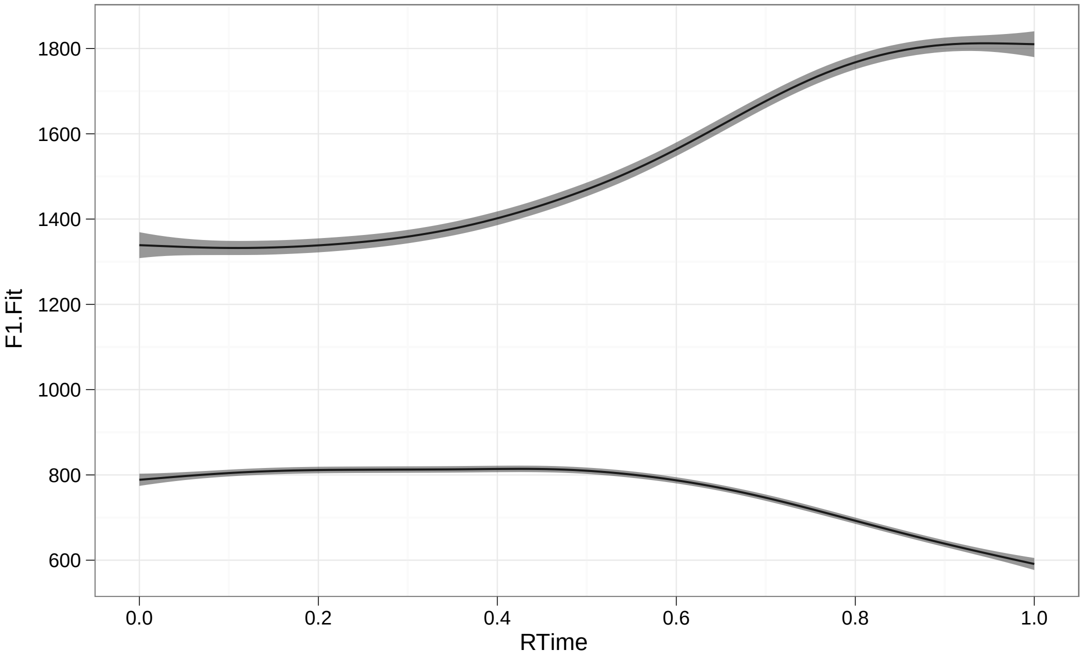

Figure 3: Abstract Formant Splines (Intercept and RTime Terms)

So far, so good. Now, we want to know if group membership significantly warps these abstract formant curves. To do that, we'll predict fitted values and standard errors for the group and interaction terms: LDur and LDur:RTime. This differs from the analysis in Davidson (2006), and importantly so. Davidson only looked at the interaction effect, which looks appropriate for her data after eyeballing it. It looks as if she had non-significant group effects, and significant interaction terms. For our dataset, we have *both* significant group effects and significant interaction effects.

Figures 4 and 5 show the group effects with 95% confidence intervals. Notice that the confidence intervals do not contain 0, so the group effects are significant. It is these group effects, plus the interaction effects, which constitute the overall effect of group membership on the transformation of the abstract formant curves in Figure 3. Failing to include these significant group effects in our prediction would result in uninterpretable

[2] I've included the intercept here so that the curves we eventually plot are in a familiar, interpretable space.

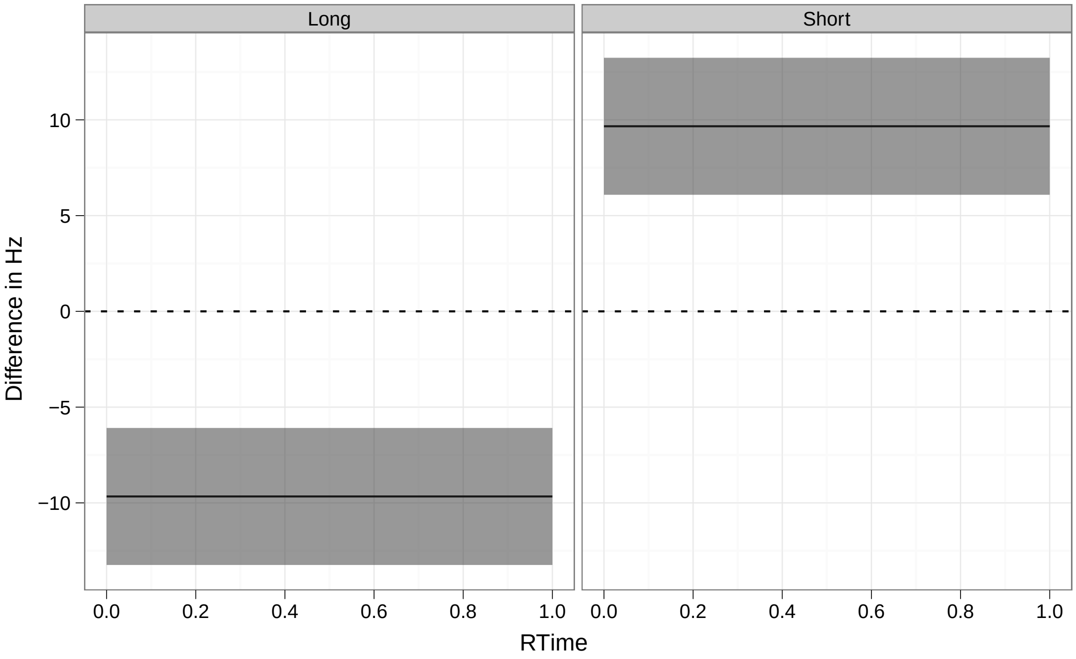

Figure 4: F1 Group effects

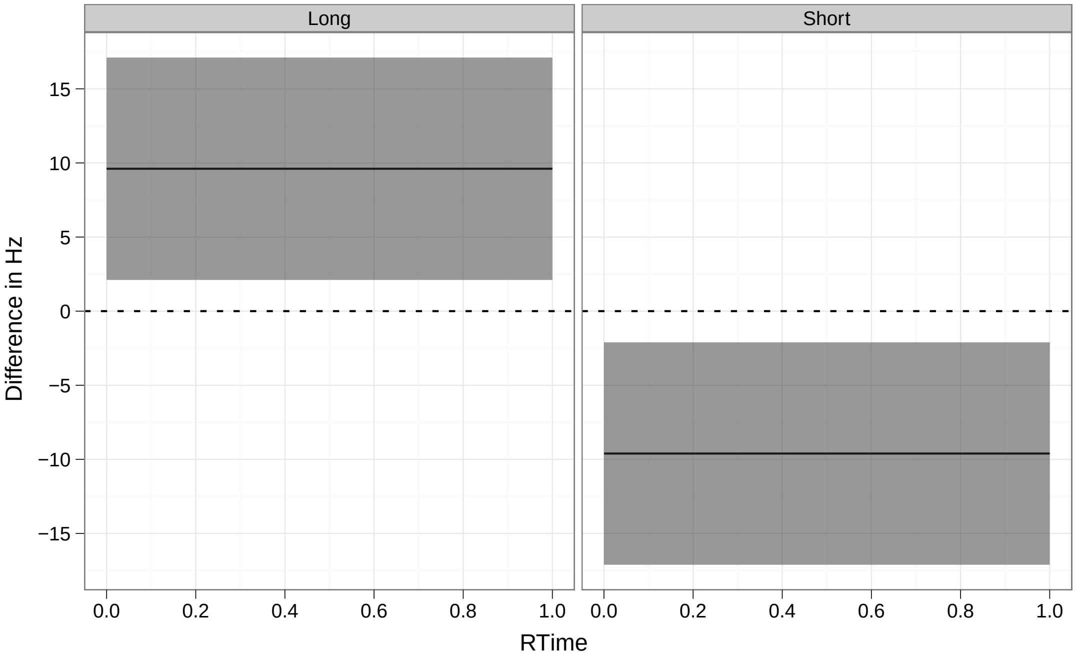

Figure 5: F2 Group Effects

plots. What's there to lose by also including group effects when they're non-significant? I don't really know, but I might guess that the confidence intervals would be larger.

At any rate, here is the code to predict the group and interaction effects for F1:

```
> grid <- expand.grid(RTime = seq(0, 1, length = 100), LDur = c("Long",
+     "Short"))
> grid$Fit <- predict(f1.model, grid, se = T, inc = c("LDur", "LDur:RTime"))$fit
> grid$SE <- predict(f1.model, grid, se = T, inc = c("LDur", "LDur:RTime"))$se.fit
```

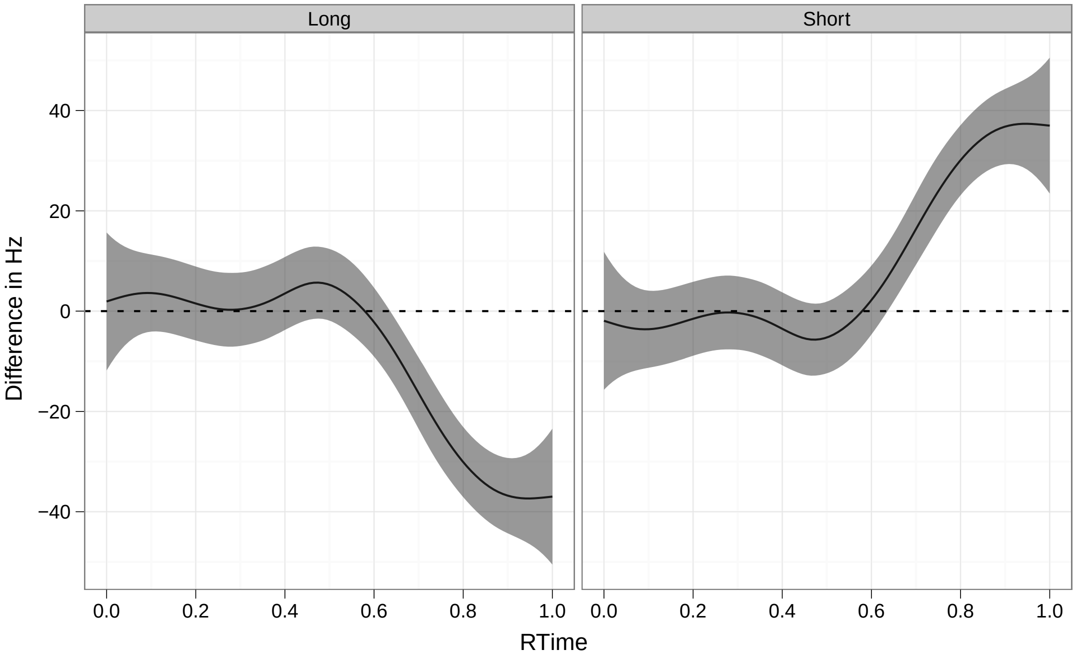

Figure 6: F1 Group + Interaction Effects

The results are plotted in Figure 6. In this plot, the values along the y-axis indicate the difference between the splines of the two groups (which is why they're mirror images). It looks as if F1 is not significantly different between the two groups until about 60% into their duration, at which point Long [ay]'s have significantly lower F1 values. This is essentially the same information given in the plot in Figure 2, but it's somewhat easier to see the size of the differences here.

Let's do the same thing for F2:

```
> grid <- expand.grid(RTime = seq(0, 1, length = 100), LDur = c("Long",
+     "Short"))
> grid$Fit <- predict(f2.model, grid, se = T, inc = c("LDur", "LDur:RTime"))$fit
> grid$SE <- predict(f2.model, grid, se = T, inc = c("LDur", "LDur:RTime"))$se.fit
```

Figure 7 plots the F2 group and interaction effects. Again, contrary to F1, F2 has significantly different formant dynamics between the long and short groups. In the area of the nucleus, the long group reaches a significantly lower F2 than the short group, and then in the glide, it reaches a significantly higher F2.

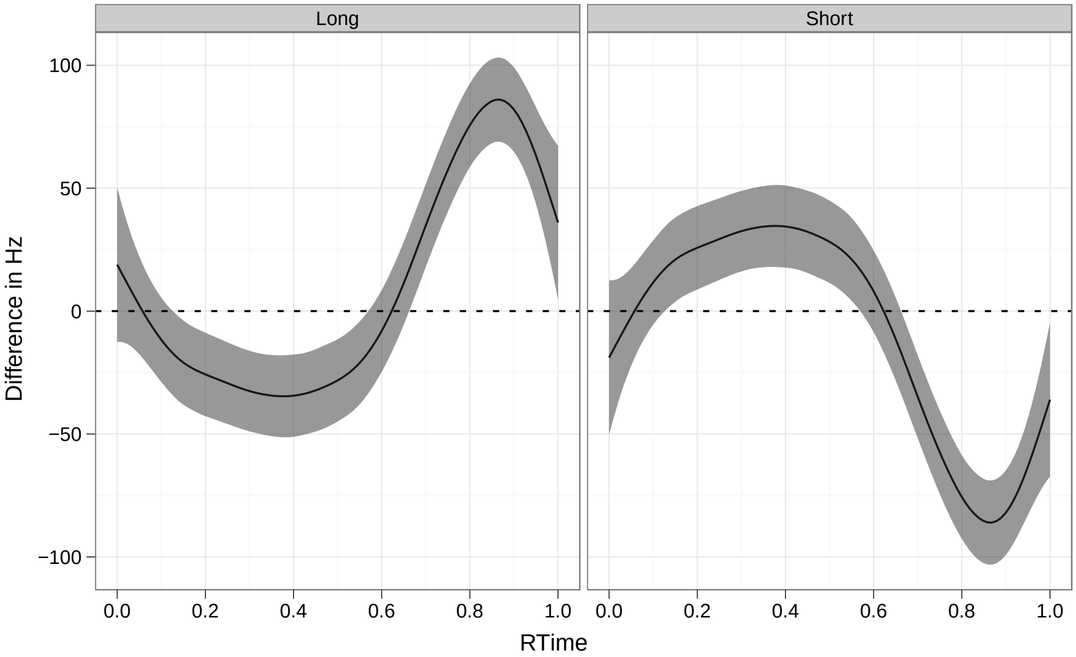

Figure 7: F2 Group + Interaction Effects

3 Movement through formant space

I've also plotted the "gestures" through formant space for each group. The coordinates for the x and y axes come from the fitted spline values. I've also tried to add confidence intervals, based on the standard errors of the x and y values at every point. It's obvious what I've done is not exactly right. There exist methods for plotting a sort of bubble area around the fitted values that represents the 95% confidence intervals in the 2D space, but I don't know how to do that yet.

The paths in Figure 8 are colored based on the timing of the gesture. They're completely blue at 0% through, completely red at 100% through, and purple at 50%. I've also plotted filled black dots at F1 maximum, and at F2 minimum and F2 maximum, since these are usually given as important landmarks in vowels for measurement. There's a filled black triangle at 1/3 through each gesture, reflecting where Keelan's automation would have measured.

4 Thoughts and Directions

I could do a better job of determining whether the significant differences in F1 and F2 dynamics between Long and Short groups is really the result of duration. I have not, in any way, controlled for contextual effects. The effects we saw of duration here may actually be linked some other important allophonic conditioning (my guess would be word final vs checked position) that is also correlated with duration.

My plan would be to first regress log(duration) against the various contextual factors of interest, and use the residuals from that regression as a standardized duration measure. Then, I would fit another ssanova model for each formant, probably with a formula like

`F1 ∼ Time + Context + ResidDuration + Context:Time + ResidDuration:Time`

and then inspect the ResidDuration+ResidDuration:Time effects. Fitting the ssanova model this way would have two benefits:

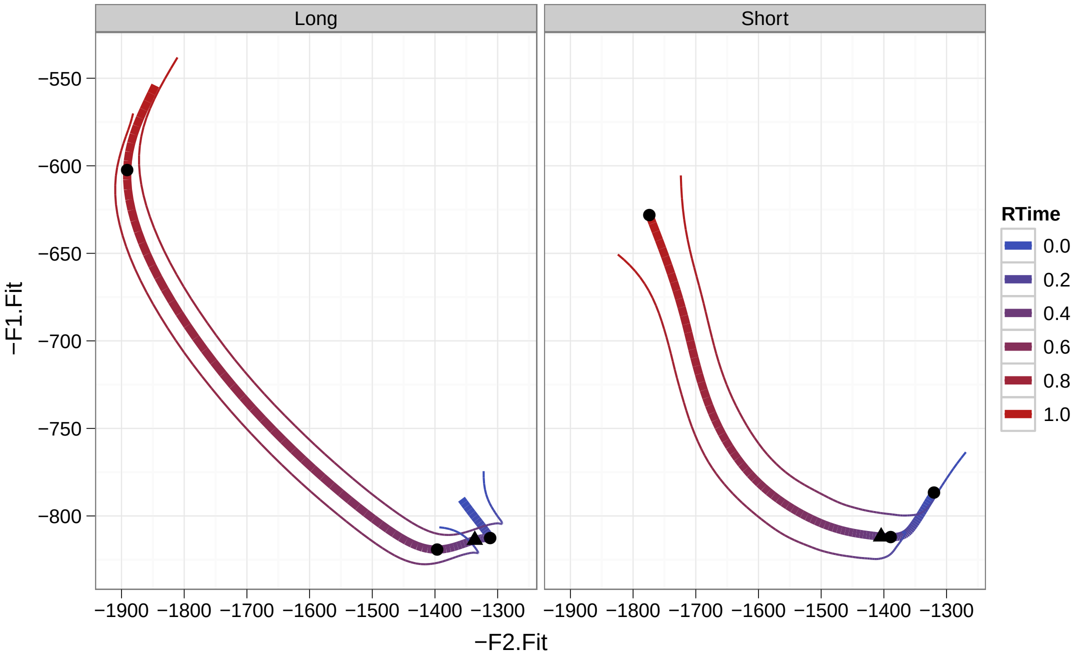

Figure 8: “Gestures” Through Formant Space

- Contextual effects would be controlled for, so remaining duration effects would (probably) be caused strictly by duration
- Fitting the duration element as a continuous variable ensures that not only would the formant curves over time be smooth, but so would the way they change as duration increases.

4.1 Bigger Questions

Accounting for dependancies in formant values would also be invaluable for this kind of analysis. Ideally, the result of doing this would be a normalized, square formant space. I have no ideas for how to go about doing this now.

A more practical question is one of methodology. Currently, when we want to analyze a person’s vowel system, we measure one or two time points in every vowel. The distributions of these individual points are typically important for research. Then, when we want to see mean values for these vowels, we take the mean F1 and F2 values of these point measurements. Would it be a better methodology to fit an ssanova for every vowel class, producing some kind of fitted average formant trajectory, and then take some measurement point from that spline as the mean value?

I do that here, briefly. First, for every vowel class, I compute a spline. The tracked formants and the splines computed for them (in red) are shown in Figure 9. As can be seen, there is considerable allophony not accounted for, especially in /oh/ and /ow/, and there are also considerable tracking errors, especially in /ey/.

```
> jean2 <- read.csv("jean2.csv")
> grid <- expand.grid(RTime = seq(0, 1, length = 100), VClass = levels(jean2$VClass))
> grid3 <- expand.grid(RTime = 1/3, VClass = levels(jean2$VClass))
> grid$F1.Fit <- NA
> grid$F2.Fit <- NA
> grid3$F1.Fit <- NA
> grid3$F2.Fit <- NA
> for (i in levels(jean2$VClass)) {
+     mod1 <- ssanova(F1 ~ RTime, data = jean2[jean2$VClass ==
+         i, ])
+     mod2 <- ssanova(F2 ~ RTime, data = jean2[jean2$VClass ==
+         i, ])
+     grid[grid$VClass == i, ]$F1.Fit <- predict(mod1, grid[grid$VClass ==
+         i, ])
+     grid[grid$VClass == i, ]$F2.Fit <- predict(mod2, grid[grid$VClass ==
+         i, ])
+     grid3[grid3$VClass == i, ]$F1.Fit <- predict(mod1, grid3[grid3$VClass ==
+         i, ])
+     grid3[grid3$VClass == i, ]$F2.Fit <- predict(mod2, grid3[grid3$VClass ==
+         i, ])
+ }
```

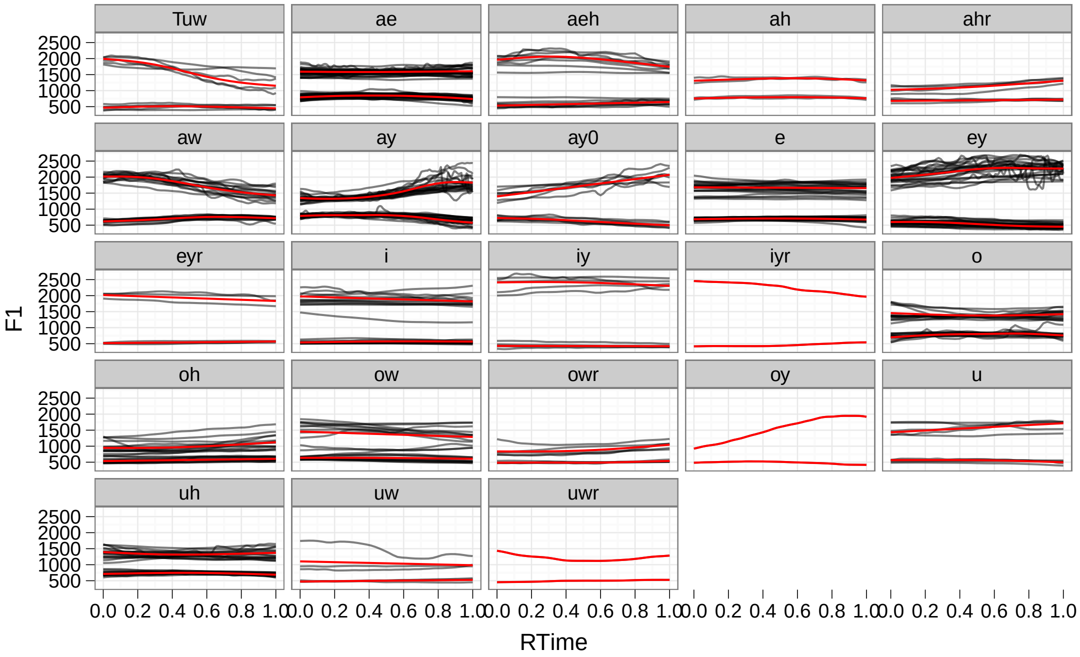

Figure 9: Tracked Formants with Splines

Next, I compare the mean values we get from averaging point measurements at 1/3 of every vowel, vs 1/3 of the smoothed spline for the vowel class. The comparison is given in Figure 10. It should be noted that the vowel classes where the two measurements are directly on top of eachother (/oy/,/iyr/,/uwr/) are the most *un*impressive examples, because there was only one token for those vowels, therefor the spline exactly matched the observed formant values.

```
> jean2$third <- abs(jean2$RTime - (1/3))
> thirds <- NULL
> for (i in levels(jean2$Word)) {
+     m <- min(jean2[jean2$Word == i, ]$third)
+     thirds <- rbind(thirds, jean2[jean2$Word == i & jean2$third ==
+         m, ])
+ }
> mF1 <- tapply(thirds$F1, thirds$VClass, mean)
> mF2 <- tapply(thirds$F2, thirds$VClass, mean)
> means <- data.frame(VClass = levels(thirds$VClass), mF1, mF2)
```

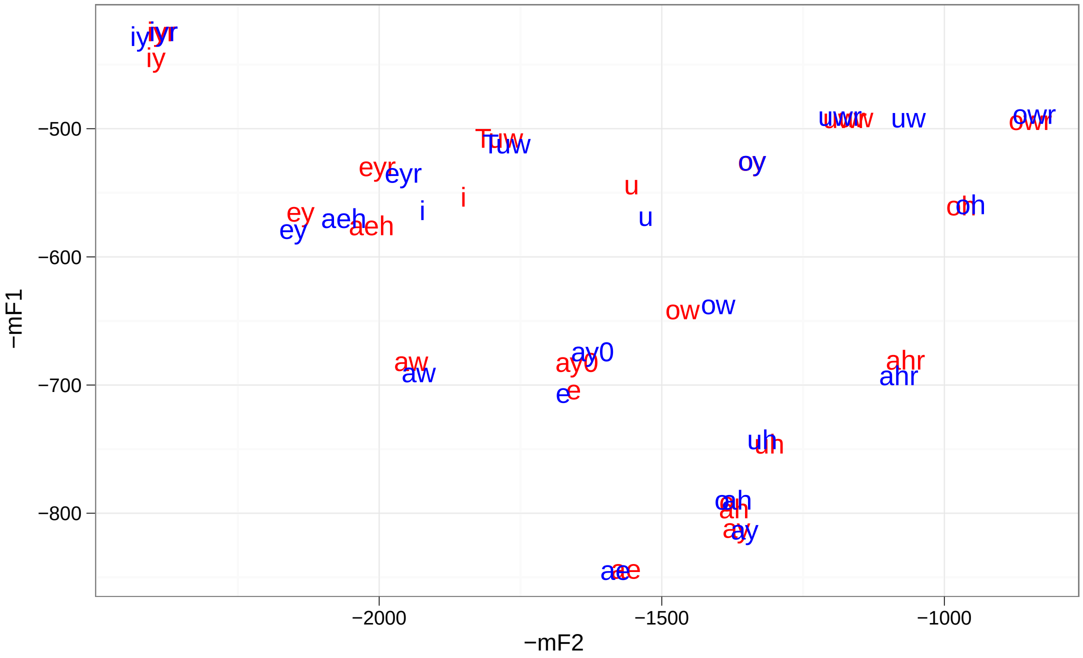

Figure 10: Comparison of Averaged 1/3 measurements(red) and 1/3 of splines(blue)

Is the spline method to mean values worth it? I don't know yet, but it at least doesn't give *bad* results.

References

Baker, Adam. 2006. Quantifying diphthongs: A statistical technique for distinguisthing formant contours. Paper presented at NWAV 35, Columbus, OH.

Davidson, Lisa. 2006. Comparing tongue shapes from ultrasound imaging using smoothing spline analysis of variance. *J. Acoust. Soc. Am.* 120:407–415.

Gu, Chong. 2002. *Smoothing spline ANOVA models*. Springer Series in Statistics. New York: Springer-Verlag.

Gu, Chong. 2009. *gss: General Smoothing Splines*. R package version 1.0-5.

Nycz, Jennifer, and Paul De Decker. 2006. A New Way of Analyzing Vowels: Comparing Formant Contours Using Smoothing Spline ANOVA. Poster presented at NWAV 35, Columbus, OH.

R Development Core Team. 2009. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org`, ISBN 3-900051-07-0.

Wang, Yuedong, and Chunlei Ke. 2009. *assist: A Suite of S-Plus Functions Implementing Smoothing Splines*. URL `http://www.pstat.ucsb.edu/faculty/yuedong/software.html`, R package version 2.0.

Wickham, Hadley. 2009. *ggplot2: An implementation of the Grammar of Graphics*. URL `http://CRAN.R-project.org/package=ggplot2`, R package version 0.8.3.