Mixed Models and Advanced Regression

Lecture Notes for STAT 505
Fall 2014

Jim Robison-Cox
Department of Mathematical Sciences
Montana State University

Contents

1 Introduction

These are notes for Stat 505, Linear Models, at Montana State University, Fall semester, 2014.

Objectives:

- We'll build basic stat computing skills in R and SAS.
- We will examine the "usual" assumptions of modeling:
  - constant variance,
  - independence, and
  - normality

  and examine ways to handle exceptions to each.
- We will work through hierarchical models in Gelman & Hill chapters 3 – 16 including logistic and Poisson regression and causal inference.

1.1 Review of Prerequisite Concepts

Prerequisites are Stat 422, Mathematical Statistics, and a regression course like Stat 410. From the latter you should have learned:

- To interpret computer output – in particular, to recognize and interpret
  - coefficient estimates, including slopes, adjustments for dummy variables, and interactions.
  - t-test for $H_0 : \beta_k = 0$ versus $H_a : \beta_k \neq 0$. Note: a test for the kth coefficient is conditional on all other predictors in the model.
  - F-test of the null hypothesis that all coefficients are 0 (except the intercept).
  - F-test (LRT) for a simpler model nested inside a more complex one.
- To diagnose non-normality through interpretation of normal quantile plot of the residuals. To use a power (Box-Cox) transformation to improve normality.
- To diagnose non-constant variance, and to address the problem using transformation or weighting. (Weights proportional to inverse variance.)
- To identify regression outliers and influential points.
- Variance of prediction (predicting a new observation for a specific row of the data matrix) is larger than variance of estimation.

1.2 Estimation and Prediction

We begin by asking why people fit linear (or nonlinear) models. Most models are fit for estimation or for prediction.

1. Estimation. The coefficient estimates are of interest. One may need a confidence interval for $E(y|\boldsymbol{x}_0)$ (where x_0 is a particular set of predictor values), or a confidence interval for a one-dimensional estimable linear contrast, $\boldsymbol{c}^\mathsf{T}\boldsymbol{\beta}$, or a confidence region for multidimensional estimable linear contrast, $\mathbf{C}^\mathsf{T}\boldsymbol{\beta}$. Of course, hypothesis tests about such quantities are also available, and might better fit the researchers' thinking.

2. Prediction. One needs an interval which will contain a new y_0 at particular value, $\boldsymbol{x}_0$, with given confidence.

Assume

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}; \;\; \boldsymbol{\epsilon} \sim (0, \sigma^2\boldsymbol{V})$$

Further, in order to make the inferences we're after, we'll also assume that errors have a Gaussian distribution (or that we have large enough n to appeal to the Central Limit Theorem).

Foundation: For estimable function $\mathbf{C}^\mathsf{T}\boldsymbol{\beta}$ where row rank of $\mathbf{C}^\mathsf{T}$ is m,

$$\mathbf{C}^\mathsf{T}\tilde{\boldsymbol{\beta}} \sim N(\mathbf{C}^\mathsf{T}\boldsymbol{\beta}, \sigma^2\mathbf{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\mathbf{C})$$

and we estimate σ^2 with the unbiased estimate $s^2 = SSE/(n-r)$. Note that this is a frequentist interpretation which takes the true $\boldsymbol{\beta}$ as fixed but unknown, and $\tilde{\boldsymbol{\beta}}$ has a sampling distribution about the true value. For a one dimensional $\boldsymbol{c}^\mathsf{T}$, it is straightforward to derive a $t(n-r)$ distribution for the standardized ratio $\boldsymbol{c}^\mathsf{T}(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta})/se(\boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{\beta}})$ where $se(\boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{\beta}}) = s\sqrt{\boldsymbol{c}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{c}}$. We use this distribution in testing individual coefficients for a full-rank model, and to get confidence intervals for such quantities. (there $\mathbf{c}^\mathsf{T}$ had a single 1 in it and the rest 0's.)

$$\hat{\beta}_k \pm t(1-\alpha/2, n-r) \times se(\hat{\beta}_k)$$

Another one-dimensional linear combination of interest is to let $\mathbf{c}^\mathsf{T} = \boldsymbol{x}_0$ be a row of the $\boldsymbol{X}$ matrix, either one of the observed rows, or if we dare to extrapolate, a row of values which could have been observed for $\boldsymbol{X}$, but were not. Using this $\mathbf{c}^\mathsf{T}$ yields a confidence interval for $E(y|\boldsymbol{x}_0)$ based on point estimate $\hat{y}_0 = \boldsymbol{x}_0\hat{\boldsymbol{\beta}}$.

$$\hat{y}_0 \pm t(1-\alpha/2, n-r) \times s\sqrt{\boldsymbol{x}_0(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0^\mathsf{T}}$$

To go to higher dimensions, we need some results from multivariate statistics which generalize from t ratios to quadratic forms.

$$\frac{(\mathbf{C}^\mathsf{T}\tilde{\boldsymbol{\beta}} - \mathbf{C}^\mathsf{T}\boldsymbol{\beta})^\mathsf{T}[\mathbf{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\mathbf{C}]^{-1}(\mathbf{C}^\mathsf{T}\tilde{\boldsymbol{\beta}} - \mathbf{C}^\mathsf{T}\boldsymbol{\beta})}{ms^2} \sim F_{m,n-r}$$

which can be inverted to give a confidence region in m dimensions. We are $(1-\alpha)100\%$ confident that the true $\mathbf{C}^\mathsf{T}\boldsymbol{\beta}$ lies in the region

$$\{\boldsymbol{\delta} : F^*(\boldsymbol{\delta}) \leq F(1-\alpha, m, n-r)\} \text{ where } F^*(\boldsymbol{\delta}) = \frac{(\mathbf{C}^\mathsf{T}\tilde{\boldsymbol{\beta}} - \boldsymbol{\delta})^\mathsf{T}[\mathbf{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^g\mathbf{C}]^{-1}(\mathbf{C}^\mathsf{T}\tilde{\boldsymbol{\beta}} - \boldsymbol{\delta})}{ms^2}.$$

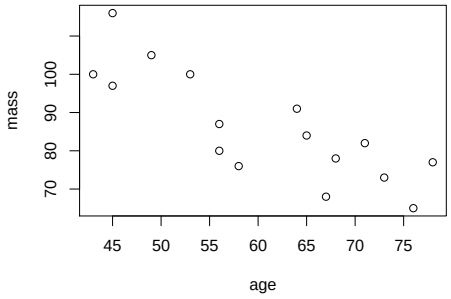
For a regression model, the confidence region is often used to find simultaneous $(1-\alpha)100\%$ confidence intervals for $E(y|x_0)$ for all values of x_0 across some range of interest. The intervals are of the same form as above with a $\sqrt{mF}$ multiplier in place of the t quantile.

$$\hat{y}_0 \pm s\sqrt{\boldsymbol{x}_0(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0^\mathsf{T} \times mF(1-\alpha; m, n-r)}.$$

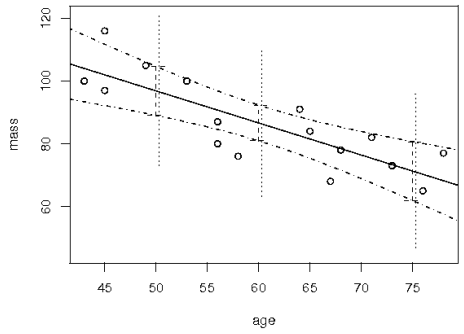
For single predictor models, the intervals can be plotted as a band named after Working and Hotelling. The W-H band is often shown around an estimated simple linear regression fit, but can also be plotted for polynomial regression where instead of $\boldsymbol{x}_0 = (1\ x_0)$ and $m = 2$, we would have an $\boldsymbol{x}_0$ vector of polynomials in x_0 and m would be one plus the order of the polynomial.

Note the basic difference between use of the t and the $\sqrt{mF}$: in the first case inference is desired about one particular $\boldsymbol{x}_0$ vector, while in the latter case, we are trying to make a statement about the entire range of x values, constraining the entire curve. Naturally the $\sqrt{mF}$ makes for a wider interval at a given x value, but a more interesting comparison would be to the problem of simultaneously estimating $E(y|x_i)$ for $i = 1, \ldots, k$ different x values. The Bonferroni approach is to split up the α level into k pieces using $t(1-\alpha/(2k), n-r)$ quantiles for each interval. Typically with k values of 3 or 4, the $\sqrt{mF}$ gives about the same width, and as k gets bigger, the Working–Hotelling band is narrower.

```
> muscle <- read.table("muscle.data", head=T)
> muscle.fit <- lm(mass ~ age, muscle)
> plot(mass ~ age, muscle, ylim =c(45,121))
> abline(muscle.fit)
> newx <- seq(40,80,length=50)
> (W <- sqrt(2*qf(.95,2,14)))
[1] 2.734554
>  qt(1-.025/3,14)
[1] 2.717755
> muscle.pred <- predict(muscle.fit, newdata =
                   data.frame(age=newx), se.fit=T)
>  muscle.pred ## partial output
$fit
     1      2      3      4      5      6      7      8
107.11 106.27 105.44 104.60 103.77 102.93 102.09 101.26

$se.fit
    1     2     3     4     5     6     7     8
4.375 4.241 4.108 3.977 3.846 3.718 3.592 3.468
> lines(newx, muscle.pred$fit - W*muscle.pred$se.fit, lty=4) #Lower  W-H band
> lines(newx, muscle.pred$fit + W*muscle.pred$se.fit, lty=4) #Upper  W-H band
> bonf.muscle.pred <- cbind(
+  predict(muscle.fit, newdata = data.frame(age=c(50,60,75)),
+                          interval="confidence",level=1-.05/3),
+  predict(muscle.fit, newdata = data.frame(age=c(50,60,75)),
+                          interval="predict",level=1-.05/3))
>  bonf.muscle.pred
          ##Confidence Int##          ##Prediction  Int##
       fit      lwr       upr      fit      lwr       upr
1 96.87121 89.08447 104.65795 96.87121 72.89524 120.84719
2 86.63532 80.96183  92.30881 86.63532 63.26006 110.01058
3 71.28148 61.92163  80.64133 71.28148 46.74944  95.81352

> arrows(c(50,60,75),  bonf.muscle.pred[,2], c(50,60,75), bonf.muscle.pred[,3],
      lty=2, angle=90,code=3, leng=.1)  ## 3 Bonferroni confidence intervals
```



```
> segments(c(50,60,75)+.3,  bonf.muscle.pred[,5], c(50,60,75)+.3,
     bonf.muscle.pred[,6], lty=3)      ## 3 Bonferroni prediction intervals
```



Prediction

If we wish to predict the y value for a new observation, we have two sources of variance. One is from the sampling distribution which affects the estimation of $E(y|\boldsymbol{x}_0)$ which we considered above, $\sigma^2\boldsymbol{x}_0(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0^\mathsf{T}$, and the second is the variation of a new observation, σ^2. The prediction interval is then:

$$\hat{y}_0 \pm t(1-\alpha/2, n-r) \times s\sqrt{1+\boldsymbol{x}_0(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{x}_0^\mathsf{T}}$$

In SAS you can obtain confidence intervals for the mean at each row of $\boldsymbol{X}$ with the option `CLM` after the model statement, and prediction intervals with `CLI`. Specify the desired confidence level in the call to the procedure with `ALPHA = .05`.

2 Statistical Software

2.1 Introduction

The language we use influences how we think. Inuit (Eskimo) reportedly has 28 different words for snow, whereas a language from a South Sea Island would have no such words. Similarly, the statistical package one uses can influence how one thinks about a statistical problem. Both R and SAS are capable statistical languages which can tackle almost any statistical problem (neither is missing some important word like "snow"), but for advanced programming the two languages differ dramatically. You already know that I use R more. I find that R better fits my way of thinking about statistical problems. I've also invested a lot of time in exploring the features of R. SAS has the advantage in settings where the same analysis is run on many similar data sets and it large data sets.

The origins of SAS go back to the early seventies (think of punch cards as the computer input mechanism) when it was begun at North Carolina State University, and it became a private venture in 1976. Historically, it has seen heavy use in the pharmaceutical industry, for instance when a drug company submits information about a new drug to the FDA, SAS datasets are a big part of the submission.

S was developed at ATT-Bell Labs, mainly by John Chambers and Trevor Hastie. ATT-Bell did not want to get into software support, and sold the rights to StatSci where it was improved and dubbed Splus. StatSci merged with MathSoft, then took the name Insightful, Inc, and in Sept 2008 was purchased by TIBCO.

In the early 1990's, Ross Ihaka and Robert Gentleman at University of New Zealand, Auckland decided a freeware version of S was needed, and they began development of R as another implementation of S. They released it to the Freeware (meaning no strings attached) community in 1996, and developers around the world are now contributing and improving it constantly. There are over 4000 contributed packages available on the CRAN (comprehensive R archive network) repository. For a while, it looked like Splus might survive as a professionally supported version of R, but certainly R now dominates the S world.

If you want to install either commercial package on your own PC, PC–SAS costs $120 annual rental from the ITC microstore and Tibco offers a free 30 day trial of Spotfire S+. I assume you already have installed R on your own computer.

We will be running SAS "On Demand" which uses a SAS owned server in the cloud.

2.2 SAS Intro

SAS divides its commands into two types. *DATA* steps are used to input and modify data and *PROC* steps are procedures which print, plot or perform tests on data. *PROC*s can also output datasets, for instance `PROC REG` allows one to output a dataset of regression coefficient estimates and allows one to output residuals and fitted values. In a typical SAS dataset, each row refers to a single unit, and each column is a variable (numeric or character). However, I sometimes save space on paper by putting several rows of data in one line. When a line contains more than one unit's data, append a `@` to the end of the list of variable names to tell SAS not to go to the next row until it's read to the end of the line.

2.2.1 SAS Help

SAS has built–in help screens which can be called up in another window when SAS is running, but I find the Web-based help easier to navigate (and it's searchable). To find links to SAS help, go to the Stat 505 or 506 webpage and click on "SAS", then either of the documentation links.

SAS help pages are quite lengthy. For each PROC you will see Overview, Syntax, Details, Examples, and References. There are other sections which list functions available to modify data and list functions for data input. I often go to the examples first to see if I can modify one of them for my analysis.

2.2.2 SAS Data Input: DATA steps

The `INPUT` command is used to define the names and types of variables. A dollar sign ($) is used to indicate that the preceding variable is a character variable. We will use `INPUT` with either the `datalines` (formerly `cards`) statement or with `INFILE`. The difference is just whether the data is shown in the command lines (use `datalines`) or contained in a separate file (use `INFILE`) in either case, I'll assume the data are separated by spaces and that each row represents a single subject or unit. Example of `datalines`:

```
DATA scores;
  INPUT name $ ID test1 test2;
  DATALINES;
  Joe  1234  95 87
  Jill  1497 85 94
```

```
  Terry  7546 84 88
;
RUN;
PROC PRINT;
RUN;
```

Note that SAS command lines end with a semicolon, but there is no punctuation within the data lines. A semicolon is the signal for the end of data entry. SAS is case-insensitive about commands, data set names, and variable names.

Example of `INFILE`:

```
DATA scores;
  INFILE "data/student.data";
  INPUT name $ id test1 test2;
RUN;
PROC PRINT;
RUN;
```

Assuming the file referred to with `INFILE` contains the same data shown after `DATALINES`, the two methods are equivalent. Each piece of code creates a dataset with three rows and 4 variables (columns). There are many more complicated variations on this simple input scheme, see the SAS documentation for `INFILE` and other commands in the help on Base SAS. In order to use the same file for both R and SAS, I like to put the variable (column) names in the first row and use the `FIRSTOBS =2` option on the `INFILE` line.

```
  infile "data/student.data" firstobs=2;
```

Missing Data: Missing numeric data should be coded as a period in the data file or in the data lines after `DATALINES`.

SAS Storage: Where does the dataset live? After either of the above creation steps, SAS puts a file called `scores` in a temporary directory. SAS erases the directory and all working data files when you exit. One must always save the code used to build data, so that data removal is not a problem.

Sequencing: A `DATA` step is computed as a loop – the sequence of commands in the data step is performed over and over, once for each row of data.

Possible Problems: First, make sure that each command ends with a semicolon. Once I've found those mistakes, usually my problems result from inconsistently typed data files. If there aren't as many bits of data on a line as there are variables in the input statement, then SAS goes to the next line and looks for more there (see `MISSOVER` for one way to catch this). For example, if we read state names like *Montana*, and *North Dakota* into SAS, you might want to change to another delimiter like comma instead of space, or include the names in quotes, or replace internal spaces with underscores as in *North_Dakota*. That's also why you should stick in a `PROC PRINT` statement after reading data, or for a big file, use `PROC UNIVARIATE` to get summaries on each variable.

2.2.3 SAS Regression Procedures

`PROC REG` works for regression where the $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{V}^{-1}\boldsymbol{X}$ matrix is non-singular, and can be used for ANOVA when you create dummy variables for each factor so that if factor A has a levels, $a-1$ dummy variables are used. `PROC ANOVA` does analysis of variance for balanced numbers (same number of observations in each cell). `PROC GLM` is used for unbalanced ANOVA and when fitting a mixture of factors (classification variables) and continuous

predictors (Analysis of covariance, ANCOVA). Why are there three choices when `PROC GLM` can handle all situations? The other procedures are more efficient for their specialized cases, so if you have a huge dataset and need to run regression, `PROC REG` will be faster. The letters GLM stand for "general linear model", which is different from a general**ized** linear model.

Commands

The list of options for SAS procedures is long, you need ready access to the manual, and/or practice at using the SAS online help. The following example has enough details to be interesting:

```
006  PROC GLM  DATA=books;
007    CLASS binding;
008    MODEL price = pages  binding;
009  RUN;
```

Line 6: after invoking a procedure, it's a good habit to write `DATA=` the name of the dataset you want to use. If you leave it out, SAS uses the last dataset created. Usually that is what you want, but the exceptions will cause trouble when you forget.
Line 7: I am using two predictors for the price of a book, pages which is numeric, and binding ("h" = hardcover, "p"=paper) which is categorical. Categorical factors have to be declared as class variables. If you forget, this run would give an error since binding is a character variable (obviously not numeric), but if you had treatments numbered 1 thru 4, forgetting a class statement would be equivalent to running regression on the treatment numbers (which are usually meaningless).
Line 8: the model statement tells SAS what predictors to use, their ordering, and what interactions to use. Here are some options for interaction:

`MODEL price = pages binding` will fit the data to two lines, one for each binding, with the same slope for pages.

`MODEL price = pages binding pages*binding` will fit the data to two lines with different slopes and intercepts. A shorter way to ask for main effects and interactions is the equivalent `MODEL price = pages|binding`
but this works only in GLM or ANOVA, not REG (build your own interaction variable.) In ANOVA models we sometimes need the "nested" notation

`MODEL y = A(B)` for instance, if y is yield of wheat per acre, factor A is State (North Dakota, Montana, South Dakota, Wyoming) and factor B is County (counties differ between states, even though two states might both have a "Garfield" county, it's not the same county in any sense).

The results of a regression procedure are automatically printed unless the noprint options is used in the first line. Procedures can also create datasets, for instance of coefficients, residuals, predicted values, or diagnostic measures. Generally this is requested after the model statement, and before the run statement:

```
OUTPUT out= fit_data p=yhat r=resid;
```

The dataset created includes the original data columns as well as the other quantities specified. Another dataset can be created to store the F-tests for each term in the model by requesting `OUTSTAt=`*filename* in the first line. In `PROC REG` there is also an `OUTEST=`*filename* option for the first line which allows one to save the regression coefficient estimates.

For more comparison of regression procedures, see the SAS documentation "Introduction to Regression Procedures" (linked from the Stat 506 page under SAS help.

2.2.4 Plotting in General, and in SAS

> The three rules of data analysis are 1. Plot your data, 2. Plot your data, and 3. Plot your data. *J. W. Tukey*

I'm a firm believer in the idea that **every** statistical inference should be accompanied by a plot to verify that the inference is not only "significant", but also plausible. Many times I've seen a statistical inference which looked impressive in terms of p-value, but was wrong when I looked at the data. You'll see what I mean in the homework.

PROC PLOT turns out crude "line printer" plots in the Output window. You can get similar output by including a `PLOT` option in `PROC REG` These are fast and useful for getting a rough picture of relationships, but use `PROC GPLOT` whenever possible, since it gives much better resolution.

I gave up on GPLOT and SAS/Graph about 20 years ago, when I found out how easy it is to plot in Splus. SAS does now offer quick high quality plots under the following choice of menus: ßSolutions: Analysis:Analyst

2.3 R Intro

For our purposes, the easiest way to run R will be using RStudio. Download it from Rstudio.com. Alternatively, you can run R from within emacs using ess (emacs speaks statistics) or, on Windows, eclipse. I will also require you to use knitr (Xie (2013)), so you need to install LaTeX.

R code creates objects which have a class. Classes include: function, numeric, factor, list, data.frame, matrix, and many other types. R is "object oriented" which means 1) objects may have attributes, 2) attributes can be passed on to a new object when it is created, 3) different methods exist to handle objects with a particular attribute. For example, `summary` is a function for printing important parts of objects. If I ask for `summary(fit)` the summary function checks for attributes of the object `fit` to find out what sort of object `fit` is, and gives an appropriate summary. For the novice R user, object orientation just means that R is "smart" about handling objects and tries to give logical summaries, plots and printouts.

Note that R is sensitive to case, `X1` and `x1` are different variables, but SAS figures that they're the same.

2.3.1 R Help

Rstudio does a nice job of organizing help windows, and I recommend it for that.

You need to become familiar with the contents of a typical R help page. Help for a function contains

- Description
- Usage with all argument defaults

- Arguments, explaining what each argument holds
- Details, which could get technical
- Value: describing what the function returns.
- References
- See Also
- Examples

As with SAS, the examples can be very informative.

2.3.2 Exiting R

To close an R session. Type `q()` (or close the window) and R will ask if you want to save the image for further analysis. The image is all data you read in plus any fitted objects you created plus any functions you built. It is very important that you save the **code** you use to produce an anlysis. Saving the image is a good idea if you did some data processing which took a long time. Otherwise, saving can clutter your workspace with extra copies of objects. That is a good reason to use descriptive names, not just "x" and "y". If you tell it to save the workspace, it saves a file called `.RData` in the current working directory, and will load that file when you restart R in that directory.

2.3.3 R Data Input: c, scan, read.table

R uses function for data creation and for reading files. By assigning the returned output of the function we can store the new data. The simplest data creator is the `c` function.

```
Names <- c("Joe","Jill","Terry")
ID <- c(1234,1497,7546)
test1 <- c(95,85,84)
```

The arrow, composed of "less than" and "minus" is used to assign values to a variable (`=` also works). As created above, `ID` and `test1` are vectors, but not 3×1 matrices. Here's a way to force the ID numbers into a column vector so that it has dimension 1 by 3.

```
ID <- matrix(ID,1,3)
```

If you want to build up an X matrix by combining columns, use `cbind` to bind columns together (or `rbind` to combine rows). I'll throw in the `rep` command here also to create the three ones needed for the intercept column.

```
> X <- cbind(rep(1,3),ID,test1)
> X
         ID test1
[1,] 1 1234    95
[2,] 1 1497    85
[3,] 1 7546    84
```

The coercion into a vector using `cbind` will work whether or not `ID` has matrix dimension attributes. (But we would get a warning if the pieces we're `cbind`ing are not all of the same length, or if one was a non-conformable matrix.) That's not quite true because of the recycling rule explained under "Comparisons", below.

Scanning in numeric data. One can do any analysis with these simple definitions of data using `c()`. However, we will usually have data files to read. One way to read in a ***numeric*** data file is with the `scan` command, then manipulating the result with `matrix`.

```
X <- scan("student.data")
X <- matrix(X,ncol=3,byrow=T)
```

Or, in one step:

```
X <- matrix(scan("student.data"),ncol=3,byrow=T)
```

The `byrow=T` option is needed here because R fills in the matrix one column at a time, unless you tell it to fill in row-by-row. One problem with the `matrix(scan(...))` approach is that matrix cannot handle non–numeric data, such as the names in the student file. The R answer is the data frame, which is similar to a SAS dataset. Again, I am simplifying. You can use `scan` to read in character data if you add the argument `what = "A"`.

Using read.table and data.frame A data frame in R is a list of columns, each of the same length (`nrow`). Here's how to create a data frame:

```
scores <- data.frame(name=Names, id=ID, score1=test1, score2=c(87,94,88))
```

We now have a dataframe called scores. To go directly from a data file to a dataframe, use the `read.table` function:

```
scores <- read.table("data/student.data", header=T)
```

I like to store column names in the first line of the file and use the option `header=TRUE` to retrieve them. (Another version of `read.table` is `read.csv` which assumes the columns are separated by commas, and that there is a header row.) If column names are not in the file, you can put them in as an option of `read.table` or let R use its default names: V1, V2, etc.

The use of `read.table` or `data.frame` automatically creates factors from character variables. That is, character data is assumed to be a label of treatments or strata, and a set of dummy variables is created in case the factor is to be used in anova (regression on dummy variables). To avoid singularities in the X matrix, the dummy variables are created with one fewer columns than the number of unique identifiers in the dataset. More on this later, but be warned that something has happened which you didn't ask for.

Missing Data: Insert the characters "NA" for missing data, or you can use a character like "`*`" and the na.string option:
`read.table("data/student.data", header=T, na.string="*")`.

Storage
R differs from Splus in data storage in that the data is stored in memory rather than on the hard drive. An advantage is that computation is usually much faster for R, but a disadvantage is that R is not able to handle really large data sets.

To organize your data workspaces, I suggest that you create a separate folder for each class or project, and start R in that folder when working on that class or project. In unix you can use the file browser to build folders or the command line:

```
mkdir stat506 consulting
```

In Windows, again, build folders for each class or project. You can copy the R icon on the desktop and rename it R-stat506, then change its properties to have it start in the stat506 folder.

Possible Problems When using `read.table`, R first counts the entries in each line of the file to see if all have the same number of databits. If some state names have a space in the middle, R will think there are too many fields on those lines. You could modify the data, as I suggested for SAS above, but a better solution is to use a comma or some other character as a separator. Then include the argument `sep=','` in the `read.table` function. As with SAS, it's important to print the dataframe (just type the name and enter) or use univariate summaries as in `summary(scores)` and look for mistakes.

2.3.4 R Regression Commands

The `lm` function is equivalent to the SAS `PROC GLM` in that it handles all linear models. You could use `aov` to fit ANOVA models, it handles balanced and unbalanced models.

Usually the output of a call to `lm()` is stored as an object so that parts of it can be used as needed. For example, to fit the same model as above:

```
bookfit <- lm(price ~ pages + binding, data=books)
```

Note the form `y ~ x1 + x2`, means y "is modeled by" main effects for `x1` and `x2`. To get an interaction the syntax is `x1:x2`, shorthand for main effects plus interaction is `x1*x2`, and nested factors are expressed as `B %in% A` or `A/B` which is equivalent to writing `A + B %in% A`. The `data=books` specification tells R which data frame to use. As in SAS, it's good practice to always specify the data frame. If it is omitted, R looks in the data directory for the price, pages, and binding variables (and in this case would fail to find them since their names are really books$price, books$pages, and books$binding unless you attached the books dataframe as in `attach(books)` first).

After fitting the `lm()` model, the object `bookfit` contains everything you might want to know about the regression fit, but you do have to ask to see the results. Just typing the name always prints (some version of) an object, so you could type:

```
bookfit
```

A better summary is given by the summary command `summary(bookfit)` which prints most of the output you saw in the `PROC GLM` output. To see the anova table for this model type `anova(bookfit)`. Note that these are **sequential** type I sums of squares (S builders don't believe in the indiscriminate use of type III sums of squares).

Other bits you might want from a linear model object are the coefficients: `coef(bookfit)`, the fitted values: `fitted(bookfit)`, and the residuals: `resid(bookfit)`. Each of these pieces will be used in plots to evaluate the effectiveness of the fit. There is a special plot method for fitted models, so you could try

```
par(mfrow=c(2,3))
plot(bookfit)
```

to see what it will show. We'll talk about these default plots when we do regression diagnostics.

2.3.5 R Plotting Commands

R will automatically open a plotting window when you first request a plot. You can open another window if you don't want a later plot to destroy an earlier one. Use `x11()` on Unix or `windows()` in windows. To save a plot, I like to use pdf files for LaTeX or png or jpeg files for a word processor. See the `dev.copy` command to copy graphical window contents to a file.

Univariate Plots The commands `boxplot(x), stem(y)`, and `hist(z)` will create boxplots, stem–and–leaf plots, and histograms. The `qqnorm()` command is useful for comparing a sample to the expected values of the normal distribution.

Bivariate Plotting The basic command is `plot(x,y)` which has options to set plotting characters, line types (dashes), axis labels, and titles. After getting a plot started you can add to it with the commands `points(), lines(), abline(), title(), legend()` and more. Look at help on the above commands and on `par()` which sets graphics parameters.

2.3.6 Reproducible Research

Problem:
If I return to a project which I worked on a few months ago, it's hard to pick it back up again and remember which files I was using and which objects I need to use again.

In writing a journal article, it might be months before I am asked a question by a reviewer, and it's hard to reconstruct the analysis. A basic tenant of the scientific method is that research must be reproducible, but we often don't do a good job of making the statistical analysis reproducible. There is a movement in computer science and in stat computing to change that behavior and make research reproducible. It also organizes the code and analysis into a single document.

Setup

- Install R and Rstudio (or emacs)
- Install LaTeXIn unix it's texlive in Windows: MikTex, but you can just install Temaker or TexWorks to get the full LaTeX plus an editor front-end.

There is a method called odfweave for sweaving into OpenOffice (now LibreOffice, a free replacement for MS Word) and one to produce html code for a web page. You're welcome to explore these, but I will assume we're all using LaTeX and knitr.

Advantages of LaTeX:

- It keeps track of figures, tables, page numbers, sections, equations, and lists with or without numbering. If you change the order of items, numbers are automatically adjusted.
- In conjunction with BibTeX, it keeps track of citations and makes it really easy to create a bibliography.
- It is great for creating mathematical equations.
- If you are considering doing a PhD, LaTeX is the best way to write a thesis in our department. I recommend it for your writing project as well.

As with any software, you must invest some time working with LaTeX to get proficient. In the beginning you will spin your wheels a lot, but soon it will increase your productivity. I recommend that you use it to create a quiz or homework for your class. Ask your course supervisor and other instructors if they have templates you can modify.

There are lots of good tutorials on LaTeX on the web.

A knitr file ends with .Rnw and contains mostly LaTeX with some R code. We first process the file with R to execute the code and build the LaTeX file, then "latex" the resulting file to produce the output. In Emacs this is two steps. Rstudio does it with a single mouse click on the "Compile PDF" icon just above the code window.

Most of the file is LaTeX code. For example the first lines tell it what type of document to create (report, letter, article, ...) and set margins and load added packages. All text is included between `\begin{document}` and `\end{document}`. Any word that begins with a \ is a special LaTeX code and is not printed, but instead it modifies the output. Dollar signs are used to begin and end "math mode", so if you want to actually use a $ you type it as `\$`. Similarly the percent sign has a special meaning - comment – meaning ignore the rest of this line. So to talk about a 95% CI, I type `95\% CI`.

Rcode is included inside a special environment beginning with `<< >>=` in column 1 and ending with `@` in column 1. We can include these options inside the `<< >>=` symbols:

1. A name for the code chunk, then a comma. This is not required, but might be more readable than the default "chunk 1", "chunk 2", etc.

2. Do you want the code to be printed? If so type `echo = TRUE,` otherwise `echo = FALSE,`. The default is TRUE.

3. Do you want the results printed at this point in the document? If not add `results = 'hide',` or `results = 'asis',` for a LaTeX table.

4. Does the code produce a figure? If so add you can set dimensions of the image as stored in a file and of the image as printed on the page. For the file, set dimensions in inches `fig.height = 4, fig.width=5,` to get the aspect ratio you want. Then set the size on the page with `out.width='.5\\linewidth'`, for example, to make the figure use half the page width. If fonts look too small, decrease fig.width and fig.height.

knitr process:

1. Process the .Rnw file running all R code. Use this to create a .tex file and save plots into .pdf files.

2. Run pdfLaTeX on the .tex file to create the document.

To print some R expression use a `\Sexpr{}` statement like "slope is `\Sexpr{round(coef(it)[2], 2)}`".

I recommend using an options chunk at the beginning of each .Rnw file to set some default sizes. Mine looks like this:

```
<<setup, include=FALSE, cache=FALSE>>=
  require(knitr)
  opts_chunk$set(fig.width=5, fig.height=4, out.width='\\linewidth', dev='pdf', concordance=TRUE)
  options(replace.assign=TRUE,width=112, digits = 3, max.print="72")
@
```

Comments on code writing: It helps to indent the inside of each code chunk, and to further indent the inside of a function definition or a for loop. Use blank lines to separate major parts of the code.

2.4 Estimability and Software

Basics:

- A linear combination of coefficients, $\boldsymbol{c}^\mathsf{T}\boldsymbol{\beta}$ is not estimable if $\boldsymbol{c}$ is not in the column space of $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$. If $\boldsymbol{X}$ is of full column rank, as in a regression setting (explanatory variables are continuous), all such linear combinations are estimable (unless one column is exactly a linear combination of the others).

  The remainder of this section deals with cases where some or all predictors are categorical (ANOVA and ANCOVA settings) and $\boldsymbol{X}$ may be less than full column rank.

- When a predictor variable is categorical and we use a "treatment effects" model then $\boldsymbol{X}$ is not of full column rank. As a simple example, the one-way ANOVA effects model

$$y_{ij} = \mu + \tau_i + \epsilon_{ij};\ i = 1, \ldots, I,\ j = 1, \ldots, n_i$$

  has $I+1$ parameters for the means and the individual $\mu, \tau_1, \ldots, \tau_I$ are not estimable. A "cell" consists of the data which share the same level of the treatment(s), and an empty cell occurs if the particular treatment was never applied (or observed). If no cells are empty, then we could use the cell means model with $\mu_i = \mu + \tau_i$, and each cell mean is estimable.

$$y_{ij} = \mu_i + \epsilon_{ij};\ i = 1, \ldots, I,\ j = 1, \ldots, n_i;$$

  $E(\bar{y}_i) = \mu_i$, implies estimability of μ_i. With either of these models, differences in treatments $\tau_i - \tau_{i'} = \mu_i - \mu_{i'}$ are estimable for not-all-missing cells i and i'. If $\boldsymbol{c}$ is a vector of length I which sums to zero, then

$$\boldsymbol{c}^\mathsf{T}\boldsymbol{\mu} = \boldsymbol{c}^\mathsf{T}\begin{pmatrix}\mu_1\\ \mu_2\\ \vdots\\ \mu_m\end{pmatrix} = \boldsymbol{c}^\mathsf{T}\begin{pmatrix}\mu+\tau_1\\ \mu+\tau_2\\ \vdots\\ \mu+\tau_m\end{pmatrix} = \mu\sum c_i + \boldsymbol{c}^\mathsf{T}\begin{pmatrix}\tau_1\\ \tau_2\\ \vdots\\ \tau_m\end{pmatrix} = 0 + \boldsymbol{c}^\mathsf{T}\begin{pmatrix}\tau_1\\ \tau_2\\ \vdots\\ \tau_m\end{pmatrix} = \boldsymbol{c}^\mathsf{T}\boldsymbol{\tau}$$

  which demonstrates that the same contrasts are estimable under either model.

  Similar results hold for two-way, multi-way, and analysis of covariance models (where at least one predictor is continuous and at least one is categorical). Coefficients for the continuous variables will be estimable (as long as the columns involved are linearly independent), and cell means and contrasts thereof will be estimable as long as we have some data in each cell.

- Trivial cases of estimability problems are generally avoidable, for instance someone might ask you to provide estimates of the τ_i's in the effects model, not realizing that they are confounded with μ. In such cases, the statistician has to explain what the τ_i's represent and determine what the consumer really wanted, perhaps the μ_i's.

  You might wonder why we don't just stick with the cell means model. In the tough cases with missing data and unbalanced data the cell means model actually gets more complex and confusing than the effects model. Focusing on the means model tends to make people complacent about estimability.

- When data is unbalanced (as in virtually all observational studies) the tough question is "What do we want to estimate?" When that has been answered, we then have to determine if it is estimable.
  Example:
  A sample of Montana residents is selected in order to estimate difference in mean age between rural and non-rural residents of the state. The following mean ages were obtained as well as information on location – east or west of the center of the state.

	Mean Ages	
	East	West
Rural	$\bar{y}_{11}$	$\bar{y}_{12}$
Urban	$\bar{y}_{21}$	$\bar{y}_{22}$

	Sample Sizes	
	East	West
Rural	n_{11}	n_{12}
Urban	n_{21}	n_{22}

	Population Sizes	
	East	West
Rural	N_{11}	N_{12}
Urban	N_{21}	N_{22}

  What would be a reasonable estimator of the difference in mean age for urban versus rural? One option is $\widehat{D}_1 = (\bar{y}_{11} + \bar{y}_{12})/2 - (\bar{y}_{21} + \bar{y}_{22})/2$, another $\widehat{D}_2 = (n_{11}\bar{y}_{11} + n_{12}\bar{y}_{12})/(n_{11} + n_{12}) - (n_{21}\bar{y}_{21} + \bar{n}_{22}y_{22})/(n_{21} + n_{22})$. Which is better? Assumptions?

Non-estimable Constraints

Many textbooks on linear models, when presenting ANOVA models say that the effects are constrained, for instance in that $\sum \tau_i = 0$. With a one-way ANOVA model, any one-dimensional constraint on a non-estimable quantity is enough to make all other parameters estimable, for example: $\sum \tau_i = 0$ or $\tau_1 = 0$ or $\tau_I = 0$ or $\mu = 0$, since each of those parameters is non-estimable, and the difference in number of columns of $\boldsymbol{X}$ and the column rank of $\boldsymbol{X}$ is one. (Monahan (2008) §3.8)

General constraints:

$$\boldsymbol{K}^{\mathsf{T}}\boldsymbol{\beta} = \boldsymbol{\delta}, \text{ where, for consistency, } \boldsymbol{\delta} \in \mathcal{C}(\boldsymbol{K}^{\mathsf{T}}) \tag{1}$$

The constraint chosen does not uniquely determine the coefficient estimates (unlike choice of a generalized inverse). In the next paragraph, I will assume $\boldsymbol{\delta} = \mathbf{0}$, which can be done WLOG because $\exists\ \boldsymbol{\beta}_0$ such that $\boldsymbol{K}^{\mathsf{T}}\boldsymbol{\beta}_0 = \boldsymbol{\delta}$ (based on $\boldsymbol{\delta}$ living in the row space of $\boldsymbol{K}$). We can work with the translated system:

$$\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}_0 = \boldsymbol{X}(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\beta}^{**} + \boldsymbol{\epsilon} \tag{2}$$

where $\boldsymbol{K}^{\mathsf{T}}(\boldsymbol{\beta} - \boldsymbol{\beta}_0) = \mathbf{0} \Longleftrightarrow \boldsymbol{K}^{\mathsf{T}}\boldsymbol{\beta} = \boldsymbol{\delta}$.

Our reason for translating is to obtain a reparameterized design matrix, $\boldsymbol{X}^*$, which is of full column rank by combining or deleting some of the columns of $\boldsymbol{X}$, and similarly reducing $\boldsymbol{\beta}$ to $\boldsymbol{\beta}^*$.

$$\boldsymbol{y} = \boldsymbol{X}^*\boldsymbol{\beta}^* + \boldsymbol{\epsilon} = [\boldsymbol{X}\mathbf{C}][\mathbf{R}\boldsymbol{\beta}] + \boldsymbol{\epsilon} \tag{3}$$

where $\mathbf{C}$, $p \times r$, is of full column rank, and $\mathbf{R}$, $r \times p$, is of full row rank. Matrices $\mathbf{C}$ and $\mathbf{R}$ relate to the constraint matrix, $\boldsymbol{K}$, in that $\mathbf{CR} = \boldsymbol{I} - \boldsymbol{K}^{\mathsf{T}}(\boldsymbol{K}\boldsymbol{K}^{\mathsf{T}})^g\boldsymbol{K}$ which projects to the null space of $\boldsymbol{K}$. Again, $\mathbf{C}$ and $\mathbf{R}$ are not unique, there are many different reparameterizations which all yield equivalent restricted estimates. When $\boldsymbol{K}$ contains $q = p - r$ independent constraints, the transformed $\boldsymbol{X}^*$ is of full column rank, so $(\boldsymbol{X}^*)^{\mathsf{T}}\boldsymbol{X}^*$ is nonsingular, and a unique estimator under the constraints is $\widehat{\boldsymbol{\beta}^*} = [(\boldsymbol{X}^*)^{\mathsf{T}}\boldsymbol{V}^{-1}\boldsymbol{X}^*]^{-1}(\boldsymbol{X}^*)^{\mathsf{T}}\boldsymbol{V}^{-1}\boldsymbol{y}$. The parameter estimate in the restricted space can be back-transformed to the original space to get $\widehat{\boldsymbol{\beta}}_r = \mathbf{C}\widehat{\boldsymbol{\beta}}^*$ where the subscript denotes an estimate of $\boldsymbol{\beta}$ which conforms to

the constraints. This $\widehat{\boldsymbol{\beta}}_r$ can also be obtained by using the $\boldsymbol{C}(\boldsymbol{C}^\mathsf{T}\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X}\boldsymbol{C})^{-1}\boldsymbol{C}^\mathsf{T}$ as the generalized inverse in $\widetilde{\boldsymbol{\beta}} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^g\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$.

Computing Packages and ANOVA models

- In **SAS**, `Proc GLM` needs a `class` statement to identify categorical predictors. It is then easy to obtain estimates of contrasts. These are based on and specified through the full effects model. However, when coefficients are printed, the intercept is actually an estimate of the mean of the "last" level of the factor, and the other coefficients estimate the differences between the means of the other levels and the last level. Suppose we have 3 levels of factor A, 4 levels of factor B, and three cells with zero observations. Using the model $y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}$, SAS uses a generalized inverse for which $\widehat{\alpha}_3 = \widehat{\beta}_4 = 0$. Note also that with this generalized inverse (as with any other) γ_{14}, γ_{23}, and γ_{31} are non-estimable, and do not show up in the list of estimable functions.

- In R a numeric variable could be used as a continuous regression variable or as a factor (what SAS calls a class variable). If there are any non-numeric entries in a column of data, then the `read.table` function will convert the column to a factor. If the column is all numeric, you can force it to be treated as categorical by writing `factor(x)`. R does not use a generalized inverse of the $\boldsymbol{X}$ matrix, but instead reduces $\boldsymbol{X}$ to a full rank reparameterization. One may choose the way this is done with the `contrasts` statement in an `options()` call. The defaults are "treatment" contrasts for R and "Helmert" contrasts for Splus. For a four level factor the reparameterized unique rows of $\boldsymbol{X}$ (ignoring the intercept column) would be one of these sets:

```
contr.treatment    contr.helmert       contr.sum        contr.sas
     0  0  0        -1   -1   -1      1    0    0         1  0  0
     1  0  0         1   -1   -1      0    1    0         0  1  0
     0  1  0         0    2   -1      0    0    1         0  0  1
     0  0  1         0    0    3     -1   -1   -1         0  0  0
```

A column of ones for the intercept is included in the model by default, but if no intercept is requested (as in `y ~ treatment - 1`, or equivalently, `y ~ 0 + treatment`) then $\boldsymbol{X}$ would not have a column of ones, but would have I columns of indicator variables. The model matrix in either case has I columns and is of full column rank. One can look at the model matrix of a fitted object with the `model.matrix` command.

```
> wheat.fit <- lm(yield ~ fertilizer, data = wheat)
> model.matrix(wheat.fit)
  (too big to print here, but the unique rows look like one of those
   above with an initial column of ones.)
```

and one can view the full $\widetilde{\boldsymbol{\beta}}$, estimated complete parameter vector, with the command `dummy.coef(wheat.fit)`. For the "treatment" setting, the estimate of τ_1 will be 0, and $\widetilde{\mu} = \bar{y}_1$. The other coefficient estimates are of the form $\widetilde{\tau}_i = \bar{y}_i - \bar{y}_1$.

Helmert contrasts have the advantage of being orthogonal, but are more difficult to interpret, and are not generally recommended.

Bottom Line

Choosing a particular reparameterization $\mathbf{CR} = \boldsymbol{I} - \boldsymbol{K}^{\mathsf{T}}(\boldsymbol{K}\boldsymbol{K}^{\mathsf{T}})^{g}\boldsymbol{K}$ is equivalent to choosing a generalized inverse, and inferences about estimable functions are not dependent on the choice of generalized inverse. **All reparameterizations yield the same inferences on estimable contrasts**.

Any stat package makes some choices for you about how to handle a less-than-full-rank $\boldsymbol{X}$ matrix. For any package you use, you need to know how it handles a categorical predictor. The standard output shows t-tests which may not be of interest, but you need to be able to explain exactly what they are testing (and those must be estimable contrasts, since we do get estimates). For each line of the output, what null and alternative hypotheses are being tested? Specify in terms of the full treatment effects model.

Example of a one-way ANOVA where 4 levels of fertilizer (Q, R, S, or T) are applied to wheat. The response is yield.

```
> with(wheat,tapply(yield,fert, mean))
    Q     R     S     T
22.34 23.83 22.88 19.85
> wheat.fit0 <-  lm(yield ~ fert, wheat)  ## one-way model####
> coef(wheat.fit0)
(Intercept)    fertR    fertS    fertT
      22.34    1.493    0.543   -2.490
> dummy.coef(wheat.fit0)
Full coefficients are
(Intercept):         22.34
fert:                    Q          R          S          T
                 0.0000000  1.4933333  0.5433333 -2.4900000
> summary(wheat.fit0)$coef
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 22.3400000  0.6901462 32.3699537 4.382242e-18
fertR        1.4933333  0.9344623  1.5980669 1.265258e-01
fertS        0.5433333  0.9344623  0.5814395 5.677790e-01
fertT       -2.4900000  0.9344623 -2.6646338 1.531204e-02

> wheat.fit2 <-  lm(yield ~ fert*var, wheat)    ## 2-way model with interactions*********
> coef(wheat.fit2)
(Intercept)       fertR       fertS       fertT        varB        varC
      17.40        5.10        5.50        2.85        6.50        5.85
 fertR:varB  fertS:varB  fertT:varB  fertR:varC  fertS:varC  fertT:varC
      -4.00       -6.15       -6.65       -4.35       -6.25       -6.90
> dummy.coef(wheat.fit2)
Full coefficients are
(Intercept): 17.4
fert:           Q     R     S     T
             0.00  5.10  5.50  2.85
var:            A     B     C
             0.00  6.50  5.85
fert:var:     Q:A   R:A   S:A   T:A   Q:B   R:B   S:B   T:B   Q:C   R:C   S:C   T:C
             0.00  0.00  0.00  0.00  0.00 -4.00 -6.15 -6.65  0.00 -4.35  -6.25 -6.90

########################  Table of Means  ##################################
> with(wheat,tapply(yield,list(var,fert), mean))
      Q    R     S     T
A 17.40 22.5 22.90 20.25
B 23.90 25.0 23.25 20.10
C 23.25 24.0 22.50 19.20
```

Note relationships between cell means and coefficient estimates. Which estimates are constrained to be zero?

```
 *############################################################  SAS one-way;
               Estimates from SAS Proc GLM one-way model
                                          Standard
   Parameter           Estimate             Error    t Value    Pr > |t|

   Intercept        19.85000000 B      0.63001439      31.51      <.0001
   Fert      Q       2.49000000 B      0.93446235       2.66      0.0153
   Fert      R       3.98333333 B      0.89097489       4.47      0.0003
   Fert      S       3.03333333 B      0.89097489       3.40      0.0030
   Fert      T       0.00000000 B       .                .         .

NOTE: The X'X matrix has been found to be singular, and a generalized
  inverse was used to solve the normal equations.  Terms whose
  estimates are followed by the letter 'B' are not uniquely estimable.
 *############################################################  SAS two-way;
                                          Standard
  Parameter               Estimate             Error    t Value    Pr > |t|

  Intercept            19.20000000 B      0.52807541      36.36      <.0001
  Variety      A        1.05000000 B      0.74681140       1.41      0.1873
  Variety      B        0.90000000 B      0.74681140       1.21      0.2534
  Variety      C        0.00000000 B       .                .         .
  Fert         Q        4.05000000 B      0.74681140       5.42      0.0002
  Fert         R        4.80000000 B      0.74681140       6.43      <.0001
  Fert         S        3.30000000 B      0.74681140       4.42      0.0010
  Fert         T        0.00000000 B       .                .         .
  Variety*Fert A Q     -6.90000000 B      1.18081251      -5.84      0.0001
  Variety*Fert A R     -2.55000000 B      1.05615082      -2.41      0.0343
  Variety*Fert A S     -0.65000000 B      1.05615082      -0.62      0.5508
  Variety*Fert A T      0.00000000 B       .                .         .
  Variety*Fert B Q     -0.25000000 B      1.05615082      -0.24      0.8172
  Variety*Fert B R      0.10000000 B      1.05615082       0.09      0.9263
  Variety*Fert B S     -0.15000000 B      1.05615082      -0.14      0.8896
  Variety*Fert B T      0.00000000 B       .                .         .
  Variety*Fert C Q      0.00000000 B       .                .         .
  Variety*Fert C R      0.00000000 B       .                .         .
  Variety*Fert C S      0.00000000 B       .                .         .
  Variety*Fert C T      0.00000000 B       .                .         .
## same warning as above ##
```

Note relationships between cell means and coefficient estimates. Which estimates are constrained to be zero?

3 Breaking All the Rules

In Stat 217 and Stat 410 we say that inference requires us to assume

- variance of residuals is constant,
- observations are independent (uncorrelated), and
- residuals have a normal distribution.

This is summarized as: $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I})$ or $e_i \sim$ iid $N(0, \sigma^2),\ i = 1, \ldots, n$ (why are those equivalent statements?)

There are ways to handle violations of any of those assumptions, which we will now examine.

3.1 Heteroscedasticity, or Non-Constant Variance

Some authors, for instance Monahan (2008) Chapter 4, write the Gauss-Markov conditions under the condition $\boldsymbol{\epsilon} \sim (\mathbf{0}, \sigma^2 \boldsymbol{I})$. If instead, we wish to consider using $\boldsymbol{\epsilon} \sim (\mathbf{0}, \sigma^2 \boldsymbol{V})$, where $\boldsymbol{V}$ is a known matrix, then consider the transformed equation,

$$\boldsymbol{V}^{-1/2}\boldsymbol{y} = \boldsymbol{V}^{-1/2}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{V}^{-1/2}\boldsymbol{\epsilon} \tag{4}$$

where $\boldsymbol{V}^{1/2}\boldsymbol{V}^{1/2} = \boldsymbol{V}$ and $\boldsymbol{V}^{-1/2} = (\boldsymbol{V}^{1/2})^{-1}$. The transformed errors have mean $\mathbf{0}$ and variance $\sigma^2 \boldsymbol{I}$. The OLS solution to the transformed equation is

$$\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1/2}\boldsymbol{V}^{-1/2}\boldsymbol{X})^{-}\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1/2}\boldsymbol{V}^{-1/2}\boldsymbol{y} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-}\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{y},$$

which is BLUE by Monahan's version of Gauss-Markov, as well as by our version. The point of the derivation is: 1) equation (4) and $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ share the same solutions, 2) BLUE for one is BLUE for the other, and 3) although the Monahan's version of Gauss-Markov looks like it's too restrictive, it can handle the general $\text{var}(\boldsymbol{y}) = \boldsymbol{V}$ case via premultiplication by $\boldsymbol{V}^{-1/2}$. Either way a large assumption is needed, 1) that errors are iid, or 2) that $\boldsymbol{V}$ is known.

3.1.1 Weighting

We will now consider the simple case where $\boldsymbol{V}$ is a diagonal matrix (meaning variances can change across the observations, but observations are still uncorrelated).

Examples:

- Instead of individual responses, our y_i's are each a mean of n_i data points with n_i not all the same. We know that the variance of a mean is σ^2/n_i, so variances differ.
- If $\boldsymbol{y}$ consists of Poisson-distributed counts, then $\text{var}(y_i) = E(y_i) = \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}$.
- If $y_i = \frac{M_i}{n_i}$ where $M_i \sim Bin(n_i, p_i)$, then $E(y_i) = n_i \times p_i/n_i = p_i$ and $\text{var}(y_i) = p_i(1-p_i)/n_i$, again a function of the mean.

When $\boldsymbol{V}$ is diagonal the ϵ_i's are uncorrelated and have variance $\text{var}(y_i) = \sigma_i^2$. The diagonal elements of $\sigma^2\boldsymbol{V}$ are σ_i^2, and the inverse matrix is diagonal, $\boldsymbol{V}^{-1} = \text{diag}(\sigma_i^{-2})$. From the Gauss-Markov theorem, we know that GLS gives BLUEs, so we should minimize $(\boldsymbol{y} - \widehat{\boldsymbol{y}})^\mathsf{T}\boldsymbol{V}^{-1}(\boldsymbol{y} - \widehat{\boldsymbol{y}})$, rather than minimizing $(\boldsymbol{y} - \widehat{\boldsymbol{y}})^\mathsf{T}(\boldsymbol{y} - \widehat{\boldsymbol{y}})$. With diagonal $\boldsymbol{V}$, the quadratic form simplifies nicely to

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\mathsf{T}[\text{diag}\{\sigma_i^{-2}\}](\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{1}{\sigma_i^2}(y_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})^2.$$

This is a weighted sum of squares with weights $w_i = 1/\sigma_i^2$.

Notice: Larger variance goes with _____ weight and smaller variance goes with _____ weight.

To use known weights, there are weight arguments in SAS and S.

```
Proc Reg; ** or Proc GLM ;
  weight w;
  model y = x;
run;
```

```
In R:
    lm(y ~ x, weights = w )
```

For the above examples, we assumed particular distributions, and could then use weights based on these assumptions. When each observation is a mean of n_i independent observations, $w_i = n_i$. For the Poisson case, weights would be $1/\hat{y}_i$. For the proportion, $w_i = n_i/(\hat{p}_i(1 - \hat{p}_i))$.

With real data rather than artificial cases, how do we know that we have a problem with non-constant variance? You should be familiar with residual diagnostics where one looks for a fan-shape in residual versus fitted plot, or a trend in a spread-location plot to indicate a problem with the assumption of constant variance.

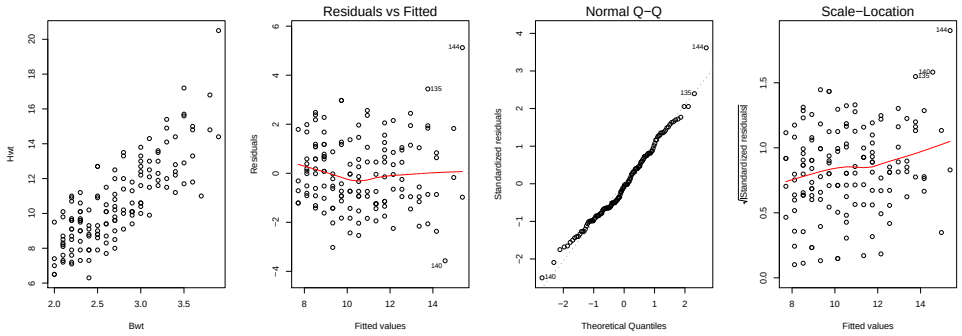
To obtain a fair comparison of residual spread, one needs to standardize each residual, dividing by its standard error. Estimated residuals do not all have the same variance, as those further from the center of the data will vary more than those close to $\hat{\boldsymbol{x}}$. Mathematically, $\mathbf{e} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y} \sim [0, \sigma^2\boldsymbol{V}(\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^g\boldsymbol{X}^\mathsf{T})]$. When $\boldsymbol{V} = \boldsymbol{I}$, $\text{Var}(\mathbf{e}) = \sigma^2(\boldsymbol{I} - \boldsymbol{H})$ where $\boldsymbol{H}$ is the hat matrix, $\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^g\boldsymbol{X}^\mathsf{T}$. The variance of the ith residual is then $\sigma^2(1 - h_{ii})$ where h_{ii} is the ith diagonal of the hat matrix. Standardized residuals are $e_i^* = \frac{e_i}{s\sqrt{1-h_{ii}}}$, and studentized residuals are similar, but replace s with $s_{(i)}$ which is computed while omitting the ith residual. In R we can plot standardized residuals versus fitted values (Scale-location plot) with the command:

```
plot(fitted.lm, which=3) ## 1 for resid vs fits, 2 for qqnorm, 3 for scale-location
```

Problems with the equal-variance assumption will show up as a general trend in the $\sqrt{|e_i^*|}$, increasing or decreasing. Some (Sen and Srivastava 1997) suggest the use of White's test for non-constant variance, but it is highly affected by outliers and non-normality. I prefer to use the residual plots.

Example

Venables and Ripley (2002) use data on cat's body weight to estimate the weight of the heart via linear regression. Question: Is there some relationship between mean and spread?




```
> par(mfrow=c(1,4))
> data(cats, package="MASS")
> plot(Hwt ~ Bwt, cats)
> cat.fit1 <-  lm(Hwt ~ Bwt, cats)
> plot(cat.fit1, which=1:3)
```

The scale-location plot of $\sqrt{|e_i^*|}$ versus $\widehat{y}_i$ shows some increase in spread from left to right. Instead of transforming, we will estimate the relationship between mean and variance and use weighted regression. Assume that variance is proportional to the 2α power of the mean.

$$\sigma_i^2 \propto \mu_i^{2\alpha} \text{ or } \sigma_i \propto \mu_i^{\alpha}$$

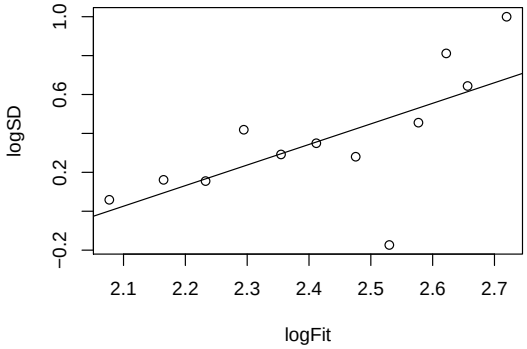
Taking logs (assuming mean responses are all positive), we have

$$\ln(\sigma) = k + \alpha \ln(\mu).$$

If we could find proxies for σ and μ, we could estimate α using regression. Possible approaches:

- Group the data into "bins" according to the fitted values. Use the mean (or median) fitted value as $\widehat{\mu}_i$ and the standard deviation of the residuals in that bin as $\widehat{\sigma}_i$.

```
> which.bins <-  cut(fitted(cat.fit1),12)
> logSD = log(tapply(resid(cat.fit1), which.bins, sd))
> logFit= log( tapply(fitted(cat.fit1), which.bins, mean))
> plot(logSD ~ logFit);    abline( lm(logSD ~ logFit))
> summary(lm(logSD ~ logFit))$coef
             Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -2.195258  0.9138305 -2.402259 0.03717186
logFit       1.057705  0.3754001  2.817541 0.01823747
```



We see that spread does tend to increase with fitted value, and obtain an estimated α of about 1. We could refit the original `lm` using weights to obtain better fitted values and residuals, iterating til convergence, but will demonstrate iteration in the next option.

- Use $\widehat{y}_i$ to represent μ_i and $|e_i|$ as a proxy for σ_i.

```
> logabsResid <-  log(abs(resid(cat.fit1)))
> logFits <-  log(fitted(cat.fit1))
> plot(logabsResid ~ logFits)
> abline(lm(logabsResid ~ logFits))
> summary(lm(logabsResid ~ logFits))$coef
             Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -2.975371  1.1598032 -2.565410 0.01134255
logFits      1.178486  0.4926489  2.392141 0.01805925
```

Again the estimate for α is just over 1. The estimate of α is based on a fitted OLS model which is not optimal. An improved model uses weights proportional to inverse variance (WLS).

```
> coef( cat.fit1)
(Intercept)         Bwt
 -0.3566624   4.0340627
> cat.fit2 <-  lm(Hwt ~ Bwt, cats, weights = fitted(cat.fit1)^(-2))
>  coef(cat.fit2)
(Intercept)          Bwt
    0.00934      3.89773
> logabsResid <-  log(abs(resid(cat.fit2)))
>  logFits <-  log(fitted(cat.fit2))
>  summary(lm(logabsResid ~ logFits))$coef
             Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -2.610505  1.3502278 -1.933381 0.05517892
logFits      1.002785  0.5735058  1.748518 0.08253549
> cat.fit2 <-  lm(Hwt ~ Bwt, cats, weights = fitted(cat.fit2)^(-2))
> summary(cat.fit2)$coef
              Estimate Std. Error     t value     Pr(>|t|)
(Intercept) 0.01013253  0.6781681  0.01494104 9.881002e-01
Bwt         3.89742610  0.2604154 14.96619021 5.447339e-31
```

The above model gives coefficient estimates which are based on weights which seem consistent with the model. We often have to iterate between estimating coefficients (based on weights) and estimating weights (for given coefficients) until the coefficient estimates stop changing.

An Easier Way

Pinheiro and Bates (2000) have developed functions which do all the work for us, and even provide a way to test to see if the power variance model is needed. Their `gls` function does the whole job:

```
> library(nlme)  # nonlinear mixed effects includes linear mixed effects and GLS
> cat.glsfit <- gls((Hwt ~ Bwt, cats, weights = varPower())
> summary(cat.glsfit)
Variance function:
 Structure: Power of variance covariate
 Formula: ~fitted(.)
```

```
 Parameter estimates:
    power
0.8617148

Coefficients:
               Value Std.Error   t-value p-value
(Intercept) -0.03454 0.6771919 -0.051004  0.9594
Bwt          3.91453 0.2580108 15.171962  0.0000

Residual standard error: 0.1870558

> anova(cat.glsfit, cat.fit1)
           Model df      AIC      BIC    logLik   Test  L.Ratio p-value
cat.glsfit     1  4 517.9115 529.7348 -254.9557
cat.fit1       2  3 523.4538 532.3213 -258.7269 1 vs 2 7.542319   0.006
```

The above call to `anova` must have the gls-fit model as the first argument because `anova` is a "generic" function. It looks for a particular flavor of itself to work with the class of object represented in the first argument. If the first argument was created by `lm`, then its class is "lm", and `anova.lm` will be used. The `anova.lm` function doesn't have a way to handle a gls-fit model. If the first argument comes from `gls`, then its class is "gls" so `anova.gls` will be used. That flavor of `anova` can handle the simpler "lm" object as well as the "gls" object.

Interpretation: the test above is for $H_0: \ \alpha = 0$ (variance is constant relative to the mean) versus $H_a: \ \alpha \neq 0$ in the model $\sigma_i \propto \mu_i^{\alpha}$. With a small p-value we reject the null and conclude that variance is proportional to a non-zero power of the mean.

3.1.2 Transformation of Response

The other approach to handling heteroschedastic data is to transform the response. Suppose

$$\sigma_i^2 \propto g(\mu_i) \text{ where } \mu_i = E(y|\boldsymbol{x}_i)$$

then a transformation may give constant variance to the y_i's.

Example: $y_i|x_i \sim \text{Poisson}(\mu_i = \beta_0 + \beta_1 x_i)$
$E(y_i|x_i) = \mu_i = \text{var}(y_i|x_i)$ Look at a particular transformation, $y^{1/2}$.
Delta Method:
Variance of a function of y is approximately the variance of y times the square of the first derivative evaluated at the mean: $\text{var}[f(y)] \approx \sigma_y^2[f'(\mu)]^2$.

Reminder: the Delta Method is an outgrowth of the Taylor series expansion about the mean.

$$f(y) = f(\mu) + (y-\mu)f'(\mu) + (y-\mu)^2 f''(\mu^*)/2 \text{ for some } \mu^* \in (y,\mu) \text{ or } \mu^* \in (\mu, y)$$

Subtract $f(\mu)$ and take expectation:

$$E(f(y) - f(\mu)) = 0 \times f'(\mu) + \sigma_y^2 f''(\mu^*)/2$$

and if we drop the terms beyond term with the second derivative and take variance,

$$\text{Var}[f(y)] \approx \text{Var}[f(\mu)] + \text{Var}[(y-\mu)f'(\mu)] = \sigma_y^2[f'(\mu)]^2$$

Apply the Delta Method to the Poisson with $f(y) = \sqrt{y}$.

$$\text{var}(y_i^{1/2}|x_i) \approx \mu_i \left[\left. \frac{dy^{1/2}}{dy} \right|_{\mu_i} \right]^2 = \mu_i \left[\frac{1}{2} \mu_i^{-1/2} \right]^2 = \frac{1}{4} \text{ which does not depend on } \mu_i$$

So this particular transformation stabilized the variance because $\text{Var}(y)^{-1}$ was a multiple of $\left[\frac{dy^{1/2}}{dy}|_{\mu_i} \right]^2$ and the dependence on μ disappeared.

Choosing a transformation

Reverse the process. If we know the functional form for variance in terms of the mean, say $\sigma^2 = g(\mu)$, then which transformation $h(y)$ will have constant variance?

Take $h'(\mu) = [g(\mu)]^{-1/2}$ or $h(\mu) = \int [g(\mu)]^{-1/2} d\mu$ (the anti-derivative).

For the cat weight data we estimated the variance as a function of μ.

$$\text{var}(y) \propto \mu^1 = g(\mu) \Longrightarrow g(\mu)^{-.5} = \mu^{-.5} = h'(\mu) \Longrightarrow h(\mu) = \mu^{+.5}$$

So we should transform y using a square root transformation, $y_i^* = y_i^{1/2}$ (It's only coincidence that the same is true for Poisson distributed data.).

Cautions

Transformations are used for three distinct purposes:

1. To stabilize variance,
2. To make a non-linear model linear, and
3. To make residuals more normal.

One transformation cannot necessarily achieve all three goals. Poisson data are typically modeled with multiplicative terms representing rates, so $log(y)$ provides a linear model. To obtain residuals closest to normal, $y^{2/3}$ is used, while $y^{1/2}$ is best for stabilizing variance.

Box-Cox Transformations (Box, G. E. P. and Cox, D. R. 1964)

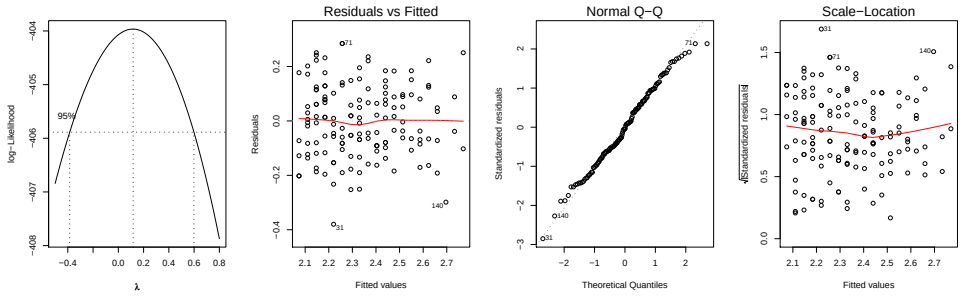
Transform y_i to

$$y^* = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0,\ y > 0 \\ \log(y) & \text{if } \lambda = 0,\ y > 0 \end{cases}$$

For a given value of λ, we can compute the likelihood of y^*, the data, under normality. The MLE of λ is the value which maximizes the likelihood under the assumption of normality.

Typically finding the exact "best" λ is not essential, and one should also keep in mind the audience which needs to understand the analysis. We will examine the range of values for λ in the 95% CI and choose a "nice" value like 0, .5, -.5, or 1 when possible. Use `library(MASS)` and the `boxcox(fittedmodel)` commands. If a plotting window is open, it shows the results graphically.

```
> library(MASS)
> boxcox( cat.fit1, lambda = seq(-.5,.6,0.1) ) ## Note: 0 is close to optimal
> cat.Tfit <- lm(log(Hwt) ~ Bwt, cats)         ## log transform
> plot(cat.Tfit, which=1:3)
```



The "subtract one and divide by λ" steps in the Box-Cox formula are used to make the function continuous in λ at zero, but are not essential; it's enough to use y^λ instead. I strongly prefer "nice" values for λ, and like to use square root, log, reciprocal, and a few others. Some stat packages spit out "the best" power estimate, but there is no reason to use a power like 0.3749. It's too hard to interpret. Do think about your audience. Microbiologists, for example, are very familiar with log transformations, so I might bend a bit to use $\lambda = 0$. People who are not mathematically experienced might be averse to transformation. In that case, weighting could be easier to explain.

3.2 Correlated Errors

Up to this point, we have assumed that $\text{Var}(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{I}$ or $\sigma^2\boldsymbol{V}$ where $\boldsymbol{V}$ was a diagonal matrix. We will now consider several types of correlations which can occur, and look at how they affect estimation and testing in linear models.

- Correlation due to different levels of randomization.
- Correlation for repeated measurements on the same units, including
  - Constant correlation models, and
  - Time series models
- Spatial models

We will start with the simplest case, that of repeated measures on an individual or unit.

3.2.1 Repeated Measures in Matrix Notation

Suppose that individual i is measured not just once, but n_i times under different treatments, and we collect those measurements into $\boldsymbol{y}_i$. Now model $\boldsymbol{y}_i$ as:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + b_i\mathbf{1} + \boldsymbol{\epsilon}_i, \text{ a vector of length } n_i$$

where $\boldsymbol{X}_i\boldsymbol{\beta}$ describes the treatment structure for individual i. The last two terms are both random terms; the bottom level $\boldsymbol{\epsilon}_i$ can be assumed to have a $(\mathbf{0}, \sigma^2\boldsymbol{I})$ structure, independent of a random effect from the individual, $b_i \sim (0, \sigma_b^2)$. Think of b_i as an adjustment to the overall intercept or mean for the effect of individual i. To use what we know about GLS estimation, combine the two error terms into a single one: $b_i\mathbf{1} + \boldsymbol{\epsilon}_i \sim (0, \boldsymbol{\Sigma}_i)$ where

$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \sigma^2+\sigma_b^2 & \sigma_b^2 & \cdots & \sigma_b^2 \\ \sigma_b^2 & \sigma^2+\sigma_b^2 & \cdots & \sigma_b^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_b^2 & \sigma_b^2 & \cdots & \sigma^2+\sigma_b^2 \end{bmatrix}$$

This correlation structure is called "compound symmetric". Alternative notation: diagonal elements could be labeled $\sigma_t^2 = \sigma^2 + \sigma_b^2$ and off-diagonals could be labeled $\rho\sigma_t^2$, and σ_t^2 could be factored out in front. The important point is that the matrix has only two values: one for all diagonal elements, and another for off-diagonals.

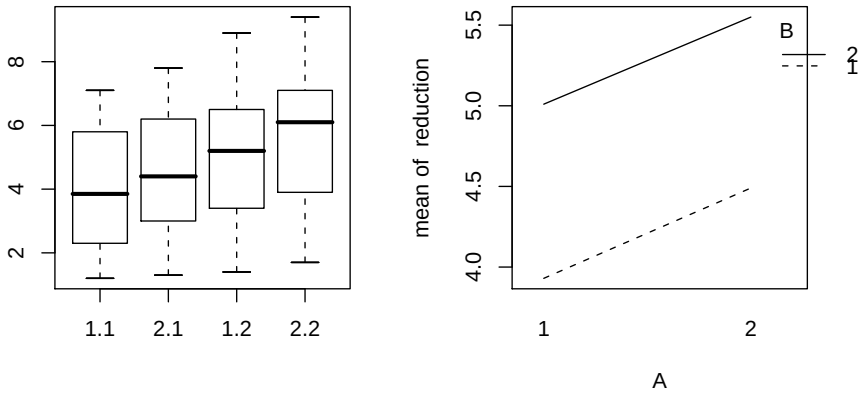
If $\boldsymbol{X}\boldsymbol{\beta}$ is a one-way ANOVA model, and we combine data from all individuals by stringing together each vector $\boldsymbol{y}_i$ in turn, we have:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \\ \vdots \\ \boldsymbol{y}_I \end{bmatrix} = \boldsymbol{X}\begin{bmatrix} \mu \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} b_1\mathbf{1}+\boldsymbol{\epsilon}_1 \\ b_2\mathbf{1}+\boldsymbol{\epsilon}_2 \\ b_3\mathbf{1}+\boldsymbol{\epsilon}_3 \\ \vdots \\ b_I\mathbf{1}+\boldsymbol{\epsilon}_I \end{bmatrix}; \quad \text{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Sigma}_I \end{bmatrix}$$

The variance-covariance of $\boldsymbol{y}$ is block diagonal with $\boldsymbol{\Sigma}_i$ differing only in size, not structure, and depends on (σ^2, σ_b^2) or on (ρ, σ_t^2). However, $\boldsymbol{V}$ is not known and the Gauss-Markov theorem does not directly apply.

We can estimate the variance σ_b^2 or the correlation, ρ and work with $\widehat{\boldsymbol{\Sigma}}$ to improve on OLS through use of $\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}^\mathsf{T}\widehat{\boldsymbol{V}}^{-1}\boldsymbol{X})^g\boldsymbol{X}^\mathsf{T}\widehat{\boldsymbol{V}}^{-1}\boldsymbol{y}$. Although optimal BLUEs are not guaranteed, it seems that Estimated GLS (EGLS) should improve on plain OLS estimation. To motivate the EGLS approach, one can think in terms of maximizing likelihood, or take a Bayesian approach and maximize the posterior.

The `gls` function in R and PROC MIXED in SAS provide EGLS options. We will look at repeated measures data from Kutner et al. (2004) Ex. 27.18 on reduction in intensity of migraine headache pain (higher scores are better) under application of two medications: A at low (1) or high (2) dosage, and B at low (1) or high (2) dosage. This is a crossover study where each subject was given each treatment at intervals far enough apart in time that no carryover effect was expected.




```
> boxplot( reduction ~ interaction(A,B), data = migraine)
> with(migraine, interaction.plot(A,B,reduction))
                           ###  A poor analysis ignoring correlations ####
> anova(migraine.lm <- lm(reduction ~ A*B, migraine))
Response: reduction
          Df  Sum Sq Mean Sq F value Pr(>F)
A          1   3.025   3.025  0.6729 0.4175
B          1  11.449  11.449  2.5467 0.1193
A:B        1   0.001   0.001  0.0002 0.9882
Residuals 36 161.844   4.496
> summary(migraine.lm)$coef
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.9300     0.6705   5.861 1.06e-06
A2            0.5600     0.9482   0.591    0.558
B2            1.0800     0.9482   1.139    0.262
A2:B2        -0.0200     1.3410  -0.015    0.988

  ## EGLS analysis ####
> anova(migraine.gls <- gls(reduction ~ A*B, migraine, correlation = corCompSymm( form = ~1|ID)))
Denom. DF: 36
            numDF  F-value p-value
(Intercept)     1 52.43538  <.0001
A               1 11.24226  0.0019
B               1 42.54962  <.0001
A:B             1  0.00372  0.9517
```

```
> summary(migraine.gls)
Correlation Structure: Compound symmetry
 Formula: ~1 | ID
 Parameter estimate(s):
      Rho
0.9401481

Coefficients:
            Value Std.Error   t-value p-value
(Intercept)  3.93 0.6704973  5.861321  0.0000
A2           0.56 0.2319802  2.413999  0.0210
B2           1.08 0.2319802  4.655570  0.0000
A2:B2       -0.02 0.3280696 -0.060963  0.9517
```

Note: There is a very strong correlation of 0.94 within an individual. Accounting for that reduces standard errors by a factor of 1/3. Do conclusions about significant effects change?

In SAS:

```
options ps=76 ls=76;
data migraine;
    infile "../data/migraine"  firstobs=2;
    input reduction ID  A  B ;

proc anova data= migraine;
  class  ID A B;
  model reduction = A B A*B;
run;
proc mixed data = migraine;
  class  ID A B;
  model reduction = A B A*B;
  repeated/subject=id type = CS;
run;
```

```
  ****************** PROC  ANOVA  Output (not optimal)  ********************;
 Source                     DF      Anova SS   Mean Square  F Value  Pr > F

 A                           1    3.02500000    3.02500000     0.67  0.4175
 B                           1   11.44900000   11.44900000     2.55  0.1193
 A*B                         1    0.00100000    0.00100000     0.00  0.9882

 ******************* PROC MIXED  EGLS  ************************************;
                      Covariance Parameter Estimates

                     Cov Parm     Subject    Estimate
                     CS           ID           4.2266
                     Residual                  0.2691

                        Solution for Fixed Effects
                                    Standard
 Effect       A    B    Estimate       Error      DF    t Value    Pr > |t|
 Intercept                 5.5500      0.6705       9       8.28      <.0001
 A            1           -0.5400      0.2320       9      -2.33      0.0449
 B                 1      -1.0600      0.2320       9      -4.57      0.0013
 A*B          1    1     -0.02000      0.3281       9      -0.06      0.9527

                      Type 3 Tests of Fixed Effects
```

Effect	Num DF	Den DF	F Value	Pr > F
A	1	9	11.24	0.0085
B	1	9	42.55	0.0001
A*B	1	9	0.00	0.9527

SAS reports two variances whereas R reports a correlation and a standard deviation. The two packages do agree $4.2266/(4.2266 + 0.2691) = 0.9401$.

Aside: Another way to analyze these data correctly would be to think of each individual as a block. We want to first remove the effect of block-to-block variation and then evaluate the treatments of interest. Fitting the model
`migraine2.lm <- lm(reduction ~ factor(ID) +A*B, migraine)` and running `anova` on that gives the same tests for A, B, and `A*B` as did `gls` above. The explanation above focuses on correlation to illustrate the advantage of EGLS, but the RBD analysis is equivalent.

3.2.2 Multiple Levels of Randomization in ANOVA

The F test statistic used in ANOVA is a ratio of two variance estimates: one based on within group differences, and the coming from differences between groups. The randomization process is key to identifying units which were treated alike. When randomization is done in stages, error terms are created at each stage.

Brief Anova Review

In a completely randomized design (CRD), each unit has the same probability of assignment to any given treatment group. In a one-way analysis where factor A has a levels, the degrees of freedom partition as:

CRD Source	df	SSq
Treatments	$a-1$	SST
Error	$n-a$	SSE
Total	$n-1$	

In a randomized block design (RBD) with b blocks each containing a units, one randomly assigns treatments with in each block giving:

RBD Source	df	SSq
Blocks	$b-1$	SSB
Treatments	$a-1$	SST
Error	$(a-1)(b-1)$	SSE
Total	$ab-1$	

In both cases, we use MSE = SSE/dfError to test for treatment effect. The RBD is a special case of two-way anova where we don't care to test for differences between blocks (If they have significant differences, then blocking was the right approach, but if not, we don't throw out block effects.) and the error term is also the interaction term because each treatment occurs only once in each block. One must assume that block by treatment interactions are negligible.

Example: a Randomized Block Design with 5 treatments (Control, A, B, C, D) and 5 blocks. Each treatment used once per block. (Snedecor and Cochran (1980), §14.2)

Source	df	SSq	Mean Sq	F value	Pr(>F)
Blocks	4	49.84	12.46	2.3031	0.10320
Treatments	4	83.84	20.96	3.8743	0.02189
Error	___	86.56	5.41		
Total	24				

At the $\alpha = .05$ significance level, what do you conclude?

Multiple levels example:

We wish to compare 4 levels of fertilizer and 3 nematode treatments on yield of turnips.

Step 1. There are 12 plots of ground available which will each be randomly assigned one of the 4 fertilizer levels. A possible randomization is:

F_3	F_2	F_1	F_3	F_2	F_4
F_1	F_4	F_2	F_3	F_4	F_1

The design at this stage is a CRD and the ANOVA table setup is:

Source	df	SSq
Fertilizer	3	
Error	8	
Total	11	

Step 2. Each plot is divided into three sub or split plots, and the three nematode treatments are randomly assigned within each plot. For example the first two plots might get:

F_3			F_2		
N_3	N_1	N_2	N_1	N_3	N_2

At the subplot level, we have a RBD for nematode treatments with 12 blocks and $12 \times 3 = 36$ units total. The ANOVA table setup is:

Source	df	SSq
Block	11	
Nematode	2	
Error	22	
Total	35	

Combine both tables into one table with two levels:

Level	Source	df
First	Plot level total df:	11
	[Fertilizer]	[3]
	[Fert w/in Plot]	[8]
Second:		
	Nematode	2
	Plot by Nematode	22
	[Nem by Fert]	[6]
	[Nem by Fert w/in Plot]	[16]
Total		35

Re-Arranged Table:

Level	Source	df
Whole Plot:		
	Fertilizer	3
	Whole Plot Error	8
Split Plot:		
	Nematode	2
	Nem by Fert	6
	Nem by Plot/Fert	16
Total		35

Use whole plot error term to test the Fertilizer effects, `Nematode by Fert w/in Plots` for effects at the split plot level. Confirm that we are comparing means based on variation of units treated alike.
Model:

$$y_{ijk} = \mu + F_i + b_{ij} + N_k + (FN)_{ik} + \epsilon_{ijk}$$

F_i Fixed fertilizer effect, $i = 1, 2, 3, 4$

b_{ij} Random Whole Plot effect $\sim N(0, \sigma_b^2)$

N_k effect of k^{th} nematode treatment, $k = 1, 2, 3$.

$(FN)_{ik}$ interactions

ϵ_{ijk} iid $N(0, \sigma^2)$ indep of b_{ij}'s

Variances of the y_{ijk} have two independent components, σ_b^2 from the random plot assignment, and σ^2 from the random sub-plot. By independence, $\text{Var}(y_{ijk}) = \sigma_b^2 + \sigma^2$. Covariances are 0 between observations in different plots, but for observations from the same plot, covariance is σ_b^2 due to the shared random term, b_{ij} and correlation is $\sigma_b^2/(\sigma_b^2 + \sigma^2)$

SAS Code (without random effects):

```
proc anova;
  class plot F N;
  model y = plot F plot*F N F*N;  * uses wrong error on test of whole  plot trt;
  test h=F e = plot*F;  * meaning test hypothesis for F effect using plot*F as error term;
run;
```

The model statement prints the entire anova table for the split plot with the **wrong** error terms for the whole plot treatments. Then the requested test comes at the end. You have to rearrange the output to get a table like that above.

R code

```
split.fit <- aov(y ~ N*F+Error(plot/F), data )
```

You'll get the right tests with `summary(split.fit)`. The `model.tables` function returns standard errors of differences in estimated means at the plot and at the split plot levels. It's hard to compare across different levels of randomization, for instance, if you wanted standard errors of the interaction terms across levels of the aov table.

Why are split plots used? They give **greater precision** to the split-plot factor, which is an advantage in some situations. In agriculture some treatments are best applied to larger tracts of ground (e.g. irrigation methods). In industrial situations, there might be similar constraints on the available units. The same analysis is used with some repeated measures on individuals, and we might refer to it as a splitplot design, even when there are no "plots" in sight, just repeated measures on individuals. For example, in clinical trials, cross-over designs like the one in §3.2.1. That example could easily be turned into a split-plot analysis by using a factor like gender for the 10 individuals in the study. Gender is measured only once and cannot change during the assignment of the crossover treatments, so it is a "whole-plot" factor. That would add another column to our $\boldsymbol{X}$ matrix, but would not change the correlation structure. Split-plot designs have this same compound symmetric structure because the whole plot is like the individual measured multiple times.

Example from Venables and Ripley (2002)
Six blocks are available with 3 whole plots in each. One of three varieties of oats is planted in each plot, using a RBD (fresh randomization for each block). Each plot is then split into 4 subplots and randomly assigned a level of nitrogen (0, .2, .4, or .6 cwt of manure). Notes:

- This analysis is dependent on the *balance* of treatments within blocks. For the design shown, blocks, N, and V are orthogonal, with V effects estimable at the whole plot stratum and N effects and interactions estimable at the splitplot stratum. In other designs, such as balanced incomplete block with random blocks, treatment effects will cross the strata and we have to be more careful.

- The error term says to use two errors, B for blocks and B:V, the block by variety interaction, in addition to the bottom level, N within B:V. We can assign the total 72 degrees of freedom to the strata as:

  - 1 to the total of all observations
  - 5 to the blocks
  - $12 = 6 \times 2$ to varieties in the same block
  - $54 = 18 \times 3$ to fertilizers in the same whole plot

Analysis of split plot in R:

```
> oats.aov <- aov(Y ~ N * V + Error(B/V), data = oats)
> summary(oats.aov)
Error: B
          Df  Sum Sq Mean Sq F value Pr(>F)
Residuals  5 15875.3  3175.1

Error: B:V
          Df Sum Sq Mean Sq F value Pr(>F)
V          2 1786.4   893.2  1.4853 0.2724
Residuals 10 6013.3   601.3

Error: Within
          Df  Sum Sq Mean Sq F value    Pr(>F)
N          3 20020.5  6673.5 37.6856 2.458e-12
N:V        6   321.7    53.6  0.3028    0.9322
Residuals 45  7968.7   177.1
```

In SAS the commands and output are:

```
options ps=76 ls=76;
data oats;
 infile "classes/stat506/data/oats";
 input Block$ Variety$ Nitrogen$ Yield;
proc anova data= oats;
  class  Variety Nitrogen Block;
  model Yield = Block Variety Block*Variety Nitrogen Nitrogen*Variety ;
  test h=Variety e=Block*Variety;
run;
```

```
** ###################  Output  ********************************;

                          The ANOVA Procedure
Dependent Variable: Yield
                                    Sum of
Source                     DF       Squares   Mean Square  F Value  Pr > F
Model                      26   44017.19444    1692.96902     9.56  <.0001
Error                      45    7968.75000     177.08333
Corrected Total            71   51985.94444

            R-Square     Coef Var      Root MSE    Yield Mean
            0.846713     12.79887      13.30727      103.9722

Source                     DF      Anova SS   Mean Square  F Value  Pr > F
Block                       5   15875.27778    3175.05556    17.93  <.0001
Variety                     2    1786.36111     893.18056     5.04  0.0106
Variety*Block              10    6013.30556     601.33056     3.40  0.0023
Nitrogen                    3   20020.50000    6673.50000    37.69  <.0001
Variety*Nitrogen            6     321.75000      53.62500     0.30  0.9322

Tests of Hypotheses Using the Anova MS for Variety*Block as an Error Term

Source                     DF      Anova SS   Mean Square  F Value  Pr > F
Variety                     2   1786.361111    893.180556     1.49  0.2724


 ***************************  Reconstructed Table  ********************
Source                     DF      Anova SS   Mean Square  F Value  Pr > F
Block                       5   15875.27778    3175.05556   No Test
Variety                     2    1786.36111     893.18056     1.49  0.2724
Variety*Block              10    6013.30556     601.33056       Error Term A
Nitrogen                    3   20020.50000    6673.50000    37.69  <.0001
Variety*Nitrogen            6     321.75000      53.62500     0.30  0.9322
Error                      45    7968.75000     177.08333   From Above ANOVA
```

I am omitting the test for blocks because it is not of interest. If it has a small p-value, then those planning the experiment were wise to block as blocking increased the precision of the analysis. However, if block effect had a large p-value, I would still leave that term in the model because experts expected to have to take block effects into consideration.

Another approach is to specify the whole plots as a random effect in a mixed model. We'll revisit the split plot later as an example of a mixed effects model.

3.2.3 Growth Curve Models

In the repeated measures and in split plots variance-covariance matrix had a simple block diagonal structure:

$$\text{Var}(\boldsymbol{y}) = \begin{bmatrix} \boldsymbol{\Sigma} & 0 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Sigma} & 0 & \cdots & 0 \\ 0 & 0 & \boldsymbol{\Sigma} & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & \boldsymbol{\Sigma} \end{bmatrix} \qquad \text{where } \Sigma = \begin{bmatrix} \sigma^2+\sigma_b^2 & \sigma_b^2 & \cdots & \sigma_b^2 \\ \sigma_b^2 & \sigma^2+\sigma_b^2 & \cdots & \sigma_b^2 \\ \vdots & & \ddots & \\ \sigma_b^2 & \sigma_b^2 & \cdots & \sigma^2+\sigma_b^2 \end{bmatrix}$$

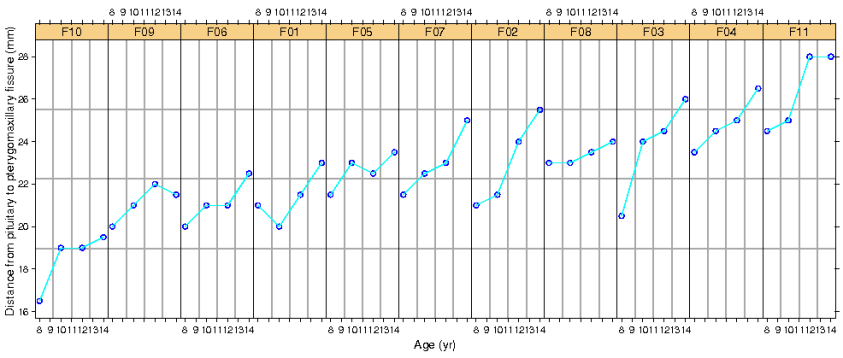
Next we'll look at a growth curve example where individuals are measured repeatedly. This also gives rise to a block diagonal structure, but the $\mathbf{\Sigma}$ is more general. The `Orthodont` data in the R package `nlme` (Pinheiro and Bates 2000) includes data on 11 girls whose teeth are measured every two years from age 8 to age 14. The objective is to model the "population average" growth curve for the population of girls of these ages. The adoption of a general $\mathbf{\Sigma}$ means that each age has it's own variance and that every two ages have a separate covariance (or correlation). In other words,

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{12} & \sigma_2^2 & \sigma_{23} & \sigma_{24} \\ \sigma_{13} & \sigma_{23} & \sigma_3^2 & \sigma_{34} \\ \sigma_{14} & \sigma_{24} & \sigma_{34} & \sigma_4^2 \end{bmatrix} \text{ assuming common variance: } \mathbf{\Sigma} = \sigma^2 \begin{bmatrix} 1 & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{12} & 1 & \rho_{23} & \rho_{24} \\ \rho_{13} & \rho_{23} & 1 & \rho_{34} \\ \rho_{14} & \rho_{24} & \rho_{34} & 1 \end{bmatrix}$$

There are 4 variances and 6 covariances to estimate in the general $\mathbf{\Sigma}$, that is: $\frac{n(n+1)}{2} = 10$ terms to estimate. Assuming common variance brings the number down to $1 + \frac{n(n-1)}{2}$, or 7 in this case. One should consider the number of parameters to estimate in light of the number of data points available. The "OrthoFem" data has $11 \times 4 = 44$ points, so it is possible to estimate 7 or 10 variance/covariance parameters.

We'll start by looking at the data with a lattice plot putting each girl in her own panel.

```
> data(Orthodont)
> plot(Orthodont, subset=Sex=="Female", col=4, type="b")
```



Notice that there is a general linear trend for individuals, but some of the lines top out or show dips. For an initial model, we will estimate an overall slope and intercept based on Estimated GLS using a general 4×4 $\mathbf{\Sigma}$ matrix.

```
> OrthoFem.gls <- gls(distance ~ age, data = Orthodont, subset = Sex=="Female",
                      correlation = corSymm(form=~1|Subject))
  ## Simple model with one variance and 6 covariances (correlations)
>  getVarCov( OrthoFem.gls)
Marginal variance covariance matrix
       [,1]   [,2]   [,3]   [,4]
[1,] 4.7311 4.0205 4.0751 3.9107
[2,] 4.0205 4.7311 4.1887 4.0468
[3,] 4.0751 4.1887 4.7311 4.4559
[4,] 3.9107 4.0468 4.4559 4.7311
  Standard Deviations: 2.1751 2.1751 2.1751 2.1751
```

```
                        ##################  Step 2  ######################
> OrthoFem.gls2 <- update( OrthoFem.gls, weights = varIdent(form=~1|factor(age)))
                              ## adds 3 more variance terms, one for each age.
> intervals(OrthoFem.gls2)
Approximate 95% confidence intervals

 Coefficients:
                lower       est.      upper
(Intercept) 15.955159 17.4219848 18.8888101
age          0.352274  0.4823211  0.6123683

 Correlation structure:
            lower      est.     upper
cor(1,2) 0.5498523 0.8400321 0.9492749
cor(1,3) 0.6120343 0.8658016 0.9579139
cor(1,4) 0.5441128 0.8402493 0.9502220
cor(2,3) 0.7027219 0.9007424 0.9692418
cor(2,4) 0.6543170 0.8836142 0.9641141
cor(3,4) 0.8453618 0.9509953 0.9850549

 Variance function:
       lower      est.    upper
10 0.6540629 0.9049686 1.252125      ## These are ratios relative to sigma.
12 0.8234096 1.1144608 1.508390      ## All intervals contain 1
14 0.8275472 1.1546989 1.611182      ## So separate variances are not needed.

 Residual standard error:
   lower     est.    upper
1.366622 2.099919 3.226686
> anova(OrthoFem.gls,OrthoFem.gls2) ## compare variance models using REML fits
              Model df      AIC      BIC    logLik   Test  L.Ratio p-value
OrthoFem.gls      1  9 154.4946 170.1336 -68.24731
OrthoFem.gls2     2 12 157.7655 178.6175 -66.88274 1 vs 2 2.729147  0.4353
```

From the test above we conclude that a single variance is adequate. Now can we simplify the correlation structure?

```
> intervals(OrthoFem.gls)
Approximate 95% confidence intervals

 Coefficients:
                 lower       est.      upper
(Intercept) 15.5150955 17.3555178 19.1959400
age          0.3558061  0.4820508  0.6082955

 Correlation structure:
            lower      est.     upper
cor(1,2) 0.5333553 0.8497764 0.9575759
cor(1,3) 0.6153696 0.8613243 0.9544376     ## These correlations are
cor(1,4) 0.5298229 0.8265770 0.9430108     ##    all very similar,
cor(2,3) 0.6608392 0.8853514 0.9644278     ##  suggesting that we could
cor(2,4) 0.5786895 0.8553747 0.9554782     ##  try a simpler
cor(3,4) 0.8267779 0.9418193 0.9812431     ##  correlation structure
 Residual standard error:
   lower     est.    upper
1.463431 2.175092 3.232831
```

```
                      ##################  Step 3  ######################
> OrthoFem.gls3 <- update(OrthoFem.gls, corr=corCompSymm(form=~1|Subject))
> anova(OrthoFem.gls3,OrthoFem.gls)
             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
OrthoFem.gls3     1  4 149.2183 156.1690 -70.60916
OrthoFem.gls      2  9 154.4946 170.1336 -68.24731 1 vs 2 4.723697  0.4505
> intervals(OrthoFem.gls3)
Approximate 95% confidence intervals

 Coefficients:
                lower       est.     upper
(Intercept) 15.6397163 17.3727273 19.105738
age          0.3734149  0.4795455  0.585676

 Correlation structure:
      lower      est.     upper
Rho 0.70742 0.8754962 0.9515169

 Residual standard error:
   lower     est.    upper
1.484850 2.210660 3.291253
```

The correlation functions used were: `corSymm` which fit a general correlation model restricted only to being symmetric and PD, and `corCompSymm` where all correlations off-diagonal have the same estimated value. Through the use of correlation and weight options, we're able to obtain a simple model which only uses two parameters to describe $\mathbf{\Sigma}$, a single variance for the diagonal ($s^2 = 2.21^2 = 4.887$) and a covariance, $\rho \times \sigma^2$ estimated by $0.875 \times 2.211^2 = 4.28$ which appears in all off-diagonal locations.

Now in SAS:

```
 data OrthoFem;
infile "../data/OrthoFem" firstobs=2;
input   distance age Subject$ Sex$;
run;
  proc mixed;
     class Subject;
     model distance = age;
     repeated /sub=Subject type=un ;
  run;
  proc mixed;
     class Subject;
     model distance = age;
     repeated /sub=Subject type=cs ;
  run;
 ############################################################## Model 1;
                           The Mixed Procedure
                     Covariance Parameter Estimates

                    Cov Parm    Subject    Estimate
                    UN(1,1)     Subject      4.4097
                    UN(2,1)     Subject      3.3522
                    UN(2,2)     Subject      3.6113
                    UN(3,1)     Subject      4.2549
                    UN(3,2)     Subject      4.0059
                    UN(3,3)     Subject      5.4769
                    UN(4,1)     Subject      4.2784
```

```
                    UN(4,2)     Subject      4.0716
                    UN(4,3)     Subject      5.3966
                    UN(4,4)     Subject      5.8795

                            Fit Statistics
                 -2 Res Log Likelihood           133.8
                 AIC (smaller is better)         153.8
############################################################ Model 2;
                    Cov Parm     Subject    Estimate
                    CS           Subject      4.2786
                    Residual                  0.6085

                            Fit Statistics
                 -2 Res Log Likelihood           141.2
                 AIC (smaller is better)         145.2
```

Note that Model 2 with the simpler compound symmetry structure has smaller AIC. More on AIC later, but it is one way to compare models, and smaller values indicate better models.

We will come back to these data after we learn about random effects to see if different slopes and intercepts are needed for each individual.

3.2.4 Time Series Correlations

A time series is a sequence of observations collected over time. It is said to be "autocorrelated" if (loose definition) observations taken at different times are not independent of each other.

`corAR1` When time intervals are all equal, we may want to assume that measurements one interval apart have correlation ρ, two intervals apart ρ^2, t intervals apart ρ^t, so $\rho_{ij} = \rho^{|i-j|}$. There is a single parameter, and the correlation matrix has a banded structure:

$$\mathbf{R} = \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

The AR stands for autoregressive, because the errors are assumed to depend on 1 observation before.

$$\epsilon_{i,t} = u_{it} + \gamma\epsilon_{i,t-1}$$

The γ parameter has to be in the interval [-1, 1], a positive value (positive autocorrelation) means that positive residuals tend to follow positives and negatives follow negatives. With a negative autocorrelation (I have never seen one in nature) positives tend to follow negatives and vice versa.

Plots of the `ACF` (autocorrelation function) are useful in determining whether the data are auto-correlated.

`corCAR1` When time units are not equally spaced, we can get a structure similar to AR1 by using Continuous AutoRegressive model (CAR1).

$$cor(\epsilon_{i,t}, \epsilon_{i,s}) = \rho^{|s-t|}.$$

We do have to specify the time covariate.

`corARMA` Auto-Regressive Moving Average can model more complex relationships – the auto regressive part can go back to more preceding time steps, and the moving average can go several steps back as well. Each AR step and each MA step take one more parameter.

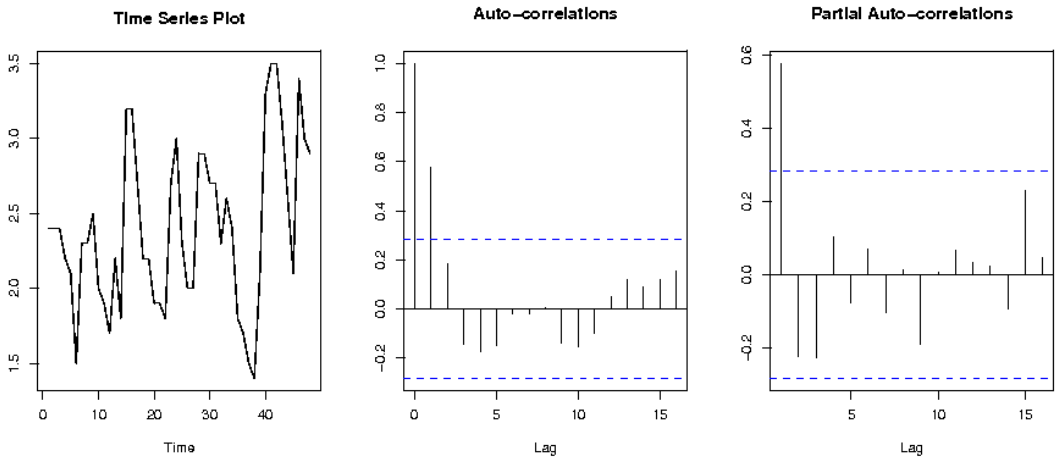
Identifying the problem

With positive autocorrelation, residuals tend to cluster above and below the 0 line because positives are usually followed by positives and negatives by negatives. Negative autocorrelation causes excessive "jumping" because positives are usually followed by negatives and negatives by positives.

A graphical display is available in R. The `acf` function computes correlations of points which are "lag" $l = 1, 2, \ldots$ time steps apart. The value at lag $= 0$ will always be 1; it is plotted only for reference. Examine the correlations as a function of lag to see if autocorrelation weakens with time. The compound symmetric structure we observed previously will show up as lines of approximately the same height.

```
> library(MASS)
> par(mfrow=c(1,3))
> plot(lh,main="Time Series Plot")
> acf(lh,main = "Auto-correlations")
> pacf(lh,main = "Partial Auto-correlations")
```

The data consists of 48 samples drawn from one woman at 10 minute intervals. Each sample is measured for the concentration of luteinizing hormone.

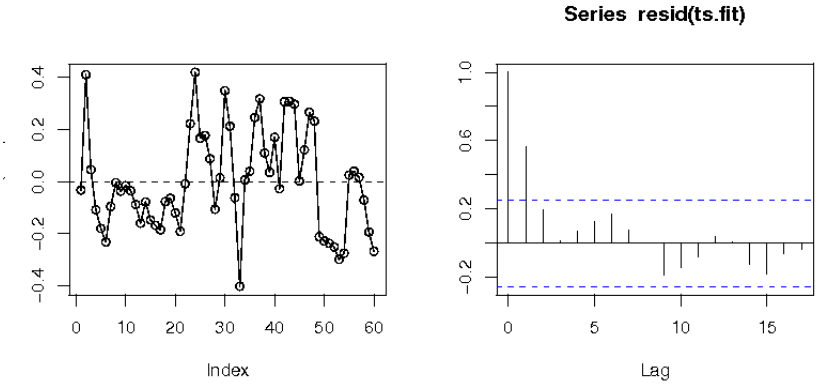


The first plot shows the raw data, the second the autocorrelations at lags 0 to 16. Lag 1 shows a fairly strong positive autocorrelation of about .57, but the other lags seem to show just noise. The dotted lines are intended to indicate non-significance (at the 5% level) of autocorrelations contained with them. The third plot is partial autocorrelation, which adjusts for observations between times t and s before computing the lag $|t-s|$ correlation. All of these appear to be ignorable.

In Stat 506 the interest is in checking **residuals** for autocorrelation.

```
> library(lmtest)
> data(bondyield)
> ?bondyield
  A multivariate quarterly time series from 1961(1) to 1975(4) with
```

```
  variables.
     RAARUS difference of interest rate on government and corporate
          bonds,
     MOOD measure of consumer sentiment,
     EPI index of employment pressure,
     EXP interest rate expectations,
     Y artifical time series based on RAARUS.
> plot(resid(lm( RAARUS ~ MOOD + EPI + EXP + RUS, data=bondyield)), typ="l")
> abline(h=0, lty=2)
> require(nlme)
> ts.fit = gls( RAARUS ~ MOOD + EPI + EXP + RUS, data=bondyield)
> acf(resid(ts.fit))
```



Note the runs of all negative and all positive residuals.

```
>  ts.fit2 <- update(ts.fit, correlation=corAR1())

> anova(ts.fit,ts.fit2)
        Model df       AIC       BIC    logLik   Test L.Ratio p-value
ts.fit      1  6  12.24085 24.284849 -0.120425
ts.fit2     2  7 -22.05554 -8.004213 18.027773 1 vs 2 36.2964  <.0001
```

Fitting auto-regressive correlation of order 1 improves the AIC by 34 units and results in a significantly better model according to the LRT.

Durbin-Watson Test for serial correlation in a single time series:

$$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

Values of d close to 2 indicate lack of autocorrelation ($\rho \approx 0$), while values close to 0 indicate positive AR1() with ρ near 1; values close to 4 indicate negative AR1(), ρ near -1.

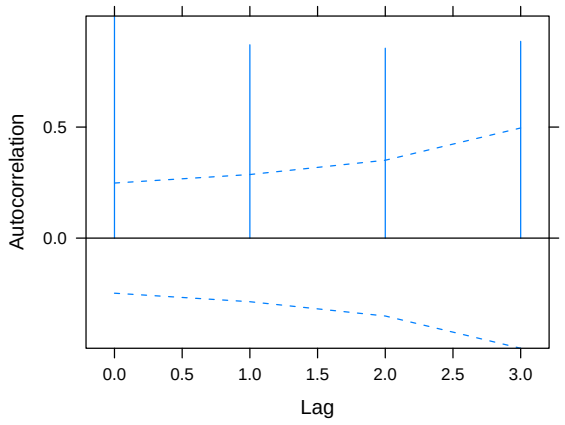
```
> dwtest(lm( RAARUS ~ MOOD + EPI + EXP + RUS, data=bondyield))

Durbin-Watson test

data:  lm(RAARUS ~ MOOD + EPI + EXP + RUS, data = bondyield)
DW = 0.8448, p-value = 2.888e-08
alternative hypothesis: true autocorrelation is greater than 0
```

With the OrthoFem data, AR1 is an alternative to compound symmetric structure, which has nearly the same log-likelihood and AIC (both use one parameter).

```
>  OrthoFem.gls4 <- update( OrthoFem.gls, correlation = corAR1(form=~1|Subject))
> AIC( OrthoFem.gls, OrthoFem.gls3, OrthoFem.gls4)
              df      AIC
OrthoFem.gls   9 154.4946
OrthoFem.gls3  4 149.2183
OrthoFem.gls4  4 149.3798
```

Recall that this data consists of 11 parallel time series. The `acf` function does not work with multiple series, but the `ACF` function in the `nlme` package produces the same plots for such cases.

```
> plot(ACF(OrthoFem.gls, form = ~1|Subject), alpha=.10)
```



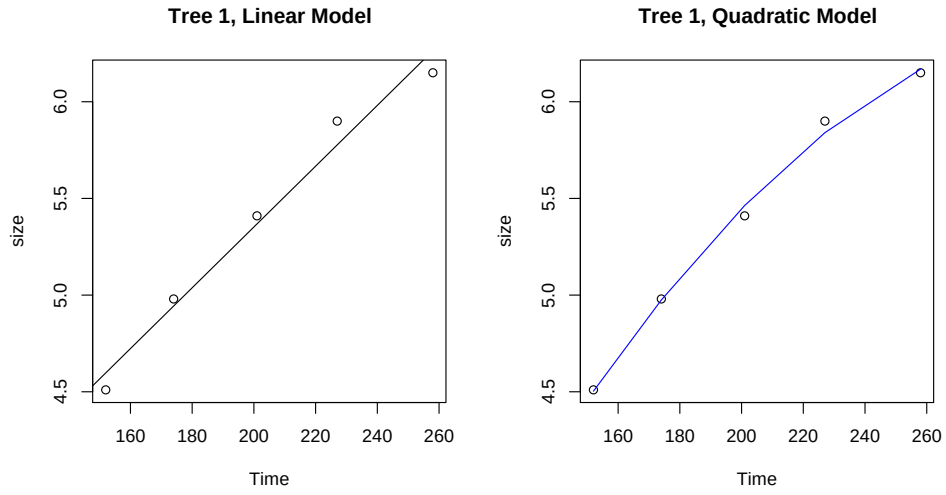
This looks like compound symmetric correlation structure.

In SAS Proc Mixed we can fit AR1() and ARMA(p,q) correlations and many other variance-covariance structures. These can be specified either in a random statement or in a repeated statement.

Facts about time series:

- Time series models are used to model a large variance-covariance matrix with a few parameters.
- In AR1(), absolute correlation decreases with lag, l, as ρ^l. If $\rho < 0$, then autocorrelations jump from negative to positive and back. In an AR(p) process, partial autocorrelations drop to zero for lag $> p$.
- MA(q) models have autocorrelations which drop to zero after lag q. The first autocorrelation in a MA1() model will be in (-.5, .5). Partial auto-correlations decrease exponentially with lag.
- We can combine AR(p) with MA(q) models to get ARMA(p, q) models.
- Time series structures work within linear models, because they define the structure of $\boldsymbol{V}$. We can't use GLS because the parameters are unknown but can use estimated GLS techniques.

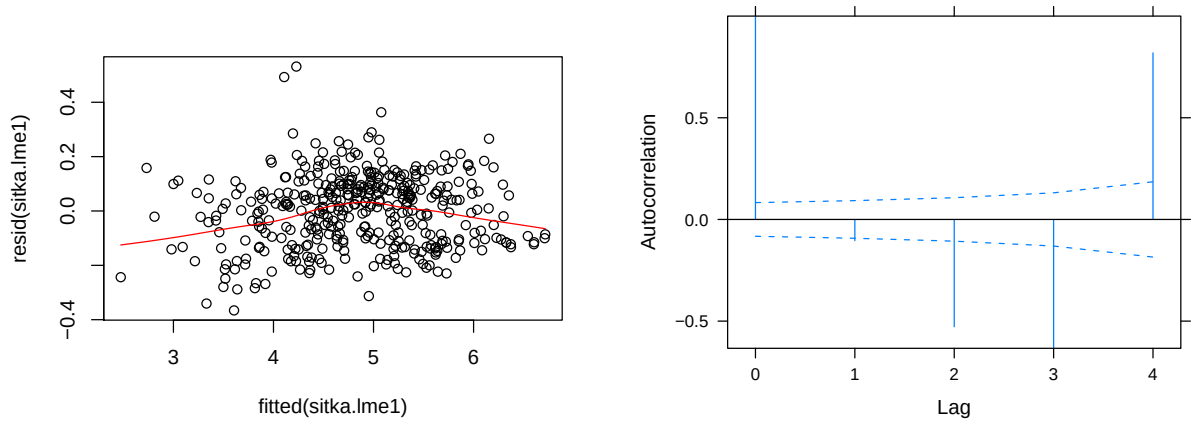
Caution:
An ill-fitting model can generate spurious correlations. For example, if I try to fit a straight line to quadratic data, I might underfit the middle points and overfit the extremes.



What happens to the autocorrelations? The above tree was measured 5 times at 24 month intervals from month 152 to 248. If the pattern for tree 1 is typical (underfit in the middle) for all 80 trees in this study, what signs do you expect to see in the following correlations:

```
times   sign(+, - or 0)
1 to 2                \
2 to 3                 \lag 1 sign overall ?
3 to 4                 /
4 to 5                /

1 to 3                \
3 to 5                  lag 2 sign overall ?
2 to 4                /

1 to 4                \ lag 3 sign overall ?
2 to 5                /

1 to 5                  lag 4 sign overall ?
```

We need to have a decent model before trying to fit correlation structures. Residual diagnostics are important.



3.2.5 Spatial Correlations

Spatial correlation structures are similar to time series in that correlations are strong for units close together, and weaken as points get farther away, so distance plays an important role. We might have a single variable which measures distance like position on a transect, or we might need two variables (x, y), (easting, northing), or (latitude, longitude) or even 3 variables to describe distance. In SAS Proc Mixed (Littell et al. 2006) and in the `nlme` S package (Pinheiro and Bates 2000), one can specify spatial correlations.

With time series data we looked at autocorrelation and partial autocorrelation plots, empirically estimated correlations for all pairs of points t time steps apart. We could do the same for spatial data pooling together information from all points distance d apart and letting d vary across the distances in the data. However, there is a strong tradition in spatial modeling to plot the "semi-variogram" instead of an autocorrelation function. Before defining semivariogram we need to talk about distance.

Represent ϵ_x as the data (we'll again be looking at residuals) collected at a point with coordinates $\boldsymbol{x} = (x_1, x_2, \ldots, x_r)^\mathsf{T}$. Similarly, ϵ_y is data from a point with coordinates $\boldsymbol{y} = (y_1, y_2, \ldots, y_r)^\mathsf{T}$ Let $d()$ be a distance function, the usual choices are:

- Euclidean distance: $d_E(\epsilon_x, \epsilon_y) = \sqrt{\sum_{i=1}^r (x_i - y_i)^2}$
- Manhattan distance: $d_{Man}(\epsilon_x, \epsilon_y) = \sum_{i=1}^r |x_i - y_i|$ (also called taxi-cab distance)
- Maximum distance: $d_{Max}(\epsilon_x, \epsilon_y) = \max_{i=1,\ldots,r} |x_i - y_i|$
- Adjacency: $d_{adj} = 1$ if point $\boldsymbol{x}$ is a neighbor of point $\boldsymbol{y}$, 0 otherwise. Adjacency is used on regions like counties or states, but does not work with semivariograms, and we won't be discussing it further.

Spatial correlation structures assume that points close together are more strongly correlated, and that correlation decreases with distance. The semivariogram is defined as:

$$\gamma[d(\epsilon_x,\ \epsilon_y), \boldsymbol{\lambda}] = \frac{1}{2}\mathrm{Var}(\epsilon_x - \epsilon_y) = \frac{1}{2}E(\epsilon_x - \epsilon_y)^2$$

We set $\mathrm{Var}(\epsilon_x) = 1, \forall \boldsymbol{x}$ (wlog because we are interested in correlations, not variances) and then γ depends only on distance, d and correlations, $\boldsymbol{\rho}$, and $\gamma(d, \boldsymbol{\rho}) = 1 - h(d, \boldsymbol{\rho})$ where $h(d, \boldsymbol{\rho})$ is the correlation of two points distance d apart. Interpretation: at distance near zero one would expect to see strong correlations and small semivariogram values. As distance increases, the semivariogram should increase asymptotically to one.

"Nugget" effects are often included to explain excess variation (lower than expected correlation) of points close to distance 0 apart, meaning that as $d \longrightarrow 0$, the semivariogram approaches the nugget value, c_0, and correlation, $h(d, \boldsymbol{\rho})$, approaches $1 - c_0 < 1$.

Two methods are used to estimate the semivariogram, each using the standardized residuals, $r_{ij} = (y_{ij} - \hat{y}_{ij})/\hat{\sigma}_{ij}$ for the $N(d)$ points which are separated by distance d.

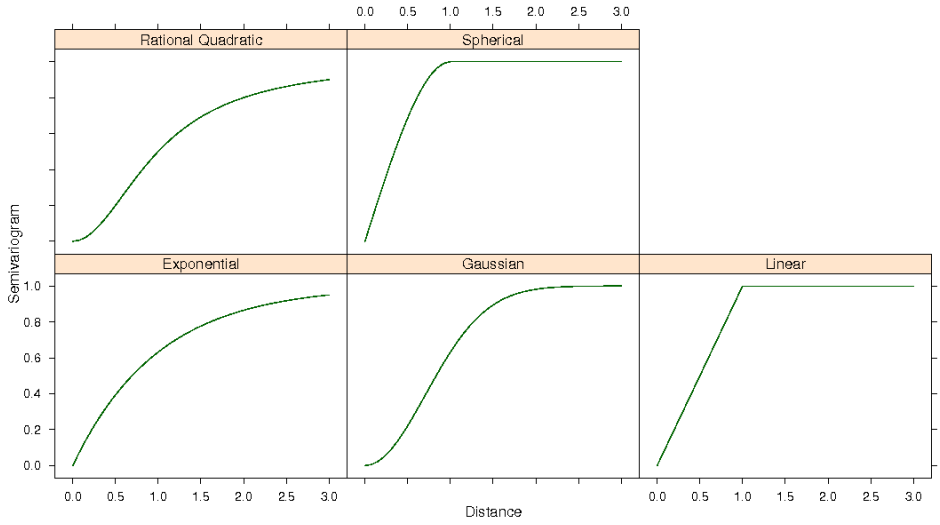
$$\text{classical: } \widehat{\gamma}(d) = \frac{1}{2N(d)} \sum_{i=1}^{M} \sum_{j,j' | d(p_{ij}, p_{ij'}) = d} (r_{ij} - r_{ij'})^2$$

$$\text{robust: } \widehat{\gamma}(d) = \left(\frac{1}{2N(d)} \sum_{i=1}^{M} \sum_{j,j' | d(p_{ij}, p_{ij'}) = d} |r_{ij} - r_{ij'}|^{1/2} \right)^4 \times \frac{N(d)}{0.457N(d) + 0.494}$$

Parametric models for semivariograms include:

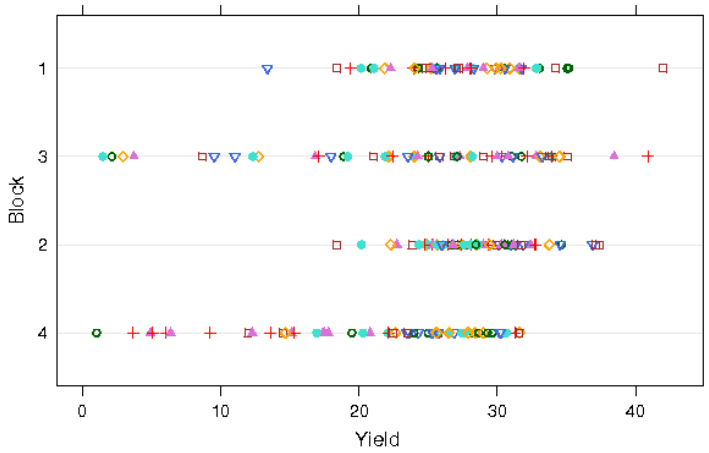
Model	Formula
Exponential:	$\gamma(d,\rho) = 1 - \exp(-d/\rho)$
Gaussian:	$\gamma(d,\rho) = 1 - \exp[-(d/\rho)^2]$
Linear:	$\gamma(d,\rho) = 1 - (1 - d/\rho)I(d < \rho)$
Rational Quadratic:	$\gamma(d,\rho) = (d/\rho)^2/[1 + (d/\rho)^2]$
Spherical:	$\gamma(d,\rho) = 1 - [1 - 1.5(d/\rho) + 0.5(d/\rho)^3]I(d < \rho)]$

The role of the above functions is similar to the role of AR and MA functions in time series models: if we select one of the above models, then we can model all the spatial correlations with only two parameters, range and nugget. Note that for Linear and Spherical models, the semivariogram is 1 for $d >= \rho$. The parameter ρ is called the range, (for all models) and is the distance at which two points are uncorrelated (or for other models, approximately uncorrelated). Here is a plot of the 5 types, each with $\rho = 1$ over distances 0 to 3.



Example

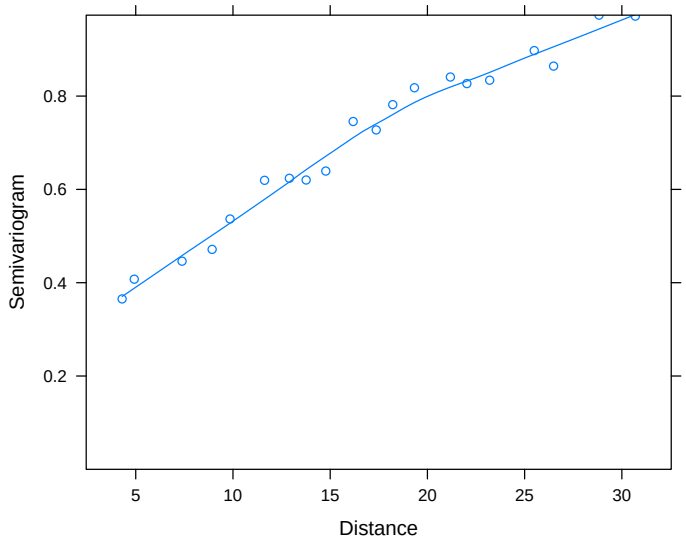
A large trial of wheat varieties used 4 blocks with each of 56 varieties in each block.



There is certainly a random block effect, but blocks are so large that we cannot assume each block is homogeneous. We can, instead of using a random block effect, assume a spatial structure, with neighboring plots assumed more similar than distant plots. Distance will be taken to be Euclidean from center of one plot to another, measured by latitude

and longitude. We start with a naive no-correlation, model, and look at the empirical semivariogram.

```
> library(nlme)
> data(Wheat2)
> Wheat.glsfit1 = gls(yield ~ variety -1, data = Wheat2)
> plot(Variogram(Wheat.glsfit1, form = ~ latitude + longitude, max = 32))
```



We see that there is an increase in the semivariogram with distance, indicating a spatial correlation. The plot does not clearly favor any of the possible structures, so I tried fitting each correlation structure and then I compared the fits. For each starting values were given to make the range 28 and nugget = 0.2.

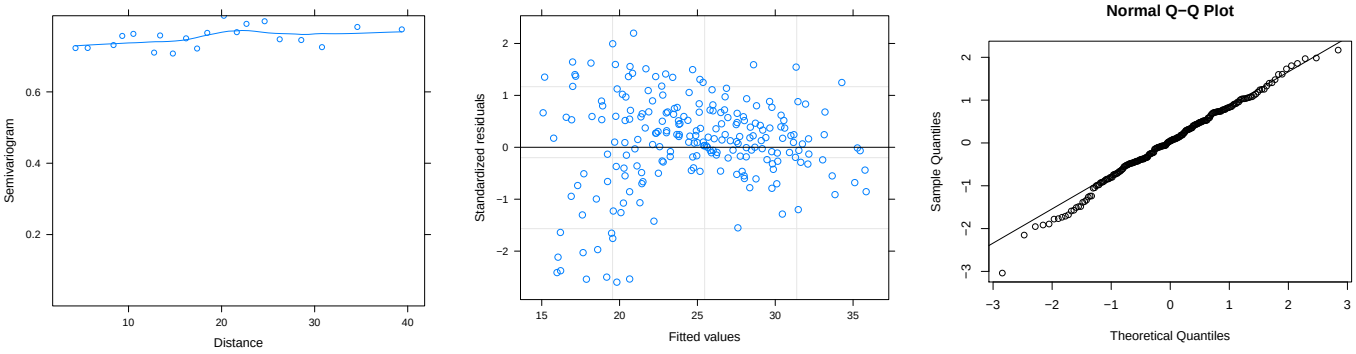
```
> Wheat.glsSpher = update(Wheat.glsfit1, corr=corSpher(c(28,.2),
    form = ~ latitude + longitude, nugget=T))
> Wheat.glsRatio = update(Wheat.glsfit1, corr=corRatio(c(12,.2),
    form = ~ latitude + longitude, nugget=T))
> Wheat.glsLin = update(Wheat.glsfit1, corr=corLin(c(28,.2),
    form = ~ latitude + longitude, nugget=T))
> Wheat.glsExp = update(Wheat.glsfit1, corr=corExp(c(28,.2),
    form = ~ latitude + longitude, nugget=T))
> Wheat.glsGauss = update(Wheat.glsfit1, corr=corGaus(c(28,.2),
    form = ~ latitude + longitude, nugget=T))
> anova(Wheat.glsfit1, Wheat.glsSpher,Wheat.glsRatio,Wheat.glsLin,Wheat.glsExp,
         Wheat.glsGauss, test=F)
               Model df      AIC      BIC    logLik
Wheat.glsfit1      1 57 1354.742 1532.808 -620.3709
Wheat.glsSpher     2 59 1185.863 1370.177 -533.9315
Wheat.glsRatio     3 59 1183.278 1367.592 -532.6389
Wheat.glsLin       4 59 1184.837 1369.151 -533.4185
Wheat.glsExp       5 59 1184.855 1369.169 -533.4274
Wheat.glsGauss     6 59 1185.102 1369.416 -533.5509
```

All the spatial correlation models improve the fit, as the log-likelihood increases by roughly 90, and there is little difference between the log-likelihoods of the different correlation structures (1.3 from smallest to largest). The rational quadratic does the best in terms of likelihood and AIC/BIC, so we will test to see if its correlation structure significantly improves the fit.

```
> anova(Wheat.glsfit1, Wheat.glsRatio)
                Model df      AIC      BIC    logLik   Test L.Ratio p-value
Wheat.glsfit1       1 57 1354.742 1532.808 -620.3709
Wheat.glsRatio      2 59 1183.278 1367.592 -532.6389 1 vs 2 175.464  <.0001
```

The two added parameters (range and nugget) improve the fit immensely, so the correlation proves its worth.

To check the fit of the model, here is a semivariogram of the adjusted residuals, a plot of residuals versus fits, and a normal "qq" plot.




```
> plot(Variogram(Wheat.glsRatio, resType="n"))
> plot(Wheat.glsRatio)
> qqnorm(resid(Wheat.glsRatio, type="n"))
> qqline(resid(Wheat.glsRatio, type="n"))
```

It is possible to use a "test for problems" approach as in Sen and Srivastava (1997). They define Geary's test for spatial correlation (similar to the Durbin-Watson statistic over times) with a weight, w_{ij} which must decrease as points i and j become farther apart. It is available in R in the `spdep` package. Searching the SAS site, I see that it is now available in SAS 9.2 Proc Variogram.

$$\text{Geary's test: } c = \frac{\sum_{i,j=1}^{n} w_{ij}(e_i - e_j)^2}{s^2}$$

Our approach will be to first plot the semivariogram to see if it increases. If so, we'll fit a few of the possible spatial correlation structures and then test to see if they improve the model via LRT, as above.

Summary of Correlation/Variance Functions in nlme

- `corSymm` general symmetric structure, $n(n-1)/2$ correlations are fit
- `corCompSymm` compound symmetric sets every correlation to be equal.
- `corAR1` Autoregressive of order 1
- `corARMA` Autoregressive of order p, moving average of order q (either p or q could be 0)
- Spatial structures: corSpher, corRatio, corLin, corExp, corGaus

- `corCAR1` is used for time series data when times are not equally spaced. It is basically the same as corExp with a 1-dimensional (time) distance.

We have also used two functions which allow the variances (or standard deviations) to change on the diagonal of $\boldsymbol{V}$,

- `varPower` sets $\sigma^2 \propto \mu^{2\alpha}$ and estimates α. (also allows variance as a power of some other predictor)
- `varIdent(form = ~1 | Factor)` estimates a separate variance for each level of some factor.

SAS Proc Mixed

In a repeated measures study where we have several units/subject measured repeatedly, use a `repeated` statement, which has some similar options, but does not specify correlations separately from variances. For instance, `UN` for unstructured (like `corSymm`), `UNR` unstructured with different variances, `CS` for compound symmetric, and `CSH` for compound symmetric - heteroschedastic. AR and ARMA(1,1) are included along with spatial structures (`SP(EXP)` for exponential) and others we've not used.

The direct product matrices are an improvement over the `nlme` models, and allow one to fit both spatial and temporal effects in the same model.

Cautions:

Any application of Estimated Generalized Least Squares is necessarily iterative. One must use residuals to estimate the variance/covariance parameters, and the residuals must come from some fitted model. The algorithms iterate between estimating $\boldsymbol{\beta}$ and estimating $\boldsymbol{V}$ until things stop changing. The maximization routines have difficulty when log-likelihood is flat over some region of the parameter space, and they do get stuck in local maxima when there's a larger global maximum elsewhere. Creating robust computer algorithms to maximize these functions is a tough problem. SAS and nlme have chosen different approaches and do not offer the same types of correlations/variance models. Even when they offer the same models, estimates may not agree due to differences in algorithms.

In R, do run the intervals command on any fitted model from `gls` because it might tell you that variances of the estimates are unobtainable ("the Hessian is singular"). In that case you'll need to simplify your model.

3.3 Non-Normality

First we should note that normality of residuals,ϵ_i, is just one way that we can obtain normality of $\tilde{\boldsymbol{\beta}}$. If $\boldsymbol{\epsilon}$ has a normal distribution, then any linear combination of $\boldsymbol{\epsilon}$ also does, including $\boldsymbol{y}$ and $\tilde{\boldsymbol{\beta}}$.

3.3.1 Central Limit Theorem

The CLT you learn in Stat 502 applies to sums or averages of independent R.V.s. We need a slightly different version to apply to coefficient estimates, which are much like means.
From Math Stat:
Lindeberg CLT (Arnold 1981) says that if $Y_1, Y_2, \ldots \sim (0, \sigma^2)$ with $\sigma^2 < \infty$, and $c_1, c_2, \ldots$ is a sequence of constants such that $\frac{\max c_j^2}{\sum c_j^2} \longrightarrow 0$ as $n \to \infty$, then $\frac{\sum_{i=1}^n c_j Y_j}{\sqrt{\sum_{j=1}^n c_j^2}}$ converges in distribution to $N(0, \sigma^2)$. The condition keeps the c_j's from increasing so fast that the last term dominates the sum.

Simple case:
Suppose we have a balanced ANOVA model (1, 2, or multiway) with n observations per cell, $Y_{ij} - \mu_i \sim$ iid$(0, \sigma^2)$. Take the mean in group i to be $\frac{1}{n}\sum(Y_{ij} - \mu_i)$, so let $c_j = n^{-1}$. The condition is easily satisfied because $\max(c_j^2) = n^{-2}$ and $\sum c_j^2 = n \times n^{-2} = n^{-1}$ so the fraction is n^{-1} which goes to 0 nicely as $n \to \infty$. We conclude

$$\frac{n^{-1}\sum(Y_{ij} - \mu_i)}{n^{-1/2}} = n^{1/2}(\overline{Y}_i - \mu_i) \xrightarrow{\mathcal{L}} N(0, \sigma^2) \text{ or } n^{1/2}\overline{Y}_i \xrightarrow{\mathcal{L}} N(\mu_i, \sigma^2).$$

Taking a linear combination of means also yields a normal distribution. For any n, the cell means are independent of each other.

Full linear model (Sen and Srivastava 1997)
Change the condition for the CLT slightly, letting $a_i = c_i/\sqrt{\sum c_i^2}$ and require $\max |a_{n_i}| \longrightarrow 0$ and $\sum a_{n_i}^2 \longrightarrow 1$. This allows the use of a triangular array of constants instead of reusing the same values (Gnedenko and Kolmogorov, 1954). The conclusion is that $\sum a_{n_i} Y_i \xrightarrow{\mathcal{L}} N(0, \sigma^2)$ as $n \to \infty$.
Linear model:
Take the simple case: $\boldsymbol{y} \sim (\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I})$ with $\boldsymbol{X}$ of full column rank, and consider the coefficient vector $(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})$, or actually the distribution of

$$\begin{aligned}
\sigma^{-2}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) &= \sigma^{-2}[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{\beta}]^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{\beta}] \\
&= \sigma^{-2}[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}(\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{\beta})]^\mathsf{T}\boldsymbol{X}^\mathsf{T}\boldsymbol{X}[(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}(\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{\beta})] \\
&= \sigma^{-2}(\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{\beta})^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}(\boldsymbol{X}^\mathsf{T}\boldsymbol{y} - \boldsymbol{X}^\mathsf{T}\boldsymbol{X}\boldsymbol{\beta}) \\
&= \sigma^{-2}\boldsymbol{\epsilon}^\mathsf{T}\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{\epsilon} \\
&= \sigma^{-2}\boldsymbol{\epsilon}^\mathsf{T}\boldsymbol{H}\boldsymbol{\epsilon} = \sigma^{-2}\boldsymbol{\epsilon}^\mathsf{T}\boldsymbol{P}_{\text{ppo}}\boldsymbol{\epsilon}
\end{aligned}$$

Use the full rank SVD or eigenvalue decomposition on $\boldsymbol{H}_n$ to obtain $\boldsymbol{H} = \boldsymbol{L}_n^\mathsf{T}\boldsymbol{L}_n$ with $\boldsymbol{L}_n\boldsymbol{L}_n^\mathsf{T} = \boldsymbol{I}_r$. Take a vector $\boldsymbol{a}$ s.t. $\boldsymbol{a}^\mathsf{T}\boldsymbol{a} = 1 = \sum a_i^2$ and let $b_n^{(i)} = \boldsymbol{a}^\mathsf{T}\boldsymbol{L}_n^{(i)}$ (a scalar), then

look at $\boldsymbol{a}^\mathsf{T}\boldsymbol{L}_n\boldsymbol{\epsilon} = \sum_{i=1}^n b_n^{(i)}\epsilon_i$.

$$|b_n^{(i)}| = |\boldsymbol{a}^\mathsf{T}\boldsymbol{L}_n^{(i)}| \leq (\boldsymbol{a}^\mathsf{T}\boldsymbol{a})^{1/2}(\boldsymbol{L}_n^{(i)\mathsf{T}}\boldsymbol{L}_n^{(i)})^{1/2} \longrightarrow 0$$

if $\max h_{ii} = \max \boldsymbol{L}_n^{(i)\mathsf{T}}\boldsymbol{L}_n^{(i)} \longrightarrow 0$. So the condition needed is that diagonals of the hat matrix, the leverages, must go to zero as $n \longrightarrow 0$. Then $\sum(b_n^{(i)})^2 = \boldsymbol{a}^\mathsf{T}\boldsymbol{L}_n\boldsymbol{L}_n^\mathsf{T}\boldsymbol{a} = \boldsymbol{a}^\mathsf{T}\boldsymbol{a} = 1$ so we conclude that $\boldsymbol{a}^\mathsf{T}\boldsymbol{L}_n\boldsymbol{\epsilon} \longrightarrow N(0,\sigma^2)$ for every norm one vector $\boldsymbol{a}$. By properties of MVN, $\boldsymbol{L}_n\boldsymbol{\epsilon}$ has an asymptotic $N(\boldsymbol{0},\sigma^2\boldsymbol{I})$ distribution. Hence

$$\sigma^{-2}(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta})^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}) \xrightarrow{\mathcal{D}} \chi^2_r.$$

Also, s^2 converges to σ^2 in probability so

$$s^{-2}(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta})(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}) \xrightarrow{\mathcal{D}} \chi^2_r.$$

In general if $\boldsymbol{C}$ is a p by q matrix of rank q and $\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta}$ is estimable, then

$$s^{-2}(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta})^\mathsf{T}[\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^g\boldsymbol{C}]^{-1}(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta}) \xrightarrow{\mathcal{D}} \chi^2_q.$$

However, the above assumes that s^2 is a "perfect" estimate of the variance. We generally prefer to not make that assumption and use the F distribution instead.

$$\frac{1}{qs^2}(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta})^\mathsf{T}[\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^g\boldsymbol{C}]^{-1}(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}}-\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta}) \xrightarrow{\mathcal{D}} F_{q,n-r}.$$

The critical assumption is that no single point is entirely "influential", but the leverage of each is going to zero. We shall see that $\sum h_{ii} = \text{trace}(\boldsymbol{H}) = \text{rank}(\boldsymbol{X})$ which does not increase as n does.

More generally, the above also works with $\text{Var}(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{V}$ for known $\boldsymbol{V}$ in that as n gets big, the sampling distribution of $\widehat{\boldsymbol{\beta}}$ converges to a normal distribution. In practice, we never know exactly how big n needs to be to make the sampling distribution close enough to normality so that our inferences based on t and F distributions are valid. Years of experience (and simulation) indicate that these distributions work well for sample sizes in the hundreds.

3.3.2 Bootstrap

Goal: to make inferences about parameters, e.g. about $\boldsymbol{C}'\boldsymbol{\beta}$ without assuming normality.

Under normality we use the sampling distribution, $\boldsymbol{C}'\widehat{\boldsymbol{\beta}} \sim N(\boldsymbol{C}^\mathsf{T}\boldsymbol{\beta},\sigma^2\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{C})$, and plug in s^2 for σ^2 to get a sampling distribution based on an F (or t) distribution. It is used for testing and confidence intervals.

The *sampling distribution* describes how a statistic varies under repeated sampling (observational) or repeated random allocation of treatments to units (experiment). We bootstrap when we don't know the sampling distribution of the statistic. (Efron, 1979, Annals of Statistics)

Instead of using the sampling distribution, we use an approximation called the *resampling distribution*. Take a new sample from the original sample allowing replacements. For each resample, compute the statistic of interest, then look at how the statistic varies across many resamples. Using notation from Efron and Tibshirani (1993).

Original data		Statistic
$\boldsymbol{X}$	$\longrightarrow$	$\widehat{\theta}$
	Resamples	
$\boldsymbol{X}^{*1}$	$\longrightarrow$	$\widehat{\theta}^{*(1)}$
$\boldsymbol{X}^{*2}$	$\longrightarrow$	$\widehat{\theta}^{*(2)}$
$\cdots$		$\cdots$
$\boldsymbol{X}^{*B}$	$\longrightarrow$	$\widehat{\theta}^{*(B)}$

We now have lots of statistics, and can look at their distribution (the resampling distribution). For example, compute their resampling standard error:

$$\widehat{se}_B(\widehat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}[\widehat{\theta}^{*(b)} - \overline{\widehat{\theta}^*}]^2} \quad \text{where} \quad \overline{\widehat{\theta}^*} = \frac{1}{B}\sum \widehat{\theta}^{*(b)}$$

Some Theory
A function which takes another function as an argument is called a functional. Think of θ as the results of applying a functional to F, the true cdf of the unknown distribution.

$$\theta = t(F)$$

Example: $\theta = median = F^{-1}(.5)$ is a functional of F.
We plug in $\widehat{F}$ to the same functional, $\widehat{\theta} = t(\widehat{F})$. In this example, $\widehat{\theta}$ is a median of the data. We cannot repeatedly sample to learn about F, but we can take resamples to get lots of $\widehat{\theta}^*$'s. Under generous conditions, the relationship between $\widehat{\theta}^*$'s and $\widehat{F}$ is like that of $\widehat{\theta}$ to F.

Example: θ is a median from 40 iid Exp(1) variates. Find a 95% CI for θ. (True value is known to be $\ln(2) = 0.6931$)

```
> y <- rexp(40)  ## original data
> med.fun <- function(y, indx) median(y[indx])
> bootM <- rep(0,500)
> for(b in 1:500){
    bootM[b] <- med.fun(y, sample(1:40, replace=T))}
> quantile(bootM, c(.025, .50, .975))
 0.385  0.779  1.081
```

Note that the sample function is applied to the index values **1:n**, rather than to the individual values. Instead of looking at a new resample of size n with some of the originals used repeatedly and some not at all, we are looking at the index numbers showing which values are in this resample. Handling index numbers instead of data values is a very handy trick when data are more complex.

Bootstrapping in Linear Models

We must consider the pairs $(\boldsymbol{x}_i, y_i)$ when resampling. Typically we have a dataset which includes both y_i and $\boldsymbol{x}_i$ in the same row, so it is convenient to simply sample from the **rows** of the data frame. There are two methods depending on how the data were collected:

1. Model–based Bootstrapping is applicable when the only $\boldsymbol{X}$ we care about is the one observed. For example in a designed experiment where $\boldsymbol{X}$ contains the settings (temperature, time, etc.) of interest, and these particular values were selected. The errors, $\boldsymbol{\epsilon} \sim (\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ are assumed exchangeable (a weaker assumption than iid).

2. Case–based Bootstrapping is used when $\boldsymbol{x}_i$ as well as y_i is random, as in observational studies, or for covariates in an experimental setting.

Model-Based Bootstrap

Each "resampled" dataset must be the same size as the original. We can use the original $\boldsymbol{X}$ matrix, the OLS estimator, $\widehat{\boldsymbol{\beta}}$, and the residual vector from the OLS fit, $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$, because all these values are known and fixed under resampling. (No surprise to you, it is better to use modified residuals here to account for the leverages, $r_i = e_i/\sqrt{1-h_{ii}}$.) Each resample is built using a resample of the residuals, $\boldsymbol{e}^*_{(b)}$, drawn with replacement from $\boldsymbol{e}$, and of the same length, n. B of the $\boldsymbol{e}^*_{(b)}$ vectors are needed. For each, create the resampled $\boldsymbol{y}^*_{(b)}$ vector as $\boldsymbol{X}\widehat{\boldsymbol{\beta}} + \boldsymbol{e}^*_{(b)}$, and compute a bootstrap coefficient vector,

$$\widehat{\boldsymbol{\beta}}^*_{(b)} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}^*_{(b)}$$

which is the object of our attention. We want to see how these coefficient vectors change under resampling. Adding zero:

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}^*_{(b)} &= (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}(\boldsymbol{y} + \boldsymbol{y}^*_{(b)} - \boldsymbol{y}) \\
&= (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} + (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}(\boldsymbol{y}^*_{(b)} - \boldsymbol{y}) \\
&= \widehat{\boldsymbol{\beta}} + (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}(\boldsymbol{e}^*_{(b)} - \boldsymbol{e}) \\
&= \widehat{\boldsymbol{\beta}} + (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{e}^*_{(b)}
\end{aligned}$$

To consider the **resampling** variance, $\widehat{\boldsymbol{\beta}}$ and $\boldsymbol{e}$ are fixed, and only $\boldsymbol{e}^*_{(b)}$ changes, so

$$\text{resamplingVar}(\widehat{\boldsymbol{\beta}}^*_{(b)}) = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}[\text{resamplingVar}(\boldsymbol{e}^*_{(b)})]\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}$$

The resampling variance of $\boldsymbol{e}^*_{(b)}$ is an $n \times n$ matrix, but we may assume it is of the form $\sigma^{*2}\boldsymbol{I}$, leaving only one parameter to estimate.

Now consider the testing situation, $H_0 : \boldsymbol{C}^\mathsf{T}\boldsymbol{\beta} = \boldsymbol{\delta}$ versus $H_a : \boldsymbol{C}^\mathsf{T}\boldsymbol{\beta} \neq \boldsymbol{\delta}$ for an estimable function. Our test statistic has been:

$$F = \frac{(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}} - \boldsymbol{\delta})^\mathsf{T}[\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{C}]^{-1}(\boldsymbol{C}^\mathsf{T}\widehat{\boldsymbol{\beta}} - \boldsymbol{\delta})/m}{s^2}$$

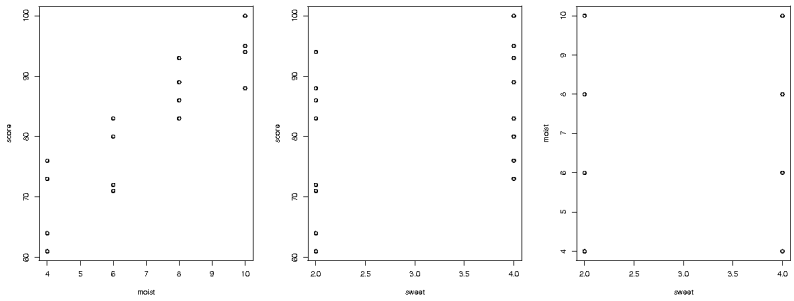
We can still use this statistic, but need to know how it varies with resampling. Multiply by m, since we are no longer trying to get a particular distribution, and for each $\boldsymbol{\beta}^*_{(b)}$, plug in $\widehat{\boldsymbol{\beta}} + (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}(\boldsymbol{e}^*_{(b)})$ to get

$$\begin{aligned}
F^* &= mF + (\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{e}^*_{(b)})^\mathsf{T}[\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{C}]^{-1}\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{e}^*_{(b)}/s^*_{(b)} \\
&= mF + {\boldsymbol{e}^*_{(b)}}^\mathsf{T}\{\boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{C}[\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{C}]^{-1}\boldsymbol{C}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\}\boldsymbol{e}^*_{(b)}/s^*_{(b)}
\end{aligned}$$

where $s^*_{(b)} = \frac{1}{n-r}\frac{n}{n-1}\sum(e^*_{(b)i} - \overline{e}_{(b)})^2$. Notice that the matrix product in the middle with $\boldsymbol{C}$'s and $\boldsymbol{X}$'s does not change under resampling. Also, under the null hypothesis, $mF = 0$. To

perform the bootstrap test, take $B = 500$ resamples and compute $F^*_{(b)}$ for each. Compare the test statistic, F, with the percentiles of the resamples. If it's more extreme than the 95^{th} percentile, then reject H_0 at $\alpha = 0.05$.

Example coded in R:




```
> scorefit <- lm(score ~ sweet + moist, data= taste)
> XtXinv <- summary(scorefit)$cov.unscaled
> X <- model.matrix(scorefit)
>  Ct <- matrix(c(0,1,0,  0,0,1),2,3, byrow=T)
> Ct                          ##contrast to test beta1=beta2=0
[1,]    0    1    0
[2,]    0    0    1
> middle <- X%*%XtXinv %*% t(Ct) %*% solve(Ct%*%XtXinv%*%t(Ct)) %*%
  Ct %*% XtXinv %*% t(X)
> fakeF <- function(e,i,middle = middle){
+  ##  e = residuals
+  ##  i = indices to use
+    em <- e[i]            ## em is the current resample from e
+    n <- length(e)
+    sSqd <- var(em)/(n-1)  * n
+    em %*% middle %*% em/sSqd  }
> bootFs <- rep(0,499)  ## set up a vector for storage.
> for(i in 1:499) bootFs[i] <- fakeF(resid(scorefit),sample(16,repl=TRUE))
  ### sample picks a random sample uses integers 1 to 16 with replacement
> summary(bootFs)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
 0.003475  0.410200  1.079000  1.693000  2.314000 11.920000
> Cb <- Ct%*% coef(scorefit)
> testF <- t(Cb)%*% solve(Ct%*%XtXinv%*%t(Ct)) %*% Cb /  summary(scorefit)$sigma^2
> testF
   258.2
# This is way bigger than the largest bootF, so reject H0: beta1=beta2=0
#  at alpha = 1/500.
##################     Should use modified residuals:
>  r <- MASS::rstudent(scorefit) * summary(scorefit)$sigma
> summary(r)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-5.66300 -1.86000  0.02752 -0.01581  1.67800  4.91000
>  summary(resid(scorefit))
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-4.40e+00 -1.76e+00  2.50e-02  1.39e-17  1.59e+00  4.20e+00
>  for(i in 1:499) bootFs[i] <- fakeF(r,sample(16,repl=TRUE))
>  summary(bootFs)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.001531 0.488300 1.203000 1.712000 2.448000 9.456000
  #############  again, none is close to our testF
```

Case–based Bootstrapping

Now we consider $(\boldsymbol{x}_i, y_i)$ as a random draw from a multivariate distribution. We'd like to sample from that same distribution, but instead have to resample n rows from the original rows, keeping the $\boldsymbol{x}_i$ and y_i together. Resample the **cases.** Then fit with `lm` to get $\widehat{\boldsymbol{\beta}}^*$.

Problem: Some resamples might represent only a fraction of the $\boldsymbol{X}$ space, making $\widehat{\boldsymbol{\beta}}^*$ unstable. For this type of sampling, we'll use the `boot` library.

```
> library(boot)
>  speed.fun <- function(data, i){
    coef(lm(I(sqrt(speed))~density + I(density^2),data=data[i,]))}
> speed.boot <- boot(speed, speed.fun, R=99)
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = speed, statistic = speed.fun, R = 99)
Bootstrap Statistics :
      original     bias    std. error
t1*  7.0026317 -1.327e-02   1.126e-01
t2* -0.0506910  2.119e-04   2.764e-03
t3*  0.0001486 -6.849e-07   1.586e-05

 ## t-ratios are
-0.0506910 /   2.764e-03 =  -18.34
and  0.0001486/ 1.586e-05 = 9.37  so we reject coefficients of 0

## Compare to:
> summary(speedfit)$coef
               Estimate Std. Error t value  Pr(>|t|)
(Intercept)   7.0026317  8.791e-02  79.656 0.000e+00
density      -0.0506910  2.721e-03 -18.631 1.532e-14
I(density^2)  0.0001486  1.732e-05   8.578 2.643e-08

> boot.ci(speed.boot, index=2)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 499 bootstrap replicates

Intervals :
Level      Normal              Basic
95%   (-0.0563, -0.0455 )   (-0.0564, -0.0451 )

Level     Percentile            BCa
95%   (-0.0563, -0.0450 )   (-0.0569, -0.0455 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

The five types are the first order normal approximation, the basic bootstrap interval, the studentized bootstrap interval, the bootstrap percentile interval, and the adjusted bootstrap percentile (BCa) interval. Take T as an estimate of θ, and to follow Davison and Hinkley (1997), let $B = R$ be the number of resamples created.

Normal Approximation Confidence Intervals

Work well if the resampling distribution is symmetric and not too long-tailed.

$$t - b_R \pm z_{1-\alpha} v^{1/2} \text{ where } v \text{ is a variance estimate}$$

The code in `library(boot)` includes the bias adjustment, b_R, and estimates v as the sample variance of the $\hat{\theta}^{*(b)}$. To check the normality assumption, a quantile-quantile plot

is recommended. For long-tailed distributions, a transformation may be applied to make the distribution more symmetric, but methods below work without transformation.

Basic Bootstrap Confidence Intervals

Use two percentiles of the empirical cdf as endpoints:

$$\hat{\theta}_\alpha = 2t - t^*_{((R+1)(1-\alpha))}, \quad \hat{\theta}_{1-\alpha} = 2t - t^*_{((R+1)\alpha)}$$

Studentized Bootstrap Confidence Intervals

Use normal approximation, but replace $Z = (T-\theta)/V^{1/2}$ with a bootstrap approximation, $z^* = (t^* - t)/v^{*1/2}$, which is computed for each bootstrap sample, and we work with its percentiles.

$$\hat{\theta}_\alpha = t - v^{1/2} z^*_{((R+1)(1-\alpha))}, \quad \hat{\theta}_{1-\alpha} = t - v^{1/2} z^*_{((R+1)\alpha)}$$

These intervals are preferred to the basic bootstrap intervals.

Bootstrap Percentile Confidence Intervals: $t^*_{((R+1)\alpha)}, \quad t^*_{((R+1)(1-\alpha))}$

1. For nicely distributed $\hat{\theta}_b$, these agree well with normal approximation. (Then all methods are OK.)

2. For skewed distributions, the percentile method is better than normal; it is equivariant under transformation.

3. The corrected Bootstrap is an improvement on this method.

BCa Intervals (Bias Corrected and adjusted)

Let $\hat{G}$ be the distribution function of T^*. Assume U^* is some transformed version of T^* with a normal distribution.

$$h(T) = U \sim N(\phi - w\sigma(\phi), \sigma^2(\phi)), \quad \text{with } \sigma(\phi) = 1 + a\phi.$$

We compute a CI for ϕ, then backtransform to an interval for θ.

$$\hat{\theta}_\alpha = \hat{G}^{-1}\left\{\Phi\left(w + \frac{w + z_\alpha}{1 - a(w + z_\alpha)}\right)\right\}$$

Writing the above in terms of simulated values, we get

$$\hat{\theta}_\alpha = t^*_{((R+1)\tilde{\alpha})}, \quad \tilde{\alpha} = \Phi\left(w + \frac{w + z_\alpha}{1 - a(w + z_\alpha)}\right)$$

Values of a and w are estimated using percentiles and score function of $\hat{G}$: $w = \Phi^{-1}[\hat{G}(t)]$ and $a = E^*[S^*(\hat{\theta})^3]/\{6\text{var}^*[S^*(\hat{\theta})^{3/2}]$. If $\tilde{\alpha}$ is too close to 0 or 1, then $(R+1)\tilde{\alpha}$ can exceed R or be less than 1.

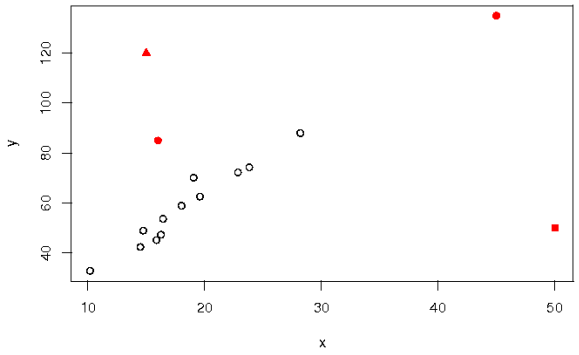
For more information, see Davison and Hinkley (1997)

3.4 Outliers and Influence

Types of Outliers in Regression:

Mark any outliers with these number codes:

1. Outliers in x
2. Outliers in y
3. Outliers in x and in y
4. Regression outliers – not unusual in x or in y, but do not fit the model.



Outliers in x are potentially influential points, outliers in y inflate estimatiors of σ, and regression outliers may indicate problems with the assumed model, either in the model for the mean or in the assumption of normality.

Outliers in General:

- Could be the most informative observations.
- Could be mistakes.
- Could throw off the analysis.

A Common Strategy for Dealing with Outliers:

- Check first to see if the outlier is a mistake.
- If no error is found, analyze two ways and let the reader of your report decide which is more believable.
  - with the questionable point using robust regression techniques.
  - without the point using least squares & normality

Alternatively, we could automatically switch to robust methods whenever we observe an outlier. However, robust methods have smaller power for detecting trends and do not give t and F test output (When you don't assume normality, you lose those nice distributions.) Outliers may (or may not) be influential points.

Influential Points are points which, if removed, change the estimate of $\boldsymbol{\beta}$ dramatically. There are several measures of influence, most relating to "leverage"

Leverage of the i^{th} point is h_{ii}, the i^{th} diagonal element of the hat matrix, $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}$. We can also express h_{ii} as $\boldsymbol{x}_i(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{x}_i^\mathsf{T}$ where $\boldsymbol{x}_i$ is the i^{th} row of $\boldsymbol{X}$. Think of the point $(\overline{\boldsymbol{x}}, \overline{y})$ as the fulcrum, then h_{ii} is large for points far from $\overline{\boldsymbol{x}}$, i.e. outliers in the $\boldsymbol{X}$ space. If the model does not include an intercept, the leverage increases with the distance from the origin.

We know that $tr(\boldsymbol{H}) =$ column rank$(\boldsymbol{X}) = k+1$, so $\sum h_{ii} = k+1$. If, as in a designed experiment, we could set our $\boldsymbol{x}$ values to be equally distance from $\overline{\boldsymbol{x}}$, then all points would

have equal leverage, $h_{ii} = (k+1)/n$. For observational studies, this is impossible, and large leverages indicate a problem. We will use the rule of thumb that points with leverage less than $2(k+1)/n$ are not exerting undue influence.

Centering does not change influence and outlier status.

Detecting Outliers

Regression outliers have large standardized residuals:

$$e_i^{(s)} = \frac{e_i}{s\sqrt{1-h_{ii}}}$$

or use studentized residuals (where (i) means without point i):

$$e_i^* = \frac{e_i}{s_{(i)}\sqrt{1-h_{ii}}} \text{ where } s^2_{(i)} = \frac{\sum_{l\neq i}[y_l - \boldsymbol{x}_l\widehat{\boldsymbol{\beta}}_{(i)}]^2}{n-k-2} \text{ and } \widehat{\boldsymbol{\beta}}_{(i)} = \left(\boldsymbol{X}_{(i)}^\mathsf{T}\boldsymbol{X}_{(i)}\right)^{-1}\boldsymbol{X}_{(i)}^\mathsf{T}y_{(i)}$$

Luckily, there is an easier way which does not involve computing n different regressions:

$$s^2_{(i)} = \frac{(n-k-1)s^2 - e_i^2(1-h_{ii})^{-1}}{n-k-2}$$

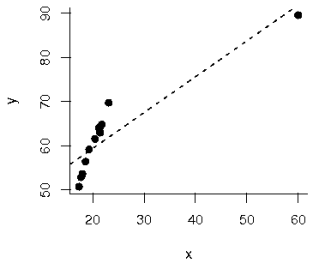
Special case for normality:

If $e_i \sim$ iid $N(0,\sigma^2)$ then $e_i^* \sim t_{n-k-2}$ which could be used to test for the inclusion of the i^{th} point. Be cautious about using this test, since it involves the huge assumptions of normality and "correct model", and using it on all cases is a multiple comparison problem.

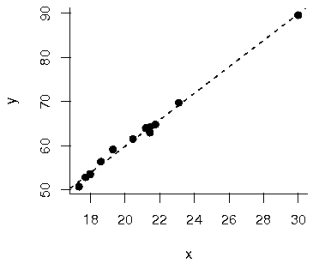
Measures of Influence

Why do we need measures?

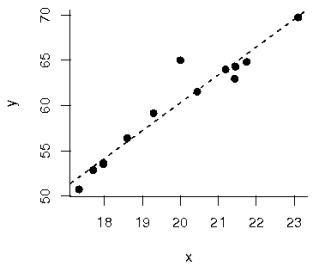
- We can have influential points which do not show up as outliers in the residuals.



- We can have an outlier in x and y which is not influential.



- We can have points with large $|e_i^*|$ which are not much to worry about.



1. DFBETA. How much does the jth coefficient, β_j, change when point i is removed? Note: this is an i by j matrix.

   Threshold: |DFBETA| > 1 in R

   $$\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}_{(i)} = \frac{(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{x}_i^\mathsf{T} e_i}{1 - h_{ii}}$$

   $$\text{standardized DFBETAS: } (\widehat{\beta}_j - \widehat{\beta}_{(i)j})/SE_{ij} = \frac{(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})_j^{-1}\boldsymbol{x}_i^\mathsf{T} e_i}{s_{(i)}(1-h_{ii})\sqrt{(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})_{jj}^{-1}}}$$

2. DFFIT. How much does the ith predicted value change when point i is removed?

   $$\widehat{y}_i - \widehat{y}_i(i) = \boldsymbol{x}_i^\mathsf{T}\widehat{\boldsymbol{\beta}} - \boldsymbol{x}_i^\mathsf{T}\widehat{\boldsymbol{\beta}}_{(i)} = \frac{h_{ii}e_i}{1-h_{ii}} \qquad \text{standardized DFFITS} = \frac{\sqrt{h_{ii}}e_i}{s_{(i)}(1-h_{ii})} = e_i^*\sqrt{\frac{h_{ii}}{1-h_{ii}}}$$

   Threshold in R: $> 3\sqrt{\frac{k+1}{n}}$

3. Covariance Ratio. Does the estimated variance-covariance matrix of $\widehat{\boldsymbol{\beta}}$ change when we delete the ith point?

   $$\frac{\det[s_{(i)}^2(\boldsymbol{X}_{(i)}^\mathsf{T}\boldsymbol{X}_{(i)})^{-1}]}{\det[s^2(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}]} = \frac{1}{1-h_{ii}}\left[\frac{s_{(i)}^2}{s^2}\right]^{k+1} = \frac{(n-k-1)^{k+1}}{(n-k-2+e_i^{*2})^{k+1}(1-h_{ii})}$$

   Threshold in R: $< 1 - 3\sqrt{(k+1)/(n-k-1)}$ or $> 1 + 3\sqrt{(k+1)/(n-k-1)}$

4. PRESS, (PREdictive Sum of Squares) residuals.

   $$e_{i,-1} = y_i - \widehat{y}_i(i) = \frac{e_i}{1-h_{ii}}$$

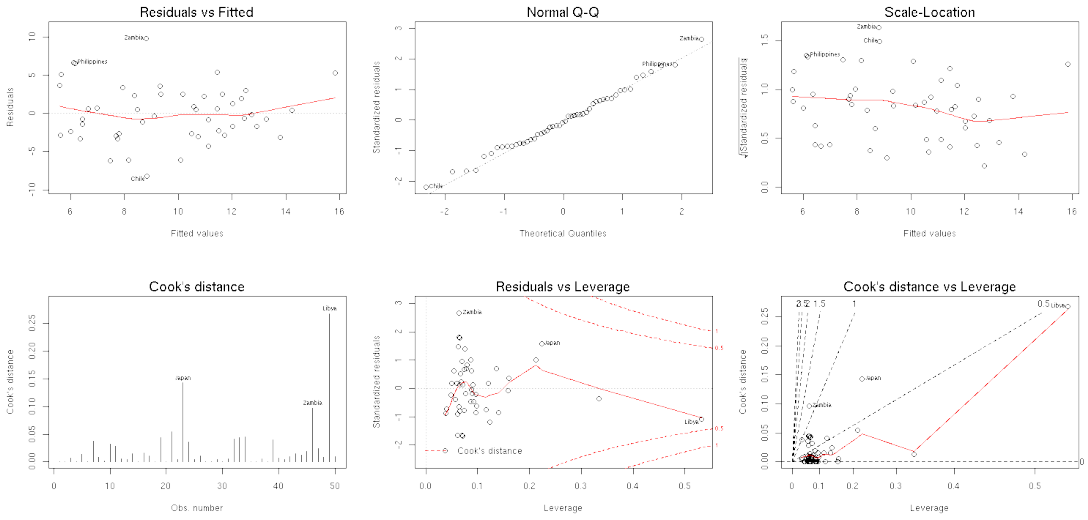
5. Cook's Distance (similar to DFFITS^2)

   $$D_i = \frac{(\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}(i))^\mathsf{T}\boldsymbol{X}^\mathsf{T}\boldsymbol{X}(\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}(i))}{(k+1)s^2} = \frac{e_i^2 h_{ii}}{s^2(k+1)(1-h_{ii})^2}$$

   Threshold in R: if $P(F \le D_i) > .5$ where $F \sim F(k+1, n-k-1)$.

```
> ?LifeCycleSavings
  Data on the savings ratio 1960-1970 from Belsley, Kuh and Welsch.
  A data frame with 50 observations (countries) on 5 variables.
       [,1]  sr     numeric  aggregate personal savings
       [,2]  pop15  numeric  % of population under 15
       [,3]  pop75  numeric  % of population over 75
       [,4]  dpi    numeric  real per-capita disposable income
       [,5]  ddpi   numeric  % growth rate of dpi
> data(LifeCycleSavings)
> lm.SR <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
> inflm.SR <- influence.measures(lm.SR)  ## builds dfbetas, dffits,
                            ## cov.ratio, cook's D, and hat diagonals
> summary(inflm.SR)
Potentially influential observations of
        lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings):

              dfb.1_ dfb.pp15 dfb.pp75 dfb.dpi dfb.ddpi dffit  cov.r  cook.d  hat
Chile         -0.20   0.13     0.22    -0.02    0.12    -0.46   0.65_* 0.04   0.04
UnitedStates   0.07  -0.07     0.04    -0.23   -0.03    -0.25   1.66_* 0.01   0.33_*
Zambia         0.16  -0.08    -0.34     0.09    0.23     0.75   0.51_* 0.10   0.06
Libya          0.55  -0.48    -0.38    -0.02   -1.02_*  -1.16_* 2.09_* 0.27   0.53_*
> dfbs <- dfbetas(lm.SR)
> dfbs[apply(dfbs,1,function(x) max(abs(x))> 1),]
           (Intercept)    pop15    pop75      dpi     ddpi
Libya           0.5507  -0.4832  -0.3797  -0.0194  -1.0245*
 ## Lots have dfbeta > .2828 when unstandardized
```

Note: This is not a very good model for these data, because 'dpi' does not add significantly to the model, and 'pop15' is highly correlated with 'pop75'. However, it does illustrate use of influence measures.



In SAS:

```
data savings;
  infile "data/savings" firstobs=2;
  input country$ SR  pop15    pop75      dpi     ddpi ;
proc reg;
  model SR =  pop15    pop75      dpi     ddpi / influence;
run;                              Output Statistics
                                          Hat Diag         Cov
        Obs     Residual     RStudent           H       Ratio      DFFITS
          1       0.8636       0.2327      0.0677      1.1928      0.0627
          2       0.6164       0.1710      0.1204      1.2678      0.0632
          3       2.2190       0.6066      0.0875      1.1762      0.1878
Chile     7      -8.2422      -2.3134      0.0373      0.6547     -0.4554
US       44      -1.1116      -0.3546      0.3337      1.6555     -0.2510
Z        46       9.7509       2.8536      0.0643      0.5116      0.7482
L        49      -2.8295      -1.0893      0.5315      2.0906     -1.1601


                ------------------------DFBETAS-------------------------
        Obs     Intercept       pop15       pop75         dpi        ddpi
          1        0.0123     -0.0104     -0.0265      0.0453     -0.0002
          2       -0.0101      0.0059      0.0408     -0.0367     -0.0082
          3       -0.0642      0.0515      0.1207     -0.0347     -0.0073
Chile     7       -0.1994      0.1327      0.2198     -0.0200      0.1200
US       44        0.0691     -0.0729      0.0375     -0.2331     -0.0327
Zambia   46        0.1636     -0.0792     -0.3390      0.0941      0.2282
Libya    49        0.5507     -0.4832     -0.3797     -0.0194     -1.0245
```

Other model options: `R` to get Cook's Distance, `PRESS` to get Press residuals.

3.5 Robust and Resistant Regression

Test for Normality

To obtain exact t and F tests, we have assumed normality of residuals, although we know that data is never "exactly" normally distributed. Some practitioners believe in testing the assumption of normality, for instance with the Shapiro-Wilks or Kolmogorov test applied to residuals. I am not a believer in this test because exact normality of residuals is not really necessary. Also, the tests mentioned have low power (too often fail to reject when normality is absent) for small sample sizes, and for large sample sizes they too often find small departures from normality. The F tests are robust to moderate departures from normality because the coefficient estimates are essentially means, and the CLT provides approximate normality for $\widetilde{\boldsymbol{\beta}}$ even when $\boldsymbol{\epsilon}$ is not Gaussian (see 3.3.1). I do believe in plotting the quantile-quantile plot to look at the distribution of residuals versus normality. This plot is supposed to show a "straight line" when the data are Gaussian, but you should run a few test plots using randomly generated normals to see how close to or far from straight the QQ plots can be.

```
par(mfrow=c(4,4))
qqnorm(rstudent(my.model))  ## qq plot of interest, say with 20 observations
for (i in 1:15)
  qqnorm(rnorm(20))      ## 15 reference plots, each with 20 random normals
```

There are several flavors of residuals, and we have seen that studentized residuals all have the same t_{n-r} distribution and provide a test for an outlier.

$$e_i^* = \frac{e_i}{s_{(i)}\sqrt{1-h_{ii}}}$$

It is best to use studentized residuals for comparing to normality.

When evaluating leverages, a rule of thumb is that the largest h_{ii} should be less than $2 \times \text{rank}(\boldsymbol{X})/n = 2r/n$, another uses .2 as the cutoff.

If, despite my advice, you still want to do tests of normality on residuals, see the `shapiro.test` function in R. The more general Kolmogorov-Smirnoff test comparing any two distributions is contained in the `ks.test` function. Consider yourself warned: for small data sets the test is not very powerful, and for large ones, you may be rejecting due to small deviations from normality.

Resistance to Outliers

Imagine changing one point at a time in a dataset, replacing a "real" observation with a draw from a different distribution – an outlier – and continuing to change more and more points. Estimators vary in the amount of degradation they can tolerate, and the smallest proportion of contaminated data which will make the estimator arbitrarily large is called the "breakdown point" of the estimator. For example, in estimating the center of a distribution, the mean has breakdown of 0 (if we change any positive fraction of the data, the mean is affected), a 10% trimmed mean has breakdown of 0.10, and the median has breakdown of 0.50.

Efficiency is another important property for estimators. It answers the question, "How well does this method perform relative to BLUE (on normally distributed data) as sample size approaches infinity?", and has values in (0,1), with larger being better.

In using resistant methods, we are trading high efficiency for high breakdown.

Robust Regression

The term "robust" means we are concerned with deviation from the normality assumption toward long-tailed distributions. We will consider robust and resistant regression methods as being similar.

Generalizing the Criterion for "Best" Coefficients

We've been using the Least Squares Criterion which minimizes

$$Q(\boldsymbol{\beta}) = \sum e_i^2$$

However, there are other functions of the residuals which we could minimize, for instance $\sum |e_i|$, which would give less emphasis to outliers. The absolute value is the L_1 norm in mathematics, in stat we speak of least absolute residuals to contrast with least squares. To generalize, write the log-likelihood as

$$f_i(y_i;\ \boldsymbol{\beta}, \sigma) = \frac{1}{\sigma} f(\frac{y_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta}}{\sigma})$$

where σ is a scale parameter. Assuming independence, the log-likelihood of the data is

$$l(\boldsymbol{\beta}, \sigma) = -n\log(\sigma) + \sum \log f[(y_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{\beta})/\sigma]$$

Differentiating leads to the estimating equations

$$\sum_{i=1}^{n} \rho'[e_i(\boldsymbol{b})/s]\boldsymbol{x}_i = \boldsymbol{0}, \quad \sum_{i=1}^{n} \rho'[e_i(\boldsymbol{b})/s]e_i(\boldsymbol{b}) = ns$$

where $\rho = -\log f$. Thinking of any possible function, ρ, the estimators which minimize such functions are termed M-estimators, because they Maximize likelihood which is assumed to be $e^{\rho(t)}$. Choosing a bounded function for ρ' limits the influence of outliers. We

typically want ρ to be symmetric about the y axis and continuous. One bounded function suggested by Huber (*Robust Statistics*, 1981, Wiley) is $\rho'(t) = \psi_c(t) = \min(1, c/|t|)t$. If $|t| < c$ it returns t, otherwise it returns $-c$ for $t < -c$ or c for $t > c$. A point with very large residual has no more influence than one with residual of c. As $c \to \infty$, we get OLS, and as $c \to 0$, we get least absolute residual (LAR) estimators.

Robust estimation can be viewed as weighted regression where outliers get less weight. It requires iterations between:

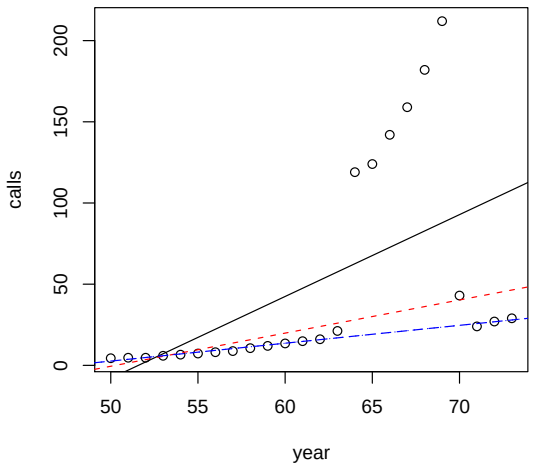
1. Finding residuals based on some coefficient and scale estimates, then computing weights,

2. Fitting a model using weights and estimating coefficients and scale.

Here is an example using the R function `rlm` for robust regression.

```
> library(MASS) ## in 1964-1969 a different data collection method was used
> data(phones)
> phones = as.data.frame(phones)
>  plot(phones)
>  abline(phones.lmfit <- lm(calls~year,phones))
>  abline(phones.Mestfit <- rlm(calls~year,phones, maxit=50), lty=2,col=2)
>  phones.MMfit <- rlm(calls~year,phones, method="MM", init = "lts", maxit=50)
```

```
>  abline(phones.MMfit, lty=3,col=4)
> coef(phones.lmfit)
(Intercept)        year
-260.059246    5.041478
> summary(phones.lmfit)$sigma
[1] 56.22339
> coef(phones.Mestfit)
(Intercept)        year
-102.622198    2.041350
> summary(phones.Mestfit)$sigma
[1] 9.03168
> coef(phones.MMfit)
(Intercept)        year
 -52.423017    1.100947
> summary(phones.MMfit)$sigma
[1] 2.128546
```



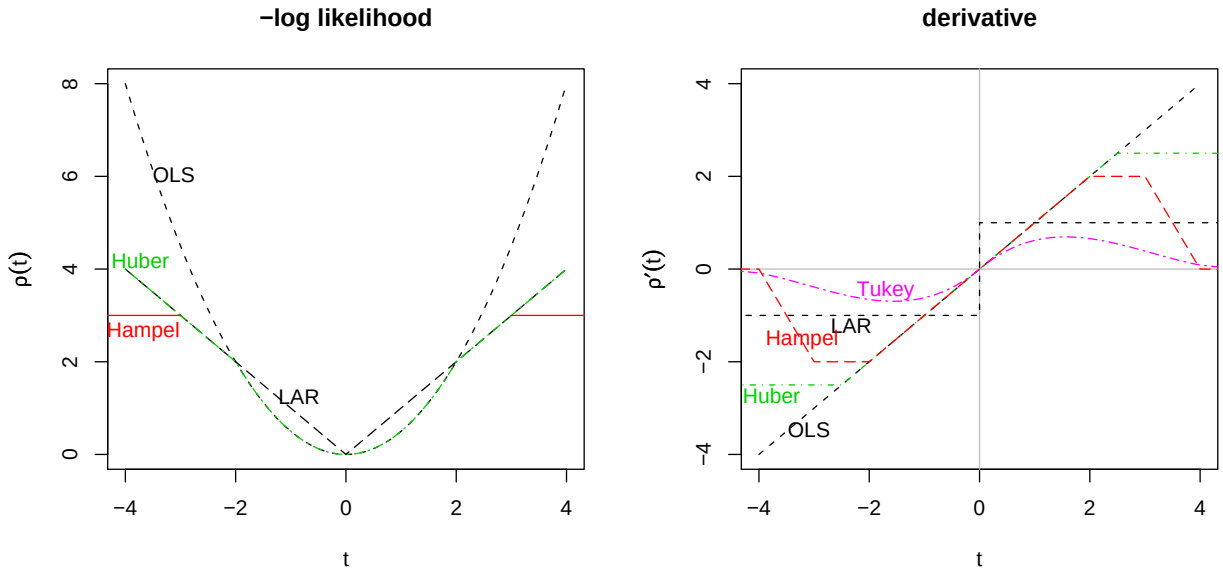

```
> phones.LTSfit <- MASS::lqs(calls~year,phones, method= "lts")
> coef( phones.LTSfit)
(Intercept)        year
 -56.162443    1.158824
>  phones.LTSfit$scale
[1] 1.248597 1.130633     ## second is more robust
```

Problem with Huber's method: it's robust, but not resistant. For resistance to outliers we need ρ which is redescending instead of convex. Examples: Tukey's bisquare: $\psi(t) = t[\max(1 - |t/R|, 0)]^2$ where $R = 4.685$ gives 95% efficiency for normal data. Hampel's

piecewise function:

$$\psi(x) = \text{sign}(x) \begin{cases} |x| & 0 < |x| < a \\ a & a < |x| < b \\ a(c-x)/(c-b) & b < |x| < c \\ 0 & c < |x| \end{cases}$$

Note that a, b, and c are needed.



If we use redescending functions to weed out outliers, there is no guarantee that the iterations will converge; there can be several solutions to the "normal" equations, so a good starting point is needed.

LMS estimation, least median of squares estimation minimizes

$$\min_b \text{median}_i |y_i - \boldsymbol{x}_i\boldsymbol{\beta}|^2$$

This is very resistant, but very inefficient. Needs no scale estimate.

LTS estimation, least trimmed squares estimation minimizes

$$\min_b \sum_i^q |y_i - \boldsymbol{x}_i\boldsymbol{\beta}|^2_{(i)}$$

Where the sum is over the smallest $q = \lfloor (n+p+1) \rfloor / 2$ squared residuals.

***S*-estimation** find the solution to

$$\sum_{i=1}^n \chi\left(\frac{y_i - \boldsymbol{x}_i\boldsymbol{\beta}}{c_0 s}\right) = (n-p)\gamma$$

with smallest scale, s. Use $\chi(u) = u^6 - 3u^4 + 3u^2$, u if ≤ 1 or 1, otherwise. Choice of $c_0 = 1.548$ and $\gamma = 0.5$ gives efficiency of 29%, which is better than LMS or LTS.

For all three of the above methods, there is no analytic solution, we must use a search algorithm to find an approximate solution. If you have lots of computer power and time,

Venables and Ripley recommend the MM-estimator which begins with the S-estimate of the coefficients, and also estimates scale with S-estimation.

SAS Procedures

New with SAS Version 9.2 is Proc RobustReg. Here is the "Overview" from their documentation:

> Many methods have been developed in response to these problems. However, in statistical applications of outlier detection and robust regression, the methods most commonly used today are Huber M estimation, high breakdown value estimation, and combinations of these two methods. The new ROBUSTREG procedure in this version provides four such methods: M estimation, LTS estimation, S estimation, and MM estimation.
>
> 1. M estimation was introduced by Huber (1973), and it is the simplest approach both computationally and theoretically. Although it is not robust with respect to leverage points, it is still used extensively in analyzing data for which it can be assumed that the contamination is mainly in the response direction.
> 2. Least Trimmed Squares (LTS) estimation is a high breakdown value method introduced by Rousseeuw (1984). The breakdown value is a measure of the proportion of contamination that an estimation method can withstand and still maintain its robustness. The performance of this method was improved by the FAST-LTS algorithm of Rousseeuw and Van Driessen (1998).
> 3. S estimation is a high breakdown value method introduced by Rousseeuw and Yohai (1984). With the same breakdown value, it has a higher statistical efficiency than LTS estimation.
> 4. MM estimation, introduced by Yohai (1987), combines high breakdown value estimation and M estimation. It has both the high breakdown property and a higher statistical efficiency than S estimation.

Here is output from Proc RobustReg using the phones data.

```
proc robustreg;
  model calls = year;
run;
proc robustreg method = MM;
  model calls = year;
run;
"M estimation"
                      Standard   95% Confidence     Chi-
Parameter DF Estimate    Error       Limits       Square Pr > ChiSq

Intercept  1 -52.3025   2.7476 -57.6878 -46.9173  362.35     <.0001
     year  1   1.0980   0.0444   1.0110   1.1851  611.70     <.0001
    Scale  1   1.6555

"MM estimation"
                      Standard   95% Confidence     Chi-
Parameter DF Estimate    Error       Limits       Square Pr > ChiSq
```

```
Intercept  1 -162.881  91.4682 -342.156  16.3933    3.17     0.0750
year       1   3.2221   1.5030   0.2762   6.1679    4.60     0.0321
Scale      0  54.2183
```

I don't know why the "M" results look like R's "MM" output and vice versa.

4 Inference

4.1 Likelihood Ratio Tests

Consider a linear model and a simpler model nested inside it. Typically we wonder if some coefficients might be zero, meaning that those predictors are not needed in our model. The simple model matrix can be represented as $\boldsymbol{X}_1$, and the complex model as $\boldsymbol{X} = [\boldsymbol{X}_1\ \boldsymbol{X}_2]$ in:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} = \boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\epsilon};\ \ \boldsymbol{\epsilon} \sim (0, \sigma^2\boldsymbol{V})$$

Assuming normality, twice the log likelihood is

$$2l(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \boldsymbol{V}) = -n\log(2\pi\sigma^2) - \log(|V|) - (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\mathsf{T}\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})/\sigma^2$$

Log likelihood is maximized at $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{y}$ and $\widehat{\sigma}^2 = (\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}})^\mathsf{T}\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}})/n$.

The log of the likelihood ratio for comparing the nested models is the difference in log likelihoods evaluated at the MLEs. Twice log likelihood evaluated at $(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2)$ is $-n\log(2\pi) - n\log(\widehat{\sigma}^2) - \log(|\boldsymbol{V}|) - n\widehat{\sigma}^2/\widehat{\sigma}^2 = k - n\log(\widehat{\sigma}^2)$ where k is a constant depending only on n and $\boldsymbol{V}$. We also need to evaluate likelihood at the LME restricted to $\boldsymbol{\beta}$ of the form $\boldsymbol{\beta}^\mathsf{T} = (\boldsymbol{\beta}_1^\mathsf{T}\ \boldsymbol{0}^\mathsf{T})^\mathsf{T}$ meaning that the coefficients associated with $\boldsymbol{X}_2$ are all 0. Under this H_0, the MLE will be called $(\widehat{\boldsymbol{\beta}}_0, \widehat{\sigma}_0^2)$ and twice log likelihood evaluated at this value is $k - n\log(\widehat{\sigma}_0^2)$. The difference in log likelihoods is then $-\frac{n}{2}\log(\widehat{\sigma}^2/\widehat{\sigma}_0^2)$. Apply monotonic transformations to this ratio to get a quantity with a known distribution, and we obtain the LRT test statistic:

$$F = \frac{(\widehat{\sigma}_0^2 - \widehat{\sigma}^2)/(r - r_1)}{\widehat{\sigma}^2/(n - r)}$$

which has a central $F_{r-r_1,n-r}$ distribution under H_0, and a non-central F otherwise. We have skipped some hard work here. To obtain an F distribution we have to show that the numerator and denominator are independent, and that each is a χ^2 RV divided by its degrees of freedom. I'll leave that for a higher level course.

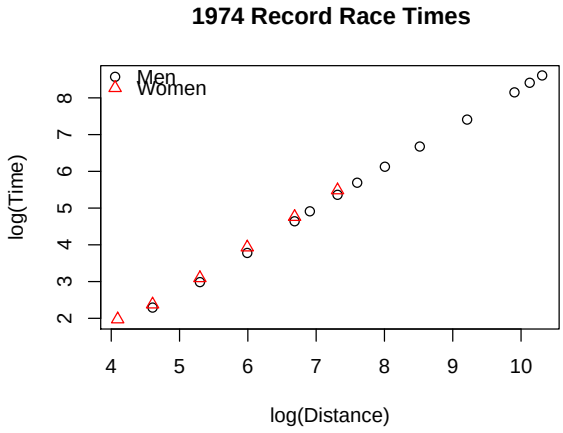
Example:
Sen and Srivastava (1997) in example 3.2 present data on the relationship between length of races (log distance) and length of record (log) times for men and women. Data are nicely linear, and questions center on similarities/differences in the line for men versus the line for women. Model:

$$\boldsymbol{y} = \beta_0\boldsymbol{1} + \beta_1\boldsymbol{x}_1 + \beta_2\boldsymbol{x}_2 + \beta_3\boldsymbol{x}_1 \cdot \boldsymbol{x}_2 + \boldsymbol{\epsilon}$$

where x_1 is log distance, x_2 is an indicator for women, $\boldsymbol{x}_1 \cdot \boldsymbol{x}_2$ is element-wise multiplication, and y is log time (sec). We will test to see if we can simultaneously remove $(\beta_2,\ \beta_3)$. S code:

```
> races <- rbind(read.table("data/e3.4",head=T), read.table("data/e3.5",head=T))
> races$ltime <- log(races$Time)
> races$ldist <- log(races$Dist)
> races$gender <- rep(0:1, c(13,6))
> race.fit1 <- lm(ltime ~ ldist, data = races)
> race.fit2 <- lm(ltime ~ ldist + gender + ldist:gender, data = races)
```

```
> anova(race.fit1, race.fit2)
Analysis of Variance Table

Model 1: ltime ~ ldist
Model 2: ltime ~ ldist + gender + ldist:gender
  Res.Df      RSS Df Sum of Sq      F   Pr(>F)
1     17 0.106458
2     15 0.057245  2  0.049213 6.4478 0.009532
> ((0.106458 -  0.057245)/  2) /( 0.057245 /15)
[1] 6.447681
> 1-pf( 6.447681, 2, 15)
[1] 0.009532228
```



The `anova` command in R applied to two or more linear models assumes that they are nested, that the simpler models come first, and that you want to test to see if the terms added explain a significant additional amount of the variation in y beyond the amount explained by the initial model. Its last line of output is the LRT. The computations after the `anova` command demonstrate how the Δ_{SSE}, F and p-value are derived.

To do this using SAS, find the relevant sums of squares and construct the F statistic by hand.

```
 options ls=76 ps=66;
 dm "log;clear;out;clear;";
 data men;
   infile "stat506/data/e3.4" firstobs=2;
    input distance time;
   ldist=log(distance);
   ltime=log(time);
   gender="M";
 data women;
    infile "stat506/data/e3.5" firstobs=2;
    input distance time;
   ldist=log(distance);
   ltime=log(time);
   gender="F";
 data racetime;
   set men women;
   drop distance time;
    if gender="M" then gend=0;
      else gend = 1;
    Dgend=gend*ldist;
 * proc print;
run;
 proc reg;
      model ltime = ldist;
      model ltime = ldist gend Dgend;
 run;                                  ******  Output ****;
Model 1                             Sum of          Mean
 Source                   DF       Squares        Square   F Value   Pr > F
 Model                     1      78.92660      78.92660   12603.6   <.0001
 Error                    17       0.10646       0.00626
 Corrected Total          18      79.03306
```

```
Model 2                          Sum of          Mean
 Source                   DF     Squares        Square   F Value   Pr > F
 Model                     3    78.97582      26.32527   6898.11   <.0001
 Error                    15     0.05724       0.00382
 Corrected Total          18    79.03306
```

You could subtract the SSE's (0.10646 - 0.05724) , divide by the difference in df = $3 - 1 = 2$, and then divide by the MSE of the full model, 0.00382 to get F = 6.448 on 2, 15 df. However, SAS does offer several ways to obtain the test we need. If we add this line after the second model statement:

```
mtest gend, Dgend;
```

Then SAS will test the simultaneous hypothesis that both coefficients are zero and give this result:

```
              Multivariate Statistics and Exact F Statistics
                           S=1    M=0    N=6.5

Statistic                   Value          F      Num DF    Den DF  Pr > F

Wilks' Lambda            0.53772101     6.4478         2        15  0.0095
Pillai's Trace           0.46227899     6.4478         2        15  0.0095
Hotelling-Lawley Trace   0.85970045     6.4478         2        15  0.0095
```

Next we will demonstrate that this really is the same test as the LRT.

Aside: Notice that SAS and R both output another F test. It is also a LRT, but (if the intercept is included in the model formula) uses the simpler model $\boldsymbol{y} = \mu\mathbf{1} + \boldsymbol{\epsilon}$ which just says that there is no relationship between response and predictor, the best thing to use for estimation is the overall mean, μ. It answers the question, "Are **any** of the predictors 'significant' when building a linear model?"

4.2 LRT as a Test of Contrasts

In general, LRT compares the likelihood of the data evaluated at the MLE found by searching the entire parameter space to the likelihood of the data evaluated at some MLE_0 a subspace of the entire parameter space. Most commonly, the entire parameter space is the space of all possible $\boldsymbol{\beta}$ and the simpler model removes some predictors and forces us to maximize likelihood over the smaller subspace where $\boldsymbol{\beta}_2 = \mathbf{0}$.

More generally, we could set up restrictions on $\boldsymbol{\beta}$ is some other way, as in a test of contrasts. This will also restrict the parameter space available under H_0.

For the race example, consider the `gender` and `ldist-by-gender` coefficients (β_2 and β_3). We want to test to see if they might both be zero. It terms of a contrast, we are testing

$$H_0 : \ C^\mathsf{T}\boldsymbol{\beta} = \boldsymbol{\delta}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ versus } H_a : \ C^\mathsf{T}\boldsymbol{\beta} \neq \boldsymbol{\delta}_0 \ \text{ where } C^\mathsf{T} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If $\text{rank}(C) = m$ ($= 2$ in our example), then the test statistic is

$$F^* = \frac{(C^\mathsf{T}\hat{\boldsymbol{\beta}} - \boldsymbol{\delta}_0)^\mathsf{T}[C^\mathsf{T}(X^\mathsf{T}\boldsymbol{V}^{-1}X)^{-1}C]^{-1}(C^\mathsf{T}\hat{\boldsymbol{\beta}} - \boldsymbol{\delta}_0)/m}{MSE}$$

and has a $F(m, n-r, \lambda)$ distribution, where $\lambda = \frac{(C^\mathsf{T}\beta-\delta_0)^\mathsf{T}[C^\mathsf{T}(X^\mathsf{T}\boldsymbol{V}^{-1}X)^{-1}C]^{-1}(C^\mathsf{T}\beta-\delta_0)/m}{2\sigma^2}$ and $\lambda = 0$ iff H_0 is true. To find these values using R, start with the summary of the full model.

```
> summary(race.fit2)
Residuals:
      Min        1Q    Median        3Q       Max
-0.102266 -0.038941 -0.006206  0.046545  0.114664

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.8231958  0.0768074 -36.757 4.44e-16
ldist         1.1122144  0.0096874 114.810  < 2e-16
gender        0.1310339  0.1505799   0.870    0.398
ldist:gender -0.0005397  0.0244299  -0.022    0.983

Residual standard error: 0.06178 on 15 degrees of freedom
Multiple R-Squared: 0.9993,     Adjusted R-squared: 0.9991
F-statistic:  6898 on 3 and 15 DF,  p-value:     0
```

Next, note that $s^2(X^\mathsf{T}\boldsymbol{V}^{-1}X)^{-1}$ is obtained by application of the `vcov` function to the linear model object.

```
> vcov(race.fit2)
               (Intercept)         ldist        gender  ldist:gender
(Intercept)   0.0058993829 -7.253156e-04 -0.0058993829  7.253156e-04
ldist        -0.0007253156  9.384582e-05  0.0007253156 -9.384582e-05
gender       -0.0058993829  7.253156e-04  0.0226743137 -3.574432e-03
ldist:gender  0.0007253156 -9.384582e-05 -0.0035744317  5.968214e-04

> diag(vcov(race.fit2))^.5
 (Intercept)        ldist       gender ldist:gender
 0.076807345  0.009687393  0.150579751  0.024429901
  ## diagonal of inv(X\tr X) * sigma gives the standard errors of beta-hat
```

We need to construct the contrast matrix.

```
> Cntrst <- rbind(c(0,0,1,0),c(0,0,0,1))
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    0
[2,]    0    0    0    1
```

Then just do the **matrix** arithmetic (matrix multiply in R used `%*%`):

```
> betahat <- coef(race.fit2)
> XtXinv <- summary(race.fit2)$cov.unscaled
> Fstar <- t(Cntrst %*% betahat) %*% solve( Cntrst %*% XtXinv %*% t(Cntrst)) %*%
              (Cntrst %*% betahat)/2 / summary(race.fit2)$sigma^2
> Fstar
[1,] 6.447748     ### Same as in the anova
```

Because the reference distribution is $F(2, 15)$ as with the anova, we get the same p-value as we got before, 0.00953. We have not proven a theorem here, but have demonstrated that there is a test of contrast which gives the same results as a LRT for nested models. That is generally the case as long as the full model is of full rank. (R does not build any less-than-full-rank models, at least not unless you really work at it.)

An equivalent test using a different model matrix.
In Sen and Srivastava (1997), p 69 to 70, they do the same two degrees of freedom test, but they set up the model differently with an intercept for men, ln(*distance*) for men, intercept for women, and ln(*distance*) for women. The contrast then tests to see if the intercepts **and** slopes are equal. The $\boldsymbol{X}$ matrix is the last four columns below.

```
> races
    Dist   Time    ltime     ldist gender M.Int       M.x F.Int      F.x
1    100    9.9 2.292535  4.605170      0     1  4.605170     0 0.000000
2    200   19.8 2.985682  5.298317      0     1  5.298317     0 0.000000
3    400   43.8 3.779634  5.991465      0     1  5.991465     0 0.000000
4    800  103.7 4.641502  6.684612      0     1  6.684612     0 0.000000
5   1000  136.0 4.912655  6.907755      0     1  6.907755     0 0.000000
6   1500  213.1 5.361762  7.313220      0     1  7.313220     0 0.000000
7   2000  296.2 5.691035  7.600902      0     1  7.600902     0 0.000000

         ...  Some lines omitted ...
    Dist   Time    ltime     ldist gender M.Int       M.x F.Int      F.x
14    60    7.2 1.974081  4.094345      1     0  0.000000     1 4.094345
15   100   10.8 2.379546  4.605170      1     0  0.000000     1 4.605170
16   200   22.1 3.095578  5.298317      1     0  0.000000     1 5.298317
17   400   51.0 3.931826  5.991465      1     0  0.000000     1 5.991465
18   800  117.0 4.762174  6.684612      1     0  0.000000     1 6.684612
19  1500  241.4 5.486455  7.313220      1     0  0.000000     1 7.313220
> race.fit3 <- lm(ltime ~ M.Int + M.x + F.Int + F.x - 1, data = races)
> summary(race.fit3)
Residuals:
      Min        1Q    Median        3Q       Max
-0.102265 -0.038941 -0.006206  0.046544  0.114663

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
M.Int -2.823195   0.076807  -36.76 4.44e-16
M.x    1.112214   0.009687  114.81  < 2e-16
F.Int -2.692162   0.129518  -20.79 1.80e-12
F.x    1.111675   0.022427   49.57  < 2e-16

Residual standard error: 0.06178 on 15 degrees of freedom
Multiple R-Squared: 0.9999,     Adjusted R-squared: 0.9999
F-statistic: 3.74e+04 on 4 and 15 DF,  p-value:     0
```

Notice that this model does not have an overall intercept. The default F test uses the null model $\boldsymbol{y} = \boldsymbol{0} + \boldsymbol{\epsilon}$, which is rather ridiculous. The third model fits just as well as `race.fit2`, since the residuals and MSE are identical, they are just different parameterizations. None of the (conditional) individual t-tests for model 3 give useful information, since they do not compare the genders. In effect the 3rd model fits independent regressions to the two groups with a common estimate of σ. To test for equality of slope and intercept, our contrast matrix is now

```
> Cntrst <- rbind(c(1,0,-1,0),c(0,1,0,-1))
[1,]    1    0   -1    0
[2,]    0    1    0   -1
```

And the same computations on this model give the same F^*.

```
> betahat <- coef(race.fit3)
>  XtXinv <- summary(race.fit3)$cov.unscaled
>  Fstar <- t(Cntrst %*% betahat) %*% solve( Cntrst %*% XtXinv %*% t(Cntrst))
```

```
          %*% (Cntrst %*% betahat)/2 / summary(race.fit3)$sigma^2
>  Fstar
[1,] 6.447748
```

You **do** need to understand how models are specified in R and in SAS and to construct your own contrasts. Naturally, in the 3000 packages available, there are some which help with contrasts. In particular, the `gmodels` package by Greg Warnes has a `make.contrasts` function to convert "human-readable" contrasts to a form R can handle, and a `fit.contrasts` function to give CI's and tests. Also, in the `nlme` package one can fit a model with `gls` or `lme` and then test a (fixed effects) contrast within a call to the `anova` function.

The same test in SAS with Proc Reg:

```
data men2;
    set men;       maleInt =1;  maleDist= ldist;
    femInt=0;       femDist=0;
    drop gender distance time ldist ;
data women2;
   set women;      maleInt =0;    maleDist= 0;
   femInt=1;      femDist=ldist;
   drop gender distance time ldist;
data racetime;
   set men2 women2;
proc print;
proc reg;
   model  ltime= maleInt femInt maleDist femDist /noint;
   mtest maleInt-femInt, maleDist-femDist;
run;
                                              *******  Output  *****;
                          Parameter Estimates
                    Parameter      Standard    T for H0:
 Variable  DF        Estimate         Error   Parameter=0    Prob > |T|
 MALEINT    1     -2.823196    0.07680744       -36.757        0.0001
 FEMINT     1     -2.692162    0.12951807       -20.786        0.0001
 MALEDIST   1      1.112214    0.00968741       114.810        0.0001
 FEMDIST    1      1.111675    0.02242712        49.568        0.0001

Statistic                    Value          F      Num DF    Den DF  Pr > F
Wilks' Lambda             0.53772101     6.4478         2        15  0.0095
```

One can also use PROC GLM to define and test a two-dimensional contrast:

```
 proc glm;
   model ltime= maleInt femInt maleDist femDist /noint;
   contrast 'One line versus 2'
     maleInt -1 femInt 1 ,
     maleDist -1 femDist 1 ;
run;

Contrast                DF    Contrast SS   Mean Square   F Value     Pr > F
One line versus 2        2     0.04921328    0.02460664      6.45     0.0095
```

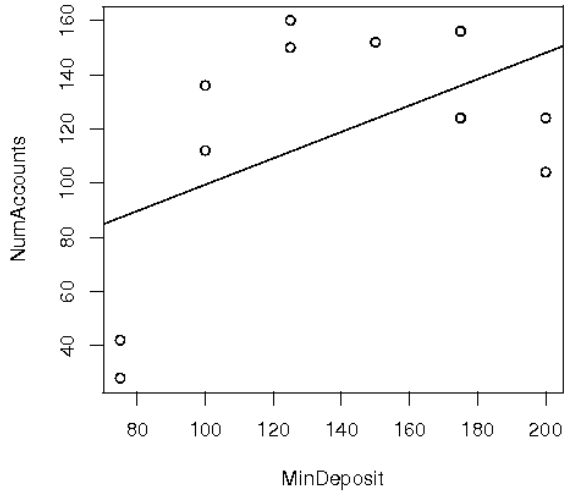
4.3 Pure Error and Nested Models, a Special LRT

The Problem

Suppose an experiment has been performed using treatments which are an ordered factor, equally spaced in time, temperature, or any controllable variable. We then have a choice between using polynomial regression and using ANOVA. If a low order polynomial describes the data well, then it gives a nicer representation, since it allows us to interpolate between the settings chosen for x. See Monahan (2008) Section 7.5.

Example From Kutner et al. (2004) §3.7

A bank has 12 suburban branches. They do an experiment to determine the relationship between "minimum balance" required to open an account and the "number of new accounts", setting minimums to be \$75, 100, 125, 150, 175, or 200, using each treatment at 2 randomly selected branches. Those opening a new account also get a gift which has value directly proportional to the size of the minimum deposit, so x is really controlling two things: minimum deposit and gift value. We might try to fit a simple linear regression.




```
> plot(banking)
> bank.fit1 <- lm(NumAccounts ~ MinDeposit, banking)
> abline(bank.fit1)
```

The line fits very poorly. How do we test to see that the fit is poor? We can use an F test based on "Pure Error" because we have some y values measured at the same value of x. In fact, we need only treat x as a categorical variable, the usual ANOVA model, to obtain an estimate of the "Pure Error".

```
> bank.fit5 <- update(bank.fit1, .~ordered(MinDeposit))
> anova(bank.fit1,bank.fit5)
Analysis of Variance Table

Model 1: NumAccounts ~ MinDeposit
Model 2: NumAccounts ~ ordered(MinDeposit)
  Res.Df   RSS Df Sum of Sq      F   Pr(>F)
1      9 14742
2      5  1148  4     13594 14.801 0.005594
```

We reject the null hypothesis that the simpler model is equivalent to the full model, and conclude that the linear regression is not adequate for expressing the details of the model.

We made a big jump here from linear to full anova model, from 2 parameters to 6. Is there some intermediate model which would be adequate? With 6 unique x values, we can fit up to a 5th order polynomial (in fact that is just another way to express the ANOVA model), note the same sum of squares as above. Both models also could output t tests.

```
> anova(update(bank.fit1, .~ poly(MinDeposit,5)))
                    Df  Sum Sq Mean Sq F value   Pr(>F)
poly(MinDeposit, 5)  5 18734.9  3747.0  16.320 0.004085
Residuals            5  1148.0   229.6
> summary(update(bank.fit1, .~ poly(MinDeposit,5)))
Coefficients:        Estimate Std. Error t value Pr(>|t|)
(Intercept)           117.091      4.569  25.629 1.69e-06
poly(MinDeposit, 5)1   71.703     15.153   4.732 0.005186
poly(MinDeposit, 5)2 -112.598     15.153  -7.431 0.000696
poly(MinDeposit, 5)3   29.356     15.153   1.937 0.110442
poly(MinDeposit, 5)4   -6.934     15.153  -0.458 0.666434
poly(MinDeposit, 5)5   -2.337     15.153  -0.154 0.883474
> summary(bank.fit5)
Coefficients          Estimate Std. Error t value Pr(>|t|)
(Intercept)            120.000      4.725  25.399 1.77e-06
ordered(MinDeposit).L   52.590     10.791   4.874 0.004578
ordered(MinDeposit).Q  -81.504     11.690  -6.972 0.000934
ordered(MinDeposit).C   21.988     11.181   1.967 0.106381
ordered(MinDeposit)^4   -5.480     11.454  -0.478 0.652507
ordered(MinDeposit)^5   -1.953     12.663  -0.154 0.883474
```

There is one missing bank so the terms of the ordered factor are not orthogonal, and the P-values are not the same as if we look at nested (sequential) LR-Tests.

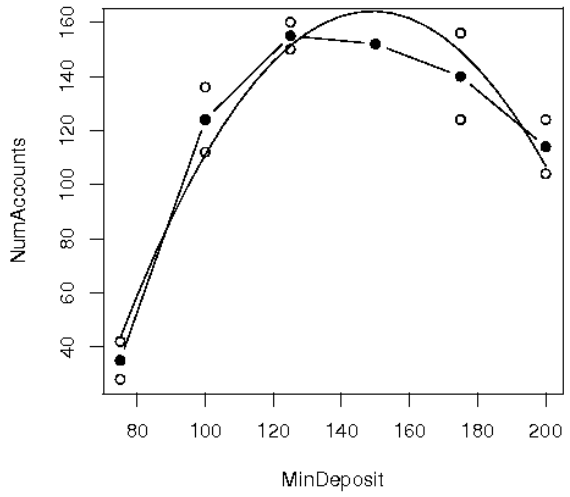
The above t-tests are CONDITIONAL – on what?

```
>  anova(bank.fit0,bank.fit1, bank.fit2, bank.fit3, bank.fit4, bank.fit5)
Analysis of Variance Table
Model 1: NumAccounts ~ 1
Model 2: NumAccounts ~ MinDeposit
Model 3: NumAccounts ~ MinDeposit + I(MinDeposit^2)
Model 4: NumAccounts ~ MinDeposit + I(MinDeposit^2) + I(MinDeposit^3)
Model 5: NumAccounts ~ MinDeposit + I(MinDeposit^2) + I(MinDeposit^3) + I(MinDeposit^4)
Model 6: NumAccounts ~ ordered(MinDeposit)
  Res.Df     RSS Df Sum of Sq       F    Pr(>F)
1     10 19882.9
2      9 14741.6  1    5141.3 22.3926 0.0051865
3      8  2063.3  1   12678.3 55.2189 0.0006955
4      7  1201.5  1     861.8  3.7534 0.1104424
5      6  1153.5  1      48.1  0.2094 0.6664337
6      5  1148.0  1       5.5  0.0238 0.8834736
```

We conclude that the linear term is significant, as is the quadratic, but cubic and higher order terms are not significant. It is possible that one term could be deemed insignificant, while another higher order term is significant. In that case, I would use the higher order model including all lower order terms. It usually makes no sense to omit lower order terms, for instance to force a regression through the origin because the intercept term is not significantly different from zero. I would only do this if a scientist can convince me that it's one of the laws of nature that $\beta_0 = 0$.

```
> XtX <- crossprod(model.matrix(bank.fit5))
> dimnames(XtX) <- list(c("Int",1:5), c("Int",1:5))
> round(XtX,3)             ## demonstrates lack of orthogonality
       Int      1      2      3      4      5
Int 11.000 -0.120  0.436  0.298 -0.378 -0.630
1   -0.120  1.986  0.052  0.036 -0.045 -0.075
```

```
2    0.436  0.052  1.810 -0.130  0.165  0.275
3    0.298  0.036 -0.130  1.911  0.113  0.188
4   -0.378 -0.045  0.165  0.113  1.857 -0.238
5   -0.630 -0.075  0.275  0.188 -0.238  1.603
           ## now add another row with x = 150 to balance the data:
> XtXbal <- crossprod(model.matrix(bank.fit5bal))
>  dimnames(XtXbal) <-  dimnames(XtX)
> zapsmall(XtXbal)            ## polynomials and ordered contrasts are orthogonal
    Int 1 2 3 4 5
Int  12 0 0 0 0 0
1     0 2 0 0 0 0
2     0 0 2 0 0 0
3     0 0 0 2 0 0
4     0 0 0 0 2 0
5     0 0 0 0 0 2
>  anova(bank.fit2, bank.fit5)
Analysis of Variance Table
Model 1: NumAccounts ~ MinDeposit + I(MinDeposit^2)
Model 2: NumAccounts ~ ordered(MinDeposit)
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1      8 2063.32
2      5 1148.00  3    915.32 1.3289 0.3635   ## non-significant Lack Of Fit
```



Why do `poly` and `ordered` differ? Here are the unique rows of the $\boldsymbol{X}$ matrix each is using:

```
## from use of model.matrix() on each fitted model, looking at unique rows
     'Orthogonal' Polynomial                          Ordered Factor:  round(contr.poly(6),4)
 0       1       2       3       4       5           0       1       2       3       4       5
 1 -0.4165  0.3906 -0.2520  0.1345 -0.0377           1 -0.5976  0.5455 -0.3727  0.1890 -0.0630
 1 -0.2468 -0.1037  0.3566 -0.4180  0.1884           1 -0.3586 -0.1091  0.5217 -0.5669  0.3150
 1 -0.0771 -0.3535  0.1712  0.3269 -0.3769           1 -0.1195 -0.4364  0.2981  0.3780 -0.6299
 1  0.0926 -0.3586 -0.2663  0.3645  0.7538           1  0.1195 -0.4364 -0.2981  0.3780  0.6299
 1  0.2623 -0.1191 -0.4137 -0.3457 -0.1884           1  0.3586 -0.1091 -0.5217 -0.5669 -0.3150
 1  0.4319  0.3650  0.2710  0.1200  0.0377           1  0.5976  0.5455  0.3727  0.1890  0.0630
Columns are orthogonal using 'poly'                 Columns are NOT orthogonal (if unbalanced)
```

We used the `poly` function in R to build orthogonal polynomials. How does it create orthogonal columns even when data are unbalanced? Consider a column vector $\boldsymbol{x}$ (n by

1) from which we want to create a set of columns for a 4th order polynomial which are orthogonal. It doesn't matter if $\boldsymbol{x}$ has repeats, or not. Starting with powers of $\boldsymbol{x}$:

$$[\mathbf{1} \;\; \boldsymbol{x} \;\; \boldsymbol{x}^2 \;\; \boldsymbol{x}^3 \;\; \boldsymbol{x}^4]$$

find

$$[\mathbf{1} \;\; \boldsymbol{x}_1^* \;\; \boldsymbol{x}_2^* \;\; \boldsymbol{x}_3^* \;\; \boldsymbol{x}_4^*]$$

such that $(\boldsymbol{x}_i^*)^\mathsf{T}\boldsymbol{x}_j^* = 0$ if $i \neq j$, 1 otherwise. Monahan (2008) illustrates the Gram-Schmidt process in Section 7.4.
Step 1: by centering $\boldsymbol{x}$ to get $\boldsymbol{x}_1^* = \boldsymbol{x} - \overline{x}\mathbf{1}$, we have a second column orthogonal to the first ($\mathbf{1}$). Use projection notation:

$$\boldsymbol{x}_1^* = [\boldsymbol{I} - \boldsymbol{H}_1]\boldsymbol{x} = [\boldsymbol{I} - \mathbf{1}(\mathbf{1}^\mathsf{T}\mathbf{1})^g\mathbf{1}^\mathsf{T}]\boldsymbol{x} = [\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}]\boldsymbol{x} = \boldsymbol{x} - \mathbf{1}\overline{x}$$

Step 2: project $\boldsymbol{x}^2$ into the null space of $[\mathbf{1} \; \boldsymbol{x}_1^*]$ to get $\boldsymbol{x}_2^*$.

$$\boldsymbol{x}_2^* = \left[\boldsymbol{I} - (\mathbf{1} \;\; \boldsymbol{x}_1^*) \begin{pmatrix} \mathbf{1}^\mathsf{T}\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{x}_1^{*\mathsf{T}}\boldsymbol{x}_1^* \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{1}^\mathsf{T} \\ \boldsymbol{x}_1^{*\mathsf{T}} \end{pmatrix}\right] \boldsymbol{x}^2$$

Step 3: project $\boldsymbol{x}^3$ into ...
Step 4: project $\boldsymbol{x}^4$ into ...

4.4 Simultaneous Inference in Linear Models

Setting

The worry is that we will be conducting a large number of tests. For example, if we conduct 20 independent hypothesis tests with $\alpha = 0.05$, and assume (for the sake of the argument) that all 20 null hypotheses are true, then the number of times a null hypothesis is rejected is a $Bin(20, .05)$ which has expectation of 1.00. We "expect" to reject one null even when all are true. If the 20 tests are considered a *family* and we focus on the *Family-wise Type I error rate*, i.e. the probability of making at least one type I error, it is quite large, $1 - 0.95^{20} = 0.64$ in this case. In a typical setting, such as testing all pairwise differences in means in an ANOVA setting, we do over 20 tests when there are six levels to our predictor, so it's not hard to get up to 20 comparisons. Controlling individual tests at the 0.05 level does not provide much control for the family.

Solutions

First, we must guard against *a posteriori* or post-hoc testing. One poor way to conduct scientific investigation is to automatically test "all pairwise comparisons" as part of an ANOVA analysis. Caveat: some researchers have assured me that theirs is a pilot study and that no prior information is available about the effects of different levels of the predictor on the response. In a true pilot study, they may have a point, but I do resist the argument. More typically, the scientist can make groupings ahead of time, for example, to say that levels a_1 and a_2 of factor A have something in common and should be contrasted to the other levels. It is a good idea to elicit planned comparisons because they avoid the problem of "fishing for significance".

Secondly, it helps enormously to use an omnibus test as an initial screening test before allowing consideration of multiple comparisons. If we fail to reject "all means are equal" with an omnibus test F test in ANOVA, then we do not go on to do the family of comparisons.

Thirdly, some multiple comparison methods are better than others. Both Neuman-Keuls and Duncan's test are faulty because family-wise error rates are bigger than α. Tests which are considered acceptable include Scheffé, Tukey, and Bonferroni comparisons.

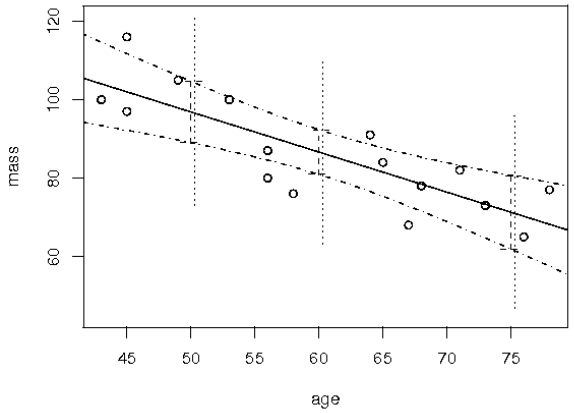
Finally, different software packages offer choices. SAS Proc ANOVA has about 20 multiple comparison procedures for comparing differences in all pairs of means of a factor. R, on the other hand, used to have none of the procedures. The general philosophy of the RCore Team has been that these procedures are dangerous and should only be used by people who are knowledgeable enough to create them themselves. In the Rcode folder off the Stat 506 web page you will find a file called multipleComparisons.R which contains functions I built to compute confidence intervals for all pairwise differences in treatment means using Scheffé, Bonferroni, and Tukey methods. (These are not in finished form).

Another general approach to handling a set of simultaneous tests for which a family-wise error rate is desired, is to adjust individual P-values to account for the simultaneous inference. See help on the `p.adjust` function in R `stats` package. It takes a string of n P-values as input and a method which could be Bonferroni (multiplies each P-value by n), Holm (Holm 1979), or Hochberg (Benjamini and Hochberg 1995) and returns larger p-values which will not result in too many rejected hypotheses. The adjusted p-value approach is explained in Wright (1992).

The `multcomp` package in R now does provide multiple comparisons and adjusted p-values.

The Working-Hotelling band about a polynomial regression curve, is intended to provide $(1-\alpha)100\%$ confidence intervals simultaneously at all x values. The band inverts the distribution used by Scheffé: $\frac{(\boldsymbol{X}^*\hat{\boldsymbol{\beta}}-\boldsymbol{X}^*\boldsymbol{\beta})^{\mathsf{T}}[\boldsymbol{X}^*(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{*\mathsf{T}}]^{-1}(\boldsymbol{X}^*\hat{\boldsymbol{\beta}}-\boldsymbol{X}^*\boldsymbol{\beta})}{ms^2} \sim F_{m,n-r}$ where $\boldsymbol{X}^*\boldsymbol{\beta}$ is a basis set of estimable functions spanning the range of interesting predictor values. For simple linear regression, it is just a column of two ones and a column of two different x values, and $m = 2$. For a polynomial regression of order p, $\boldsymbol{X}^*$ would have $p+1$ rows and columns, and $m = p+1$. The center of each interval is a $\hat{y} = \boldsymbol{x}^*\hat{\boldsymbol{\beta}}$, and points within $se(\hat{y})w$ are included in the interval where $w = \sqrt{mF(1-\alpha; m, n-m)}$. The reasoning behind this approach is identical to the Scheffe procedure.

Example from Kutner et al. (2004) Researchers are studying the loss of muscle mass as we age. They have a measurement of muscle mass for each of 16 women who range in age from 40 to 80.




```
> muscle <- read.table("muscle.data", head=T)
> muscle.fit <- lm(mass ~ age, muscle)
> plot(mass ~ age, muscle, ylim =c(45,121))
> abline(muscle.fit)
> newx <- seq(40,80,length=50)
> W <- sqrt(2*qf(.95,2,14))
> W
[1] 2.734554
>  qt(1-.025/3,14)
[1] 2.717755
> muscle.pred <- predict(muscle.fit, newdata =
                  data.frame(age=newx), se.fit=T)

>  muscle.pred ## I shortened the output
$fit
        1         2         3         4         5         6         7         8
107.10711 106.27152 105.43594 104.60036 103.76477 102.92919 102.09361 101.25802

$se.fit
       1        2        3        4        5        6        7        8
4.375219 4.240817 4.107762 3.976190 3.846253 3.718122 3.591991 3.468078
> lines(newx, muscle.pred$fit - W*muscle.pred$se.fit, lty=4) #Lower  W-H band
> lines(newx, muscle.pred$fit + W*muscle.pred$se.fit, lty=4) #Upper  W-H band
> bonf.muscle.pred <- cbind(
+  predict(muscle.fit, newdata = data.frame(age=c(50,60,75)),
```

```
+                         interval="confidence",level=1-.05/3),
+  predict(muscle.fit, newdata = data.frame(age=c(50,60,75)),
+                         interval="predict",level=1-.05/3))
>  bonf.muscle.pred
       fit      lwr       upr      fit      lwr       upr
1 96.87121 89.08447 104.65795 96.87121 72.89524 120.84719
2 86.63532 80.96183  92.30881 86.63532 63.26006 110.01058
3 71.28148 61.92163  80.64133 71.28148 46.74944  95.81352
  ##      Confidence Interval##       #Prediction Interval#

> arrows(c(50,60,75),  bonf.muscle.pred[,2], c(50,60,75), bonf.muscle.pred[,3],
      lty=2, angle=90,code=3, leng=.1)  ## 3 Bonferroni confidence intervals
> segments(c(50,60,75)+.3,  bonf.muscle.pred[,5], c(50,60,75)+.3,
     bonf.muscle.pred[,6], lty=3)      ## # Bonferroni prediction intervals
```

4.5 Confidence Regions

Confidence intervals for individual coefficients are discussed in many of our lower level stat classes. One uses the estimate and the standard error output in the coefficient table to build the $100(1-\alpha)\%$ CI:

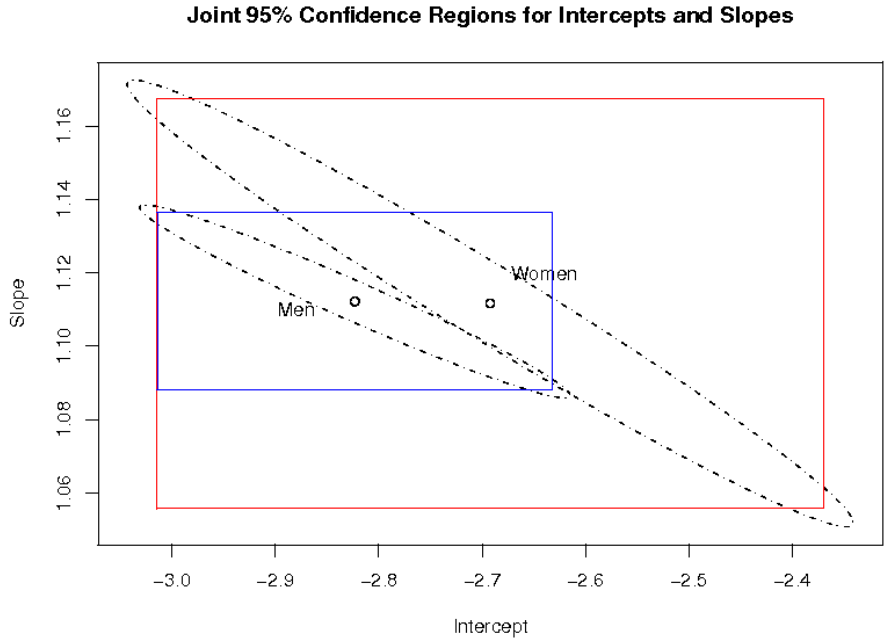
$$\widehat{\beta}_i \pm SE(\widehat{\beta}_i) \times t(1-\alpha/2, n-r) \tag{5}$$

In this course, we need to build **simultaneous** $100(1-\alpha)\%$ confidence regions for several (k) of the parameters. Two approaches are available

1. Bonferroni rectangular regions. Split the value of α into k pieces so that $\alpha = \alpha_1 + \cdots + \alpha_k$. (Typically α is split into k equal pieces.) Use the above formula to build a $100(1-\alpha_k)\%$ CI for each of the k parameters. The intersection of the k intervals (in $\Re^k$) is an at least $100(1-\alpha)\%$ confidence region for the k- dimensional parameter vector. The Bonferroni approach is generally quite conservative.

2. F-based ellipsoidal regions. For an estimable $\boldsymbol{C}'\boldsymbol{\beta}$, we invert the LRT statistic formula
$$F(\boldsymbol{\delta}_0) = \frac{(\boldsymbol{C}'\tilde{\boldsymbol{\beta}} - \boldsymbol{\delta}_0)'[\boldsymbol{C}'(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X})^{-}\boldsymbol{C}]^{-1}(\boldsymbol{C}'\tilde{\boldsymbol{\beta}} - \boldsymbol{\delta}_0)}{q\widehat{\sigma}^2}$$
to find the set of possible $\boldsymbol{\delta}_0$'s which are "close" to $\boldsymbol{C}'\tilde{\boldsymbol{\beta}}$ in that
$$F(\boldsymbol{\delta}_o) \leq F^{1-\alpha}_{q,n-r}$$

With three parameters, the region is a "football-shaped" ellipsoid in $\Re^3$, but for visualization we are restricted to 2-dimensions. The race times data from S&S which has just slopes and intercepts is a good data set to for illustration. The R code below plots rectangular and elliptical 95% confidence regions for (women's intercept, women's slope) and for (men's intercept, men's slope). Why are the regions for women so much larger than those for men? (They were fit in a single model using a common variance parameter.)



R Code

```
> races <- read.table("http://www.math.montana.edu/~jimrc/classes/stat506/data/races",head=T)
> race.fit3 <- lm(ltime ~ -1 + M.int + M.x + F.int + F.x, data=races)
> summary(race.fit3)

Call:
lm(formula = ltime ~ M.int + M.x + F.int + F.x - 1, data = races)

Residuals:
      Min        1Q    Median        3Q       Max
-0.102266 -0.038941 -0.006206  0.046545  0.114664

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
M.int -2.823196   0.076807  -36.76 4.11e-16
M.x    1.112214   0.009687  114.81  < 2e-16
F.int -2.692162   0.129518  -20.79 1.80e-12
F.x    1.111675   0.022427   49.57  < 2e-16

Residual standard error: 0.06178 on 15 degrees of freedom
Multiple R-Squared: 0.9999,Adjusted R-squared: 0.9999
F-statistic: 3.74e+04 on 4 and 15 DF,  p-value: < 2.2e-16

> library(ellipse)
> plot(ellipse(race.fit3,which=3:4),type="l", lty=4,xlab="Intercept",ylab="Slope")
> lines(ellipse(race.fit3),lty=4)
> title("Joint 95% Confidence Regions for Intercepts and Slopes")
> text(c(-2.88,-2.64),c(1.11,1.12),c("Men","Women"))
> points(coef(race.fit3)[c(1,3)], coef(race.fit3)[c(2,4)])
      ## Add 95% Bonferroni confidence region for women
      ## use estimates and their standard errors from the summary coefficient table

> 1-.05/4  ### split up the alpha into 2 parts,
           ### then halve that for 2-sided intervals
[1] 0.9875
> rect(-2.692162 -qt(.9875,15)*  0.129518,
        1.111675 -qt(.9875,15)*  0.022427,
       -2.692162 +qt(.9875,15)*  0.129518,
        1.111675 +qt(.9875,15)*  0.022427, col=2,density=0)
      ## Add 95% Bonferroni confidence region for men
> rect(-2.823196 -qt(.9875,15)* 0.076807,
        1.112214 -qt(.9875,15)* 0.009687,
       -2.823196 +qt(.9875,15)* 0.076807,
        1.112214 +qt(.9875,15)* 0.009687, col=4,density=0)
```

5 Choices in Model Construction

5.1 Fundamentals of Modeling

Conditional Inferences

In the usual coefficient table output for a linear model, each t-test is conditional on the other variables included in the model. If one variable is added or removed, you should expect estimates of the other coefficients to change. That is a generalization; if one column of $\boldsymbol{X}$ is orthogonal to all others, removing it will not change coefficient estimates, but may increase s^2, and thus affect the standard errors of the other coefficients.

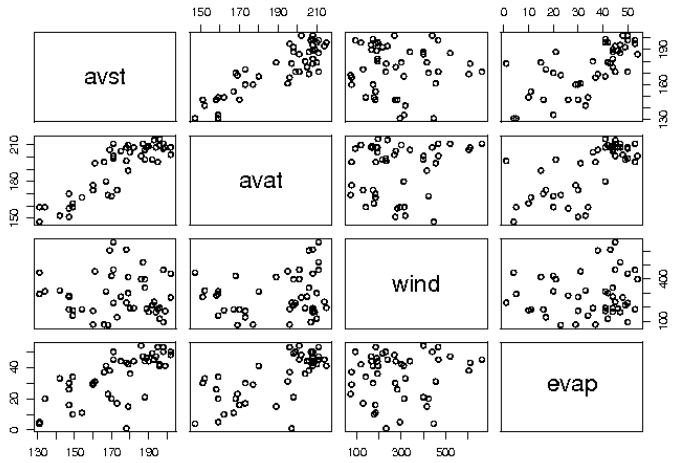
A second setting where conditionality is important is in evaluating effects of multiple predictors by using L-R Tests for nested models. When the predictor columns are not orthogonal, then the order in which terms are added or deleted makes a difference. In the following data, we try to predict evaporation from the soil based on average soil temperature (avst), average air temperature (avat), and wind speed.

```
> soil.fit1 <- lm(evap ~ avst + avat + wind, data=soil)
> anova(soil.fit1)
Analysis of Variance Table

Response: evap
          Df Sum Sq Mean Sq F value    Pr(>F)
avst       1 4558.0  4558.0 41.8642 8.362e-08  ## avst first
avat       1  502.0   502.0  4.6109   0.03759  ## avat after avst
wind       1    9.3     9.3  0.0857   0.77122  ## wind after the others
Residuals 42 4572.8   108.9
```

The first test compares the model with only a mean to a simple linear regression model using avst to predict evap. The second test is between that SLR model and a model using two predictors, avst and avat. The p-value of 0.038 assess the significance of adding the second predictor (not of the second predictor on its own). The third line tests between the bivariate regression and a model with three predictors, so it looks at the effect of wind after avat and avst.

```
> soil.fit2 <- lm(evap ~ wind + avat + avst, data=soil)
> anova(soil.fit2)
Analysis of Variance Table

Response: evap
          Df Sum Sq Mean Sq F value    Pr(>F)
wind       1   89.9    89.9  0.8260    0.3686  ## wind first
avat       1 4783.4  4783.4 43.9341 4.957e-08  ## avat after wind
avst       1  196.1   196.1  1.8007    0.1868  ## avst last
Residuals 42 4572.8   108.9
```

The ordering makes a difference because the predictors are correlated, as we can see in this "pairs" plot:



On the Order of Variable Entry

Before we get to variable selection, I want to warn you about my attitude to model building. I believe strongly that

> **Science must drive the model selection, not black-box statistical techniques.**

Most studies scientists perform are based on previous studies, and are intended to improve our understanding by building on previous knowledge. Applying this principle to variable selection means that we should first include any variables which experts in the field of study would agree are important predictors. If a study is trying to build a better model by adding predictors, then those predictors must be evaluated based on "additional" explanatory power on top of the explanatory power of the "known" variables.

In the past, textbooks and stat courses have made large distinctions between ANOVA models with classification predictors and regression models with continuous predictors. With Proc GLM in SAS and `lm` in R, we no longer need to focus on whether a predictor is categorical or continuous. Of course, we have to include it in a class statement (make it a factor) when appropriate, but the more important consideration is contextual: is this a variable which experts will feel must be included in the model, or is it one which is only now being considered? Historically, the differences between regression, ANOVA, and ANCOVA models were important because of computational issues. Those issues have been solved, so now we can focus on the science, and we should spend more time understanding the science behind the data.

Corollary: The practice of assuming that Type III sums of squares give the best comparisons does not take into account any hierarchy among the predictors, and I feel it is dangerous. It allows users to avoid the serious question of the order of predictor entry into a grand model, which is important to think about.

The important difference is between variables all can agree should be in the model and those which we expect to "prove themselves" as contributors. An example is the block effect in a randomized block design. If it is significant, then the researchers did the right thing by blocking, they have explained part of the underlying variation as being due to differences between blocks. But if the block effect is nonsignificant, does that mean it

should be removed? No. Experts in agricultural trials expect to see differences between blocks. If those differences are small, that does not mean we should dump that variation into the error term. Better to keep it separated and look at treatment effects after adjusting for blocks.

Another approach to model comparison which seems sensible to me is based on *apriori* identification of a suite of models. Each model is fitted to the data, and a likelihood or information criterion (usually AIC or AICc) is computed. The models with superior AIC are declared to be the best-fitting models. More on this in the next section.

5.2 Variable Selection

The Problem

Suppose there are many potential explanatory variables to use in building a model to explain the response, $\boldsymbol{y}$. Claim: fitting extra variables creates unnecessary complexity and inflates the variances of the coefficient estimates. However, leaving out variables which should be included biases our coefficient estimates and inflates the residual variance. Monahan (2008) covers this in section 4.4.

William Ockham (c. 1285 – 1349) is credited with stating, "Plurality ought never be posited without necessity" (Ockham's razor shaves off whatever is unnecessary). So we prefer simple models if they adequately represent the data.

Bias versus Variance Trade-off

Take a case of underfitting where the "true" model is

$$\boldsymbol{y} = [\boldsymbol{X}_1\ \boldsymbol{X}_2] \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix} + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim (\boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

but we omit $\boldsymbol{X}_2$ and $\boldsymbol{\beta}_2$. Then the estimate for $\boldsymbol{\beta}_1$ is $\hat{\boldsymbol{\beta}}_1 = (\boldsymbol{X}_1\boldsymbol{X}_1^\mathsf{T})^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{y}$ which has expected value

$$E(\hat{\boldsymbol{\beta}}_1) = (\boldsymbol{X}_1\boldsymbol{X}_1^\mathsf{T})^{-1}\boldsymbol{X}_1^\mathsf{T}[\boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{X}_2\boldsymbol{\beta}_2] = \boldsymbol{\beta}_1 + (\boldsymbol{X}_1\boldsymbol{X}_1^\mathsf{T})^{-1}\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2\boldsymbol{\beta}_2.$$

If $\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_2$ then $\boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2 = \boldsymbol{0}$ and there is no bias in the estimation of $\boldsymbol{\beta}_1$, but in observational studies, we never see orthogonality, so there is some bias in the estimation of $\boldsymbol{\beta}_1$ dependent on $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and the true $\boldsymbol{\beta}_2$.

Conclusion 1. Underfitting causes bias in coefficient estimates.

Now turn the problem around to look at overfitting. The true model is

$$\boldsymbol{y} = \boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{\epsilon}; \quad \boldsymbol{\epsilon} \sim (\boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

but we fit $[\boldsymbol{X}_1\ \boldsymbol{X}_2] \begin{pmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \end{pmatrix}$. Let $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\mathsf{T}, \boldsymbol{\beta}_2^\mathsf{T})^\mathsf{T}$ be estimated by $\widehat{\boldsymbol{\beta}}$, an unbiased estimator, since the true $\boldsymbol{\beta}_2 = \boldsymbol{0}$. We then look at the variance covariance matrix of the estimator.

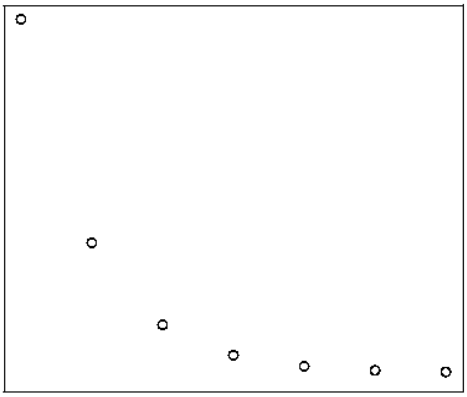
$$\mathrm{cov}(\widehat{\boldsymbol{\beta}}) = \sigma^2 \begin{bmatrix} \boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_1 & \boldsymbol{X}_1^\mathsf{T}\boldsymbol{X}_2 \\ \boldsymbol{X}_2^\mathsf{T}\boldsymbol{X}_1 & \boldsymbol{X}_2^\mathsf{T}\boldsymbol{X}_2 \end{bmatrix}^{-1}$$

Under the overfit model, $\text{cov}(\widehat{\boldsymbol{\beta}}_1) = \sigma^2[\boldsymbol{X}_1^\top\boldsymbol{X}_1 - \boldsymbol{X}_1^\top\boldsymbol{X}_2(\boldsymbol{X}_2^\top\boldsymbol{X}_2)^{-1}\boldsymbol{X}_2^\top\boldsymbol{X}_1]^{-1}$[1] whereas the correct model with just $\boldsymbol{X}_1$ has $\text{cov}(\widehat{\boldsymbol{\beta}}_1) = \sigma^2(\boldsymbol{X}_1^\top\boldsymbol{X}_1)^{-1}$. We'd like to show that $(\boldsymbol{X}_1^\top\boldsymbol{X}_1)^{-1} \leq [\boldsymbol{X}_1^\top\boldsymbol{X}_1 - \boldsymbol{X}_1^\top\boldsymbol{X}_2(\boldsymbol{X}_2^\top\boldsymbol{X}_2)^{-1}\boldsymbol{X}_2^\top\boldsymbol{X}_1]^{-1}$ or equivalently that $\boldsymbol{X}_1^\top\boldsymbol{X}_1 \geq \boldsymbol{X}_1^\top\boldsymbol{X}_1 - \boldsymbol{X}_1^\top\boldsymbol{X}_2(\boldsymbol{X}_2^\top\boldsymbol{X}_2)^{-1}\boldsymbol{X}_2^\top\boldsymbol{X}_1$ or simply that $\boldsymbol{X}_1^\top\boldsymbol{X}_2(\boldsymbol{X}_2^\top\boldsymbol{X}_2)^{-1}\boldsymbol{X}_2^\top\boldsymbol{X}_1$ is non-negative definite. Proof: $\boldsymbol{X}_1^\top\boldsymbol{X}_2(\boldsymbol{X}_2^\top\boldsymbol{X}_2)^{-1}\boldsymbol{X}_2^\top\boldsymbol{X}_1 = \boldsymbol{X}_1\mathbf{P}_2\boldsymbol{X}_1^\top$, a quadratic form with symmetric $\mathbf{P}_2$. The amount of additional variance in $\widehat{\boldsymbol{\beta}}_1$ would be 0 if $\mathbf{P}_2\boldsymbol{X}_1 = \mathbf{0}$, i.e., if $\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_2$.

Conclusion 2. Overfitting increases variance of coefficient estimates.

Of course, one does not actually get to look at the true variance of $\widehat{\boldsymbol{\beta}}$, but at the estimated $s^2 = \widehat{\sigma}^2$. Including extra terms will typically decrease s^2, since another column in $\boldsymbol{X}$ will increase the sum of squares due to the regression model and decrease SSE. If the added term(s) really are unneeded, we do not expect to see much change in the variance estimate.

Expected change in s^2.



Selection Criteria:
Numerous criteria have been proposed to distinguish between models fit with varying numbers of parameters and with different predictors. An argument to keep in mind during this discussion of variable selection: Suppose I want to model vector y which is of length 26, but am too lazy to collect related predictors. I could simply generate 50 columns $[\boldsymbol{X}_1\ \boldsymbol{X}_2\ \cdots\ \boldsymbol{X}_{50}]$, each containing 26 random normals to use as predictors. If we build an $\boldsymbol{X}$ matrix by finding the best combination of the 50 predictors, we will obtain a multiple regression model which explains a high proportion of the variation in $\boldsymbol{y}$. This is not a valid strategy, but is meant to make you skeptical about the practice of sifting through lots of potential predictors to find a few that do a good job.

1. R-squared: $R^2 = 1 - \frac{SSE}{SSy} = 1 - \frac{\sum(y_i - \widehat{y}_i)^2}{\sum(y_i - \overline{y}_i)^2}$, the proportion of variance in $\boldsymbol{y}$ explained by the model. Bigger is better, and R^2 always increases as variables are added, so it never says "No" to added terms. Caution: The fitted model must include an intercept column for R^2 to have meaning.

[1] See Monahan (2008) exercise A.72 for a commonly used expansion of a matrix inverse:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\,(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \\ -(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & (\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \end{bmatrix}$$

2. Adjusted R-squared: $R^2_{adj} = 1 - \frac{\sum(y_i - \widehat{y}_i)^2}{\sum(y_i - \overline{y}_i)^2} \times \frac{n-1}{n-r}$, where $\text{rank}(\boldsymbol{X}) = r$.
   Bigger is better, but it can get smaller when a term is added to the model which does not improve the fit appreciably.

3. Mallow's $C_p = \frac{Q_p}{\widehat{\sigma}^2} - [n - r]$ is computed for each submodel of the full model, $\boldsymbol{X}_r\boldsymbol{\beta}$ (r=rank($\boldsymbol{X}_r$)). For a submodel using $\boldsymbol{X}_p$ with rank $p < r$, Q_p is the residual sum of squares $\boldsymbol{y}^{\mathsf{T}}(\boldsymbol{I} - \mathbf{H}_{X_p})\boldsymbol{y}$ and $\widehat{\sigma}^2$ is the MSE from the full model, using $\boldsymbol{X}_r$. The purpose of C_p is to find a submodel for which C_p is approximately p, and if several are roughly as close, we would choose the one with fewer terms. Why close to p? Assume that s^2 is an excellent estimator of σ^2.

$$E(Q_p)/\sigma^2 - (n - 2p) = \frac{(n-p)\sigma^2}{\sigma^2} - (n - 2p) = n - p - (n - 2p) = p$$

   If we assume that the FULL model provides a good estimate of σ^2, C_p will approximately equal p when bias is negligible. One can compute C_p for several models, then plot them on the y axis against p on the x axis. One might add the $y = x$ line to make it easier to see when $C_p \approx p$.
   Assumption: The full model is available and close to the truth. We want to keep adding terms until we get a submodel which is about as good as the full.

4. Likelihood.
   One can assume a distribution for the errors (Gaussian, typically) and compare two models with a LRT. Larger likelihoods occur as the number of parameters increases, bigger is better, so we need to use some test to see when the improvement is large enough to warrant the more complex model.

   (a) One rule of thumb used by "evidential" statisticians is that a ratio of 8 or more (.125 or less) means that the model on the top (bottom) of the ratio is substantially better than the other model.

   (b) The LRT uses the difference in $-2\ln(\hat{L})$ as having an asymptotic χ^2 distribution with difference in the number of parameters. If normal errors are assumed and σ^2 is estimated by s^2, an F test is used. Since bigger likelihood is "better", we are trying to reduce $-2\ln(\hat{L})$. With **nested** models we can test to see if the additional parameters significantly improve the fit.

5. Adjusted Likelihood (1) **BIC** or **SBC**
   The Bayesian setup for model selection is to identify a large class of possible models and give each some prior probability (the sum of which over all models is 1). One then uses Bayes rule to compute posterior model probabilities as

$$P(\text{Model } A|\text{data}) \propto P(\text{Model } A) \times P(\text{data}|\text{Model } A).$$

   The Bayes Factor is the ratio of two likelihoods: $BF_{AB} = P(\text{data}|\text{Model } A)/P(\text{data}|\text{Model } B)$ and can be reported so that readers can supply their own priors for A and B and find the posterior likelihood ratio by multiplying prior likelihood ratio by BF.

$$\text{Posterior Ratio } = \frac{P(\text{Model } A|\text{data})}{P(\text{Model } B|\text{data})} = \frac{P(\text{Model } A)}{P(\text{Model } B)} \times \frac{P(\text{data}|\text{Model } A)}{P(\text{data}|\text{Model } B)} = \text{Prior Ratio } \times BF_{AB}$$

Schwarz' Bayesian Information Criterion adjusts $-2\ln(\hat{L})$, adding a penalty for the number of parameters. We want to minimize the quantity:

$$BIC = -2\ln(\widehat{L}) + p\ln(n).$$

The Bayesian connection is that, asymptotically, the BIC approximates $\ln(BF_{AB})$. It is designed to compare two or more models, one of which is assumed to be the "true" model. One advantage of BIC over LRT is that models need not be nested to use the BIC.

The BIC is criticized as allowing too few parameters to enter the model. It is commonly used as an evaluation technique without the assumption of a Bayesian approach.

6. Adjusted Likelihood (2) **AIC**
The Akaike Information Criterion (1973) also adjusts $-2\ln(\hat{L})$, adding a penalty for the number of parameters. We want to minimize the quantity:

$$AIC = -2\ln(\widehat{L}) + 2p.$$

Some stat packages multiply the formula by -1, and change minimize to maximize, so do be careful when using another stat package. For $\ln(n) = 2$, i.e. $n = e^2$, AIC and BIC agree, but for more usual cases where $n > 9$, the AIC is more liberal, giving less of a penalty to added terms, so more terms are added than when using BIC.

Justification is based on the predictive distribution of future data, the AIC picks the model which gives the "best approximation" (nearest in Kullback–Liebler distance) to a true unknown model. Theoretically, the AIC works well if the complexity of the model grows with sample size.

7. Adjusted Likelihood (3) **AICc**
AIC is known to apply too small a penalty when sample sizes are small, so this version adds more penalty for small n, but approaches AIC as n gets large. The Corrected AIC is intermediate between AIC and BIC, since it uses a multiplier bigger than 2, but generally smaller than $\ln(n)$.

$$AIC_c = -2\ln(\widehat{L}) + 2p\frac{n}{n-p-1} = AIC + \frac{2p(p+1)}{n-p-1}$$

Hurvich & Tsai (1989) *Biometrika*. As $n \to \infty$, $\text{AIC}_c \to \text{AIC}$
AIC, AIC_c, and C_p all assume that the true model is of high dimension. They give basically similar results for normal–error models with large n.

8. A Graphical Method: **Adjusted Variable Plots**
Suppose that we start a model-building procedure by picking one column of $\boldsymbol{X}$ (perhaps using the column with strongest correlation to $\boldsymbol{y}$). We then consider adding other columns, one at a time, at each stage choosing the best of the remaining columns. We could fit all the models with one added predictor and compare the fits based on any of the above statistics, but instead we want to **see** how the candidate variables are related to the residuals of the current model. A simple approach would be to plot the residuals versus each of the candidate variables, and look for trends.

However, since all the columns of $\boldsymbol{X}$ might be correlated, and we know collinearity could be a problem, it's not really the new candidate variables we need to plot against; but rather the **additional** effects of candidate variables that we want to see. We should plot the residuals (y axis) against $(\boldsymbol{I} - \mathbf{H}_1)\boldsymbol{X}_2$, not just $\boldsymbol{X}_2$, where $\boldsymbol{X}_1$ is the collection of variables already in the model, and $\boldsymbol{X}_2$ contains the candidate variables.

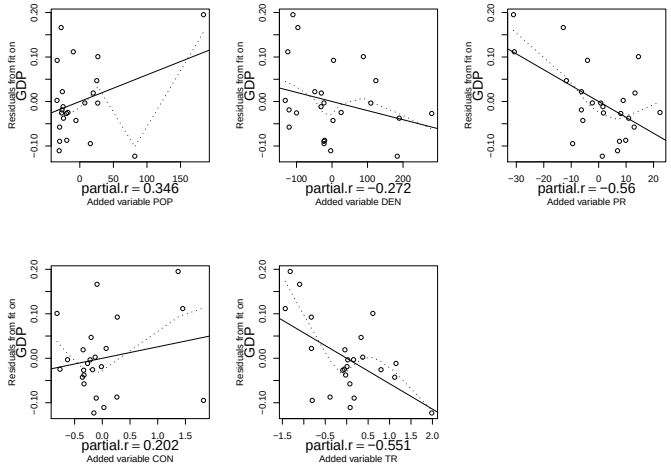
$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\epsilon} \\ (\mathbf{I} - \mathbf{H}_1)\boldsymbol{y} &= (\mathbf{I} - \mathbf{H}_1)\boldsymbol{X}_1\boldsymbol{\beta}_1 + (\mathbf{I} - \mathbf{H}_1)\boldsymbol{X}_2\boldsymbol{\beta}_2 + (\mathbf{I} - \mathbf{H}_1)\boldsymbol{\epsilon} \\ &= (\mathbf{I} - \mathbf{H}_1)\boldsymbol{X}_2\boldsymbol{\beta}_2 + (\mathbf{I} - \mathbf{H}_1)\boldsymbol{\epsilon} \end{aligned}$$

[Boik's "Annihilator" applied to both new predictor and to $\boldsymbol{y}$.]
The nice thing about the plot is that we can see trends that are not just linear and can compare to a "smoothed" estimate of the trend. Also one can see if there is a general linear trend or if there is a single outlying point with high leverage.

Added Variable Plots

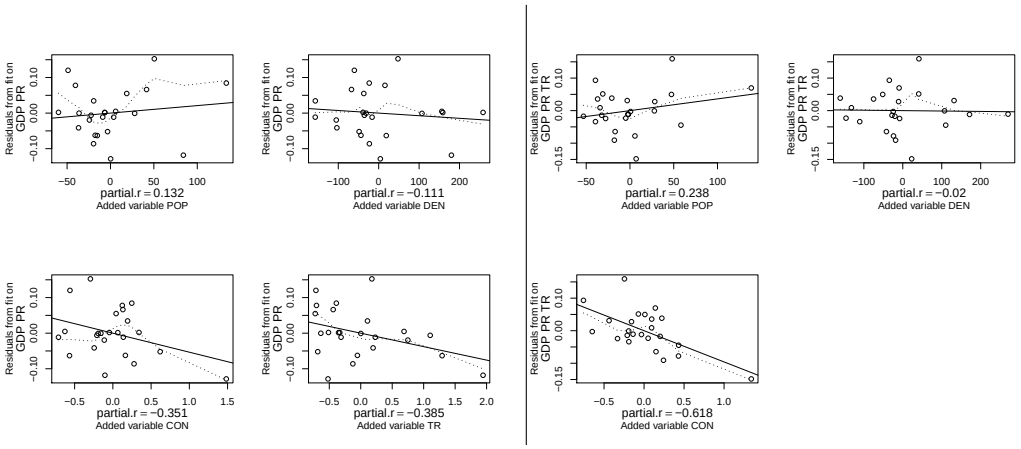
In a study of 24 countries, the response AO, auto ownership, is collected along with six possible predictors: population, Density, gross domestic product, price of fuel, conservation, and train use.

```
auto.own <- read.table("data/autoUse.txt",head=T)
round(cor(auto.own), 3)[1,]                      ## just 1st row
    AO    POP    DEN    GDP     PR    CON     TR
 1.000  0.278 -0.042  0.728 -0.327  0.076 -0.119
 ## highest pairwise correlation is with GDP
auto.fit1 <- lm(AO ~ GDP, auto.own)
x11()
 added.variable.plots(auto.fit1)          ## shown below
```

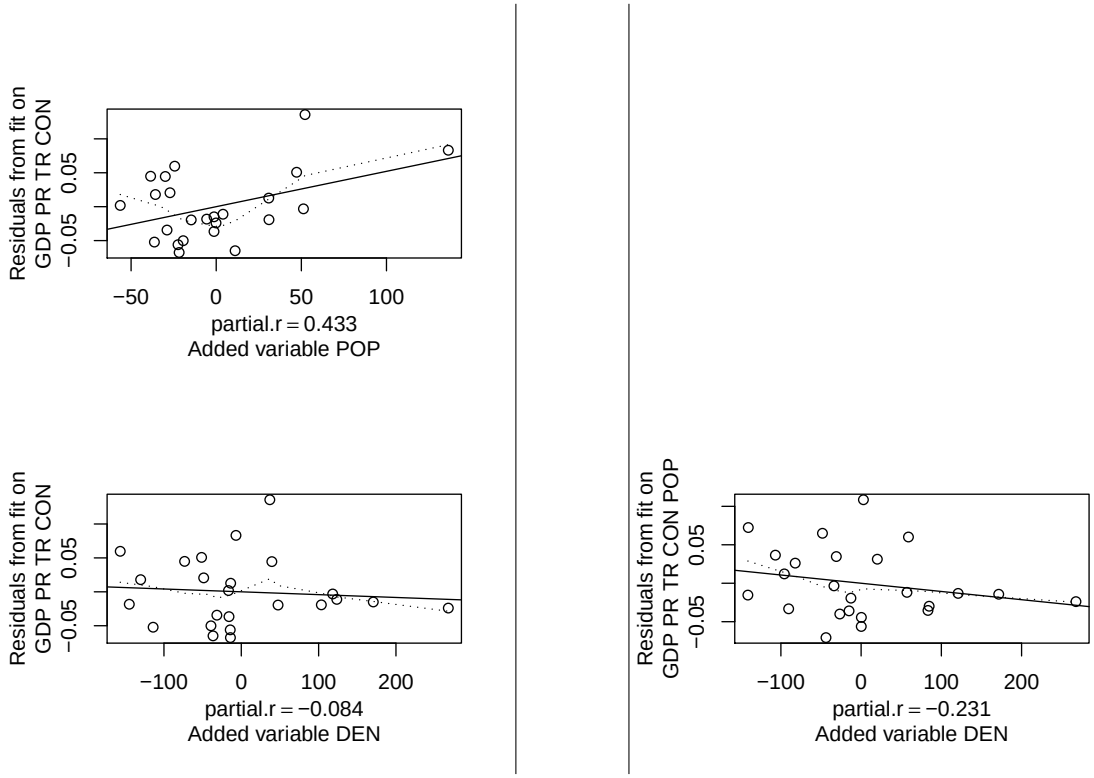


The first model uses only GDP to predict AO, and the plots show regressions of residuals from this model on the residuals of each other predictor in turn. Partial correlation is the correlation between AO and X_k after adjusting for the effect of GDP. The predictor with strongest partial correlation is added to the model.

```
auto.fit2 <- update(auto.fit1, .~. + PR)
 added.variable.plots(auto.fit2)          ## below left
```




```
auto.fit3 <- update(auto.fit2, .~. + TR)
 added.variable.plots(auto.fit3)          ## above right
auto.fit4 <- update(auto.fit3, .~. + CON)
 added.variable.plots(auto.fit4) ## below left
```



Keep adding variables, one at a time.

```
auto.fit5 <- update(auto.fit4, .~. + POP)
 added.variable.plots(auto.fit5)
auto.fit6 <- update(auto.fit5, .~. + DEN)
                      ## compare models with F tests, AIC, BIC, adj.Rsqd
anova(auto.fit1, auto.fit2, auto.fit3, auto.fit4, auto.fit5,auto.fit6)
#  Res.Df      RSS Df Sum of Sq       F    Pr(>F)       AIC     BIC       R^2adj
```

```
#1     22 0.160955                                      -48.00    -45.64    0.50879
#2     21 0.110462  1  0.050493 19.2089 0.0004059    -55.03    -51.50    0.64683
#3     20 0.094061  1  0.016401  6.2395 0.0230470    -56.89    -52.18    0.68423
#4     19 0.058099  1  0.035962 13.6809 0.0017826    -66.45    -60.56    0.79469
#5     18 0.047202  1  0.010897  4.1456 0.0576319    -69.44    -62.37    0.82394
#6     17 0.044686  1  0.002515  0.9568 0.3417076    -68.75    -60.51    0.82351
AIC(auto.fit1, auto.fit2, auto.fit3, auto.fit4, auto.fit5,auto.fit6)
BIC <- 1:6
for( i in 1:6)
   BIC[i] <- AIC(list(auto.fit1, auto.fit2, auto.fit3, auto.fit4, auto.fit5,auto.fit6)[[i]],
                    k=log(24 ))
adj.rsquared(auto.fit1, auto.fit2, auto.fit3, auto.fit4, auto.fit5,auto.fit6)
 drop1(auto.fit6)
#Single term deletions
#Model:
#AO ~ GDP + PR + TR + CON + POP + DEN           (Wrapped Output)
#      Df Sum of Sq      RSS      AIC                  Df Sum of Sq      RSS      AIC
#<none>                0.045 -136.867          #CON    1     0.043    0.088 -122.583
#GDP    1     0.230    0.275  -95.273          #POP    1     0.013    0.058 -132.738
#PR     1     0.027    0.071 -127.622          #DEN    1     0.003    0.047 -137.553
#TR     1     0.044    0.089 -122.431
```

Conclusion? Which model(s) do you prefer? Do all criteria agree?

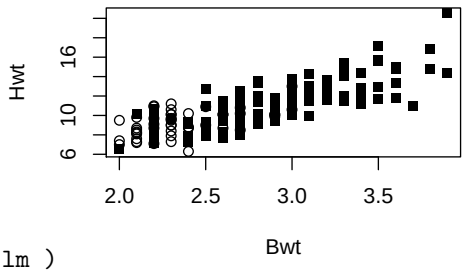
Model Averaging

A recently developed approach is to recognize that the selection of a model is dependent on the data obtained. If we were to gather a new sample of data from the same process which generated the original data, the data would differ, and even though we used the same model selection criteria, we would probably choose a different model. The usual report from a regression analysis ignores such error and assumes that we knew the correct model all along. The standard errors of predictions are obtained from the formula for prediction variance. However, not knowing the true model adds another layer of variability which is hard to gauge. Two approaches have been suggested.

1. **Bayesian Model Averaging**. The identity of the true model is considered a parameter with a discrete distribution. Use intensive computer simulation to approximate the posterior of this parameter. For each model, compute predictions or estimates as needed, then look at the distribution of these values across the "model ID" parameter space, for instance finding mean and variance for the estimates. Hoeting et al. (1999)

2. **Frequentist Model Averaging**. Burnham and Anderson (2002) suggest combining information over several models based on model weights which are based on AIC's. Hjort and Claeskens (2004) expand on that idea and discuss some variations. For example, if I have three models under consideration and am using AICc as my criterion, a comparison might give this table:

```
> require(MASS)
> plot(Hwt ~ Bwt, pch = c(1,15)[as.numeric(Sex)], cats)
> cat.1line.lm = lm(Hwt ~ Bwt, data = cats)
> cat.2lines.lm = lm(Hwt ~ Bwt*Sex, data = cats)
> cat.parallel.lm <-  update(cat.1line.lm, . ~ Bwt + Sex)

## using a comparison function I wrote to get AICc:

> AICc(144,  cat.2lines.lm, cat.1line.lm ,  cat.parallel.lm )
                 df      AIC     AICc     Delta    weight       ER
cat.2lines.lm     5 519.9828 520.2727 0.0653835 0.4147713 1.033232
cat.1line.lm      4 522.0472 522.2198 2.0125084 0.1566736 2.735336
cat.parallel.lm   3 520.1216 520.2073 0.0000000 0.4285550 1.000000
```



The column labeled Delta is the difference in AICc between the model on a line and the smallest–AICc model. The weight is given by $w_i = e^{\delta_i} / \sum_j e^{\delta_j}$, so that weights sum to 1. in this case the one–line model gets much less weight than the two–lines or parallel–lines models. one could build an "AICc averaged" model by multiplying coefficient estimates by weights and summing over the three models.

```
> F.intercept <-  c(coef(cat.2lines.lm)[1], coef(cat.parallel.lm)[1], coef(cat.1line.lm)[1])
> M.intercept <-  c(sum(coef(cat.2lines.lm)[c(1,3)]), sum(coef(cat.parallel.lm)[c(1,3)]), coef(cat.
> F.slope <-  c(coef(cat.2lines.lm)[2], coef(cat.parallel.lm)[2], coef(cat.1line.lm)[2])
> M.slope <-  c(sum(coef(cat.2lines.lm)[c(1,3)+1]), coef(cat.parallel.lm)[2], coef(cat.1line.lm)[2]
> weights <- matrix(c( 0.415,0.157, 0.429),ncol=1)
> rbind(F.intercept,F.slope,M.intercept,M.slope) %*% weights
             [,1]   [,2]   [,3]
F.intercept  1.24 -0.065 -0.153
F.slope      1.09  0.640  1.731
M.intercept -0.49 -0.078 -0.153
M.slope      1.79  0.640  1.731
> rbind(F.intercept,F.slope,M.intercept,M.slope) %*% weights
              [,1]
F.intercept  1.019
F.slope      3.465
M.intercept -0.722
M.slope      4.160
```

Again, model averaging is an attempt to include an often ignored source of variation: model selection is conditional on the available data, and re-collecting data might change the models we select by any of the criteria.

The black boxes

there are methods for people who want to "insert data, let the computer decide which model is best." they are not recommended by any statistician i know, but they are available and commonly used, so we need to know about them. a major problem is that most of these methods can guarantee that the best model (by whatever criterion) has been located. there might be another good model which was never evaluated. to search across all possible subsets of a large list of predictors is time consuming, and if interactions are also considered, the list of possible models can be enormous.

Stepwise Methods:

1. **Forward**: Start with a small set of predictors (just intercept, perhaps) and add one at a time if they can pass some test for inclusion. Continue until no variables pass the test.

2. **Backward**: Start with the largest model and allow one variable at a time to be deleted if it fails some test of significance. Continue until no more terms fail the test.

3. **Stepwise**: Start with any number of predictors. At each step check for other variables to add, as in "Forward". Check for variables to drop, as in "Backward". Continue until no more changes are indicated.

Commonly used criteria: AIC, BIC, C_p, R^2_{adj}, or F test with different significance levels for inclusion and exclusion. In SAS, Proc Reg has a `SELECTION =` option after a model statement. Options include `FORWARD, BACKWARD, STEPWISE, CP`, and `ADJRSQ`. The first three use F tests, and the user sets the significance levels `SLENTRY` and `SLSTAY`. In practice people tend to set the entry significance level higher than that for removal. Then it's easier for a variable to enter the model than it is to stay, but the variable does get to join the party (and might force out some other predictor).

In R the `step()` function uses the AIC criterion or C_p.

There are methods which guarantee to find the best model involving only main effects by searching all possible subsets of a list of possible predictors from each term singly to all pairs, on up to all terms included. In SAS, Proc Reg has the option `selection=RSQUARE` in a model statement, and in R the `leaps` function in the `leaps` library does the same. A disadvantage is that you have to build your own columns of dummy variables for a factor, and you cannot insist that all columns for some factor stay or go as one inseparable group. If you are considering a quadratic in x_k, the "best" model could keep x_k^2 and drop x_k, which is not wise.

Final Comments:
Much of the above discussion of ways to pick predictor variables assumes that we have no other background information to guide the search for a reasonable set of predictors. The preferred situation is when the science and previous research about the study informs us and tells us which models "make sense." If scientists can come up with a handful of reasonable models, then one could use a criterion like AIC_c to pick the one preferred by this data. Searching blindly should be discouraged. There are situations where we explore a new field, "pilot studies" where the above methods might be all we have to rely on. In such cases, I prefer to explore the models manually with the help of added variable plots.

5.3 Interactions

Definition: An interaction between two predictors means that the effect of one predictor on the response varies with the value of the second predictor.

If interactions are present, it might be difficult to interpret main effects. This makes the process of model building difficult, because we might have to include large numbers of interactions in a multiple regression situation. Two-way interactions are relatively easy to interpret. If the two predictors are continuous, then a main effects model is a plane, and picking any value for x_1 slices through the 3D space to give a line for the relationship between x_2 and y. Choosing a different x_1 value will change the intercept, but not the slope (the effect of x_2 is the same for all x_1 values. Adding an interaction between x_1 and x_2 warps the $E(y)$ plane, and allows slopes for the regression of y on x_2 to vary with x_1.

A two-way interaction between classification variables (factors) is present the effect of one predictor changes with the level of the second predictor. If one predictor (x_1) is continous and the other categorical (factor A), then an interaction means that the slope of the regression of y on x_1 depends on which level of A we are examining.

A three way interaction between x_1, x_2, and x_3 means that the interaction between x_1 and x_2 varies with the value of x_3. Similarly for higher way interactions: you can pull out one predictor, x_k and say that the remaining effects depend on the value of x_k.

Here is an example of a three way interaction from an experiment conducted at MSU.

Example (Martell and Leavitt 2002):
A total of 303 subjects (students in a psych class) were asked to view a video of a group working to accomplish a task. After the video was concluded, they filled out a questionnaire about what they had seen. The video showed a group of ROTC candidates building a bridge across a pool from ropes and planks in order to transport themselves and a box across the pool.

Three Factors in an Experiment

- Performance Cue
  Before viewing the video, subjects are told either:
  - (Positive) "The group you are about to watch did quite a good job (top quarter of all groups rated by experts)." or:
  - (Negative) "The group you are about to watch did quite a poor job (bottom quarter of all groups rated by experts)."

  Previous research has established that individuals are strongly influenced by such cues. This research was performed to determine if groups evaluations are more resistant to the influence of performance cues.

- Group Size
  After viewing the video, subjects are randomly assigned to a group size.

  group of 1 Individuals: $n_1 = 103$ individual subjects

  group of 4 Groups of 4: $n_2 = 50$ groups

  If in a group of 4, instructions are to come to consensus about each entry in the questionnaire. Note: it is known in advance that groups tend to exhibit smaller variance than individuals.

- Type of Behavior
  The list of behaviors in the questionnaire includes
  - Effective behavior which facilitated the groups efforts (loudly encouraging others to cross).
  - Non-helpful behavior which was counterproductive (dropping a rope).

  Each questionnaire contained the same list of behaviors, so the type of behavior is a repeated measure on each group.

Responses of Interest

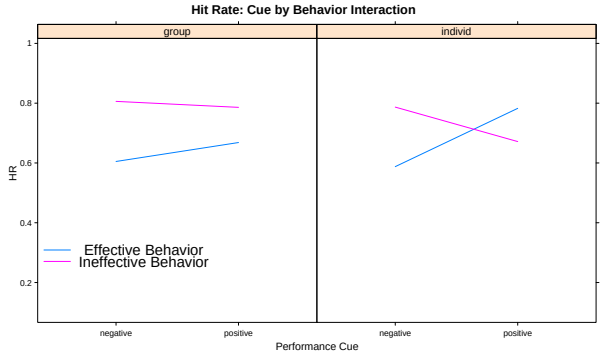
Subjects are asked if certain behaviors occurred in the video. Responses to each behavior were "yes" (occurred) or "no"(did not occur). In the list of 22 effective behaviors, 11 actually occurred and 11 did not. In the list of 18 non-effective behaviors, again half actually occurred.

- Hit Rate
  The proportion of "yes"es out of the behaviors which **did** occur. Computed separately for effective and ineffective behaviors.

$$\frac{\text{number of correct yes's} + .5}{\text{number of behaviors which occurred} + 1}$$

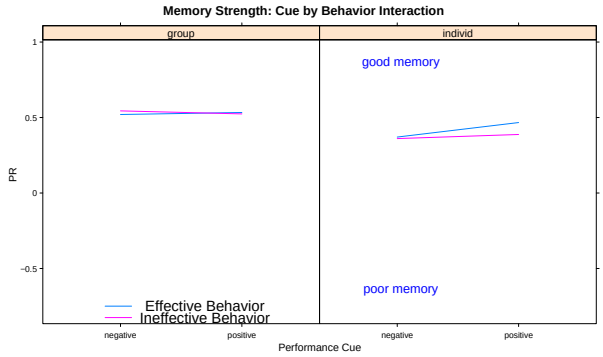
- Memory Strength
  Hit Rate - False Alarm Rate. Ranges from -1 (no memory) to +1 (perfect memory).

The **Hit Rate** response.



- For groups: weak interaction
- For individuals: strong interaction (may even swamp main effects)
- 3-way interaction between Cue, Behavior and Group Size, due to different 2-way interactions.

The **Memory Strength** response.



- For groups: no interaction
- For individuals: weak interaction?
- Very weak (if any) 3-way interaction between Cue, Behavior and Group Size.

Split Plot Analysis of Hit Rate

Ignore the unequal variances problem, and use a "Split Plot" analysis. Each group (whether of 4 or of 1) records answers for the entire questionaire including positive and negative behaviors, so behavior is "applied" later, after the performance cue and after units are assigned to groups. We expect to see some correlation of response within groups and between behaviors.

```
> HRaov <-  aov(HR ~ cue * group * behavior + Error(ID), data = exp2)
                ##   main effects, 2-way and 3-way interactions
> summary(HRaov)
Error: ID
            Df  Sum Sq Mean Sq F value  Pr(>F)
Pcue         1 0.07886 0.07886  3.9632 0.04833
group        1 0.00459 0.00459  0.2305 0.63185
Pcue:group   1 0.00542 0.00542  0.2726 0.60236
Residuals  149 2.96485 0.01990

Error: Within
                     Df Sum Sq Mean Sq F value    Pr(>F)
behavior              1 0.2256  0.2256  7.4571  0.007081
Pcue:behavior         1 1.1347  1.1347 37.5093 7.715e-09
group:behavior        1 0.2611  0.2611  8.6316  0.003830
Pcue:group:behavior   1 0.2116  0.2116  6.9949  0.009051
Residuals           149 4.5073  0.0303
```

However, the above analysis ignores the "groups are less variable" issue.

Mixed Model (preferred) on Hit Rate

```
> library(nlme)
> HRlme <- lme(HR ~ Pcue * group * behavior, data = exp2, random=~1|ID)
> HRlme <- update(HRlme, weights= varIdent(form = ~1|group))
> intervals(HRlme)         #   gives  Approximate 95% confidence intervals
 Random Effects:          #                 (skip Fixed effects intervals)
  Level: ID               #  random constant for ID of individual or group
                        lower         est.        upper
sd((Intercept)) 3.592019e-30 0.0002269009 1.433289e+22
                  ##    ^^ might as well be zero to infinity
 Variance function:          lower     est.    upper     ##  ratio of variances
                  individ 1.101539 1.307539 1.552063
 Within-group standard error:        lower      est.     upper
                                 0.1129827 0.1301163 0.1498482
> anova(HRlme)     numDF denDF  F-value p-value
Pcue                   1   149    2.911  0.0901
group                  1   149    0.204  0.6524
behavior               1   149   17.866  <.0001
Pcue:group             1   149    0.258  0.6124
Pcue:behavior          1   149   38.762  <.0001
group:behavior         1   149   13.270  0.0004
Pcue:group:behavior    1   149   10.054  0.0018
```

Comparison of Split plot and Mixed Model Analyses

Effect	`aov` F	`lme` F
`Pcue`	3.96	2.91
`group`	0.23	0.20
`behavior`	7.46	17.87
`Pcue:group`	0.27	0.26
`Pcue:behavior`	37.5	38.76
`group:behavior`	8.63	13.27
`Pcue:group:behavior`	6.99	10.05

Mixed Model:

$$y_{ijkl} = \mu + \alpha_i + \beta_j + \gamma_k + (\alpha\beta)_{ij} + (\alpha\gamma)_{ik} + (\beta\gamma)_{jk} + (\alpha\beta\gamma)_{ijk} + b_{ijl} + \epsilon_{ijkl}$$

Where α_i is the effect of performance cue, $i = 1, 2$
β_j is the effect of groupsize, $j = 1, 2$
γ_k is the effect of behavior type, $k = 1, 2$, and l indexes the raters.
Combinations represent the interaction effects, e.g. $(\alpha\beta)_{ij}$ for the combined Pcue-group effect. b_{ijl} is a constant which differs between individual raters; it does not depend on the behavior type, and is assumed independent of the residual error, ϵ_{ijkl} $(l = 1, 2)$. Residual error, ϵ, is assumed $N(0, \sigma_4^2)$ in groups, and $N(0, \sigma_1^2)$ for individuals. The software estimates σ_4^2 and the ratio σ_1^2/σ_4^2.

Memory Strength Mixed Model Analysis

```
> MSlme <- update(HRlme, MS~.)
> anova(MSlme)
                    numDF denDF   F-value p-value
(Intercept)             1   149 1837.5700  <.0001
Pcue                    1   149    0.5277  0.4687
```

```
group                 1   149   35.6662  <.0001    ***
behavior              1   149    1.6671  0.1986
Pcue:group            1   149    2.1890  0.1411
Pcue:behavior         1   149    1.9371  0.1661
group:behavior        1   149    1.5304  0.2180
Pcue:group:behavior   1   149    0.1640  0.6861
```

Non-significant interactions.
Only GroupSize has a strong effect.

5.4 Multicollinearity

The Problem

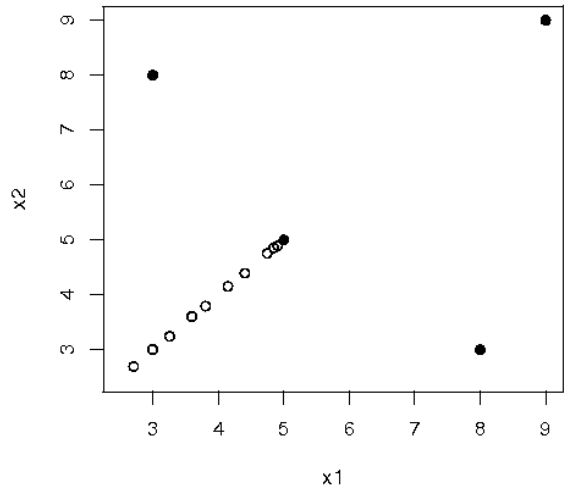
In ANOVA (and related) models the $\boldsymbol{X}$ matrix can have less than full column rank, but we have to then be careful to restrict inference to estimable functions of $\boldsymbol{\beta}$. In multiple regression, $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ is almost never singular (of less than full column rank), but in observational studies where values of $\boldsymbol{X}$ are observed, not set (as in designed experiments), we often have strong correlations between the columns of $\boldsymbol{X}$, and $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ may be **ill-conditioned**, meaning one column of $\boldsymbol{X}$ is "almost" a linear combination of the other columns. Ill-conditioning can occur when the scale of one column is very small, and that of another is very large. Computers have trouble representing numbers very close to 0 and very large numbers, and to have both in the same matrix is an even larger challenge. Rescaling to give all columns variances of 1 will solve this problem, and does not hurt anything (unless you're tied to the particular scales you are using.)

Example 10.1 from Sen and Srivastava (1997), the open circles in the plot below.

Data table:

x_1	2.705	2.995	3.255	3.595	3.805	4.145	4.405	4.745	4.905	4.845
x_2	2.695	3.005	3.245	3.605	3.795	4.155	4.395	4.755	4.895	4.855

Several different $\boldsymbol{y}$ vectors illustrate how a small change in $\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$ affects $\hat{\boldsymbol{\beta}}$. The problem is the high correlations between columns `x1` and `x2` and among the $\boldsymbol{y}$'s.




```
> round(cor(example10.1),4)
       x1     x2     y1     y2     y3
x1 1.0000 0.9999 0.9979 0.9971 0.9950
x2 0.9999 1.0000 0.9973 0.9974 0.9950
y1 0.9979 0.9973 1.0000 0.9919 0.9952
y2 0.9971 0.9974 0.9919 1.0000 0.9918
y3 0.9950 0.9950 0.9952 0.9918 1.0000
```

We can see in the plot that the open circles, x_1 and x_2, are almost equal across their range from 2.7 to 4.9. Two of the filled symbols show parts of the plane which are not included in the dataset.

The third dark point is an extrapolation point which has the same value for x_1 and x_2. If we build a model using both predictors:

$$\boldsymbol{y} = \beta_1\boldsymbol{x}_1 + \beta_2\boldsymbol{x}_2 + \boldsymbol{\epsilon}$$

(Note that we are forcing y to be zero when both x's are, a very strong assumption.) Then

$$\boldsymbol{X}^\mathsf{T}\boldsymbol{X} = \begin{bmatrix} 160.9172 & 160.9298 \\ 160.9298 & 160.9433 \end{bmatrix} \text{ and } (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1} = \begin{bmatrix} 1001.134 & -1001.050 \\ -1001.050 & 1000.972 \end{bmatrix}$$

We fit the three very similar $\boldsymbol{y}$ vectors, and see that estimates of β_2 vary wildly.

y_1	4.10	4.34	4.95	5.36	5.64	6.18	6.69	7.24	7.46	7.23
y_2	4.10	4.73	4.81	5.30	5.75	6.26	6.61	7.13	7.30	7.32
y_3	4.06	4.39	5.02	5.23	5.57	6.50	6.65	7.26	7.48	7.39

```
> fit1 <- lm(y1 ~ x1 + x2 -1, example10.1)
> fit2 <- update(fit1, y2~.); fit3 <- update(fit1, y3~.)
```

	Estimate	Std. Error	t value	s
fit1	-3.701334	2.607239	-1.419638	0.0824
fit2	2.903238	2.978656	0.9746807	0.0941
fit3	0.9747165	4.662316	0.2090627	0.1474

Recall that the estimated variance matrix for $\widehat{\boldsymbol{\beta}}$ is $s^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$, and let's look at that matrix:

```
> cvb <-  0.08241^2* summary(fit1)$cov.unscaled
  6.799107 -6.798537   #s = 0.08241 is small
 -6.798537  6.798009   #but  est. variance/covariance of beta-hat is large
```

And the correlation matrix:

```
>  diag(1/sqrt(diag(cvb)))%*%  cvb %*% diag(1/sqrt(diag(cvb)))
  1.0000000 -0.9999969
 -0.9999969  1.0000000
```

We see that the correlation is practically -1. Next look at standard error of predictions for the four black points in the plot above:

```
> predict(fit1, newdat=list(x1=c(5,9,3,8),x2=c(5,9,8,3)),se.fit=T)$se.fit
 0.03249747  0.05849544 13.03560052 13.03791735
```

Prediction standard errors are small for the first two points where $x_1 = x_2$, even though we extrapolate way out to 9, but `se.fit`s are huge for points perpendicular to the $x_1 = x_2$ line. It's as if x_1 and x_2 are fighting over a single slope, and what one wins, the other loses.

Definition: A data matrix, $\boldsymbol{X}$ is said to be highly multicollinear if there is a column of $\boldsymbol{X}$ which is strongly correlated with a linear combination of the other columns. A simple case is when two columns share the same information and are highly correlated (pairwise), but this definition also includes cases where no two columns are highly correlated.

Is Multicollinearity a problem?

If you know that you need to include all the predictor variables in the model, then multicollinearity is not a problem. Also, if the purpose of the study is for **PREDICTION**, then you don't need to worry about having too many similar variables in the model. It is a problem to deal with when we are interested in describing relationships between the predictors and the response, and in trying to build a useful but parsimonious model.

Dealing with Multicollinearity

We'll see below that our choices include:

- drop one of the predictors,
- create a new predictor which combines several columns, or
- gather more data from the region where the two variables differ. Advice for those planning experiments is always to include the full range of possible data, partly to avoid this problem.

Intercept Problems

Frequently slope estimates are strongly correlated with the intercept estimate, because the mean for the predictor is far from zero. Centering a predictor does remove this correlation, and is a good first attempt to reduce multicollinearity. Scaling all predictors to have sum of 1 will prevent problems which arise from vastly different scales (don't try to center and scale sum to 1, but you could center to 0 and scale to have variance 1). These fixes are "cheap and easy", and do not create problems, though to interpret coefficients and make predictions for new $\boldsymbol{x}$ values, the transformations should be recorded (mean and sum or standard deviation of the original columns). If these simple measures remove the multicollinearity problem, then it was not a serious problem to begin with, rather it was an artifact of the center and scale chosen.

Detecting Multicollinearity

Four diagnostics are available: tolerances, variance inflation factors, condition numbers, and variance components. Note: the correlation matrix of $\boldsymbol{X}$ does show us if two columns are highly collinear (r near ± 1), but misses problems when one column is almost a linear combination of two or more other columns. It's also a good idea to look at a matrix of scatterplots, where each picture is a plot of one pair of columns.

Tolerance

Regress a column, $\boldsymbol{x}_j$ on all other columns, $\boldsymbol{X}_{-j}$, and record the R_j^2 value, the proportion of variation of $\boldsymbol{x}_j$ explained by the regression on the other columns. R_j^2 will be near 1 when collinearity occurs.

$$\text{TOL}_j = 1 - R_j^2$$

Tolerances close to 0 are indicative of collinearity problems.

Variance Inflation Factor, the inverse tolerance, contains essentially the same information

$$\text{VIF}_j = \frac{1}{\text{TOL}_j}$$

Large values of VIF indicate collinearity problems.

Condition numbers of $\mathbf{X}_s^\mathsf{T}\boldsymbol{X}_s$ based on the scaled matrix, $\boldsymbol{X}_s$.

Create the scaled version of $\boldsymbol{X}$ by dividing each column by its sum. In matrix notation, column sums are $\mathbf{1}^\mathsf{T}\boldsymbol{X}$, and we need $\mathbf{D}$ to be a diagonal matrix of the inverses of the column sums. Then $\boldsymbol{X}_s = \boldsymbol{X}\boldsymbol{D} = \boldsymbol{X} \times \text{diag}(1/\mathbf{1}^\mathsf{T}\boldsymbol{X})$. From matrix theory, we know that the sum of eigenvalues is equal to the trace of a matrix. Since the diagonal of $\boldsymbol{X}_s^\mathsf{T}\boldsymbol{X}_s$ is all ones, the sum of eigenvalues is the number of columns ($k+1$ in using k predictors and an intercept), $\sum_{j=0}^{k} \lambda_j = \text{tr}(\boldsymbol{X}_s^\mathsf{T}\boldsymbol{X}_s) = k+1$. The condition number associated with the jth eigenvalue is

$$\eta_j = \sqrt{\frac{\lambda_{\max}}{\lambda_j}}$$

Large condition numbers occur when a particular λ_j is small relative to $\lambda_{\max}$, a rule of thumb is watch out for values over 30.

Variance Components of $\boldsymbol{X}$

To identify which columns of $\boldsymbol{X}$ are causing problems, use the Spectral (eigenvector)

Decomposition of a matrix:

$$\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X} = \boldsymbol{\Gamma}\boldsymbol{L}\boldsymbol{\Gamma}^{\mathsf{T}}$$

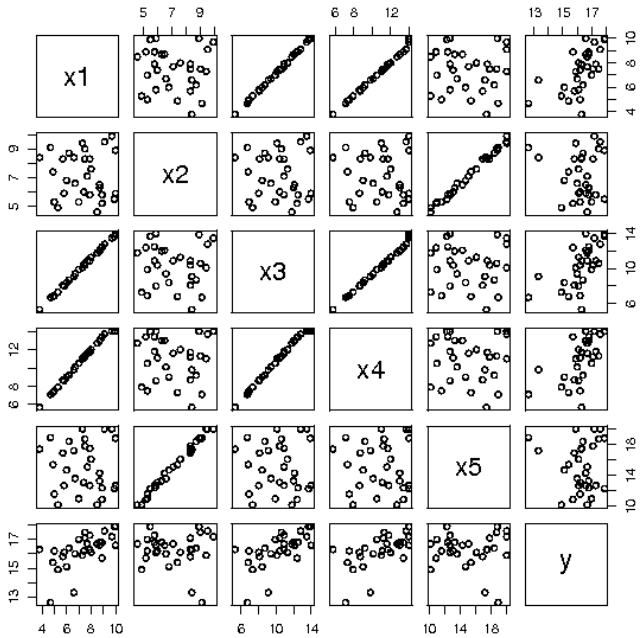
where $\boldsymbol{L}$ is the diagonal matrix of eigenvalues and the rows of $\boldsymbol{\Gamma}$ are the eigenvectors associated with their respective eigenvalues. We know $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathsf{T}} = \mathbf{I}$. Letting $\boldsymbol{\beta}^{(s)}$ be the coefficients under the scaled and centered model, $\text{cov}(\hat{\boldsymbol{\beta}}^{(s)}) = \sigma^2(\boldsymbol{X}_s\boldsymbol{X}_s^{\mathsf{T}})^{-1} = \sigma^2\boldsymbol{\Gamma}\boldsymbol{L}^{-1}\boldsymbol{\Gamma}^{\mathsf{T}}$, so the variance of the jth coefficient estimate is $\text{var}(\hat{\beta}_j^{(s)}) = \sigma^2\boldsymbol{g}_j^{\mathsf{T}}\boldsymbol{L}\boldsymbol{g}_j = \sigma^2\sum_{l=0}^{k}\lambda_l^{-1}\gamma_{jl}^2$. This suggests that we can break up the variance of the jth coefficient estimate into pieces of the form $\lambda_l^{-1}\gamma_{jl}^2$ for column l of the $\boldsymbol{X}_s$ matrix. The proportions of variance attributed to different variables are:

$$\phi_{jl} = \frac{\lambda_l^{-1}\gamma_{jl}^2}{\sum_l \lambda_l^{-1}\gamma_{jl}^2}$$

When a large condition number calls attention to a particular column, j, of $\boldsymbol{X}$ we can scan across the columns of ϕ_{lj} to see which other columns are most strongly associated with the jth column. However, SAS and the R functions I've built use the transpose, since they line up nicely, and we look in row j for large values. Start with the bottom row. If you see two large values in that row, those two variables are highly associated. As you move up, it can become harder to distinguish, you have to look for large values relative to $1 - \sum(\phi_{lj})$ where the sum is over the values in the lower rows.

Example 10.2 from Sen and Srivastava (1997)
Start with scatterplots:

```
> supervis <- read.table("data/e10.3",head
> pairs(supervis)
```



Variables denote: y = job satisfaction, x_1 = social contact with the supervisor, x_3 = supervisor's perceived interest, x_4 = support from supervisor, x_2 and x_5 measure the supervisors "drive."

```
> super.fullfit <- lm( I(-log(1-y/20)) ~ ., data=supervis)

> round(vif(super.fullfit),2)
(Intercept)          x1          x2          x3          x4          x5
       0.00      665.90      117.97      715.47      100.13      116.76
> round(tol(super.fullfit),5)
(Intercept)          x1          x2          x3          x4          x5
         NA     0.00150     0.00848     0.00140     0.00999     0.00856
```

```
> round(collin(super.fullfit),3)
  Eignvls    cond (Intercept)    x1    x2    x3    x4    x5
1   5.867   1.000       0.000 0.000 0.000 0.000 0.000 0.000
2   0.115   7.140       0.002 0.000 0.001 0.000 0.001 0.001
3   0.018  18.240       0.790 0.000 0.002 0.000 0.000 0.001
4   0.000 136.914       0.148 0.043 0.000 0.029 0.985 0.000 # << x1, x3, x4
5   0.000 174.879       0.053 0.002 0.934 0.002 0.000 0.944 # << x2 & x5
6   0.000 398.296       0.006 0.955 0.064 0.968 0.014 0.054 # << x1 & x3
> super.reducedfit <- update(super.fullfit, .~ I(x1+x4+x3) + I(x2 + x5))
> round(collin(super.reducedfit),3)
  Eignvls   cond (Intercept) I(x1 + x4 + x3) I(x2 + x5)
1   2.939  1.000       0.002           0.006      0.005
2   0.047  7.907       0.001           0.540      0.423
3   0.014 14.345       0.997           0.454      0.573 # no problem
> round(collin(super.fullfit,center=T),3)
  Eignvls         cond (Intercept)   x1    x2    x3    x4    x5
1   4.480 1.000000e+00           0 0.00 0.003 0.000 0.000 0.002
2   1.506 1.725000e+00           0 0.00 0.000 0.000 0.001 0.000
3   0.010 2.136900e+01           0 0.00 0.951 0.000 0.000 0.960 # << x2 & x5
4   0.003 3.831600e+01           0 0.03 0.001 0.042 0.992 0.000 # << x1, x3 & x4
5   0.000 1.056430e+02           0 0.97 0.046 0.958 0.007 0.038 # << x1 & x3
> round(collin(super.reducedfit, center=T),3)
  Eignvls         cond (Intercept) I(x1 + x4 + x3) I(x2 + x5)
1   2.552 1.000000e+00           0           0.998          0
2   0.448 2.388000e+00           0           0.002          1
3   0.000 1.994683e+15           1           0.000          0 # No problem
```

Another Example Looking again at the exercise from Sen and Srivastava (1997) referred to under added variable plots (8), we wish to explain auto ownership (AO = cars per person) based on the following variables:

- POP – population in millions,
- DEN – population density
- GDP – gross domestic production = per capita income in $US
- GASPRICE – gasoline price, US cents per liter
- GASUSE – consumption of gasoline per year in tonnes
- TRAINUSE – 1000 passenger km per person on trains and buses.

All data is from 1978. The countries represented include New Zealand, Canada, US, and European countries from UK to Turkey. Objective: to model relationships between the predictors and the response.

```
options ls=76 ps=76;
*proc print data=sasuser.auto;
run;
  title  "Raw data";
proc reg data = sasuser.auto;
  model  OWNERSHP =   POP  DEN   GDP  GASPRICE  GASUSE  TRAINUSE/
           collin;
 run;
```

```
Dependent Variable: OWNERSHP
                              Parameter Estimates
                         Parameter      Standard    T for H0:
    Variable  DF          Estimate         Error   Parameter=0    Prob > |T|
    INTERCEP   1          0.469733    0.09094306         5.165        0.0001
    POP        1          0.000591    0.00026569         2.224        0.0400
    DEN        1         -0.000107    0.00010939        -0.978        0.3417
    GDP        1          0.032236    0.00344529         9.357        0.0001
    GASPRICE   1         -0.004050    0.00127071        -3.188        0.0054
    GASUSE     1         -0.107598    0.02648398        -4.063        0.0008
    TRAINUSE   1         -0.071482    0.01748227        -4.089        0.0008

                           Collinearity Diagnostics
                        Condition  Var Prop  Var Prop  Var Prop  Var Prop
   Number  Eigenvalue       Index  INTERCEP  POP       DEN       GDP
        1     5.40778     1.00000    0.0004    0.0067    0.0080    0.0043
        2     0.72004     2.74051    0.0000    0.4700    0.0255    0.0015
        3     0.46785     3.39984    0.0018    0.1273    0.3565    0.0066
        4     0.21679     4.99442    0.0003    0.1059    0.5390    0.0182
        5     0.11912     6.73785    0.0064    0.0140    0.0105    0.8986
        6     0.05948     9.53509    0.0071    0.2381    0.0564    0.0612
        7     0.00895    24.58392    0.9840    0.0381    0.0040    0.0096

          Var Prop  Var Prop  Var Prop
   Number  GASPRICE  GASUSE    TRAINUSE
        1    0.0009    0.0018    0.0037
        2    0.0025    0.0080    0.0137
        3    0.0011    0.0380    0.0094
        4    0.0002    0.0569    0.2442
        5    0.0327    0.0026    0.0445
        6    0.1704    0.2824    0.6565
        7    0.7923    0.6103    0.0281
```

The highest condition number is 24, still below our threshold of 30, but it is strongly associated with the intercept. Notice how centering changes the output.

```
proc standard  data= sasuser.auto mean=0 out=scaled;
  var  POP  DEN   GDP  GASPRICE  GASUSE  TRAINUSE ;
 title "Centered";
proc reg ;
  model  OWNERSHP =   POP  DEN   GDP  GASPRICE  GASUSE  TRAINUSE/
          collin;
run;
```

```
                              Parameter Estimates
                         Parameter      Standard    T for H0:
    Variable  DF          Estimate         Error   Parameter=0    Prob > |T|
    INTERCEP   1          0.272667    0.01046545        26.054        0.0001
    POP        1          0.000591    0.00026569         2.224        0.0400
    DEN        1         -0.000107    0.00010939        -0.978        0.3417
    GDP        1          0.032236    0.00344529         9.357        0.0001
    GASPRICE   1         -0.004050    0.00127071        -3.188        0.0054
    GASUSE     1         -0.107598    0.02648398        -4.063        0.0008
    TRAINUSE   1         -0.071482    0.01748227        -4.089        0.0008

                           Collinearity Diagnostics
```

Number	Eigenvalue	Condition Index	Var Prop INTERCEP	Var Prop POP	Var Prop DEN	Var Prop GDP
1	2.63490	1.00000	0.0000	0.0139	0.0343	0.0108
2	1.34661	1.39882	0.0000	0.2186	0.0894	0.1608
3	1.00000	1.62324	1.0000	0.0000	0.0000	0.0000
4	0.85612	1.75435	0.0000	0.1192	0.1817	0.6330
5	0.58046	2.13057	0.0000	0.1719	0.5971	0.0591
6	0.33143	2.81958	0.0000	0.4763	0.0956	0.1022
7	0.25048	3.24337	0.0000	0.0000	0.0019	0.0343

Number	Var Prop GASPRICE	Var Prop GASUSE	Var Prop TRAINUSE
1	0.0423	0.0415	0.0443
2	0.0190	0.0063	0.0289
3	0.0000	0.0000	0.0000
4	0.0013	0.0055	0.0005
5	0.0015	0.0354	0.2586
6	0.4341	0.0383	0.4684
7	0.5019	0.8731	0.1992

Example 3 from the SAS PROC REG pages. The objective is to model the relationships of several predictors to Oxygen uptake.

```
*------------------Data on Physical Fitness------------------*
   | These measurements were made on men involved in a physical   |
   | fitness course at N.C.State Univ. The variables are Age      |
   | (years), Weight (kg), Oxygen intake rate (ml per kg body     |
   | weight per minute), time to run 1.5 miles (minutes), heart   |
   | rate while resting, heart rate while running (same time      |
   | Oxygen rate measured), and maximum heart rate recorded while |
   | running.                                                     |
   | ***Certain values of MaxPulse were changed for this analysis.|
   *-------------------------------------------------------------*;

proc reg data=sasuser.fitness graphics;
      model oxy=   age weight runtime rstpulse runpulse maxpulse
       /tol vif collin;
```

Variable	DF	Tolerance	Variance Inflation
INTERCEP	1	.	0.00000000
AGE	1	0.66101010	1.51283618
WEIGHT	1	0.86555401	1.15532940
RUNTIME	1	0.62858771	1.59086788
RSTPULSE	1	0.70641990	1.41558865
RUNPULSE	1	0.11852169	8.43727418
MAXPULSE	1	0.11436612	8.74384843

Collinearity Diagnostics

Number	Eigenvalue	Condition Index	Var Prop INTERCEP	Var Prop AGE	Var Prop WEIGHT	Var Prop RUNTIME
1	6.94991	1.00000	0.0000	0.0002	0.0002	0.0002
2	0.01868	19.29087	0.0022	0.1463	0.0104	0.0252
3	0.01503	21.50072	0.0006	0.1501	0.2357	0.1286
4	0.00911	27.62115	0.0064	0.0319	0.1831	0.6090
5	0.00607	33.82918	0.0013	0.1128	0.4444	0.1250
6	0.00102	82.63757	0.7997	0.4966	0.1033	0.0975
7	0.0001795	196.78560	0.1898	0.0621	0.0228	0.0146

Number	Var Prop RSTPULSE	Var Prop RUNPULSE	Var Prop MAXPULSE
1	0.0003	0.0000	0.0000
2	0.3906	0.0000	0.0000
3	0.0281	0.0012	0.0012
4	0.1903	0.0015	0.0012
5	0.3648	0.0151	0.0083
6	0.0203	0.0695	0.0056
7	0.0057	0.9128	0.9836

Which variables show up in the bottom row as inter-related?

Now Standardize (Or use collinoint to remove the intercept effect) and it changes:

```
proc reg data=sasuser.fitness graphics;
      model oxy=   age weight runtime rstpulse runpulse maxpulse
       /tol vif collinoint;
```

Variable	Label	DF	Pr > |t|	Variance Tolerance	Inflation
Intercept	Intercept	1	<.0001	.	0
age	age(years)	1	0.0322	0.66101	1.51284
weight	weight(kg)	1	0.1869	0.86555	1.15533
runtime	1.5 mile time(min)	1	<.0001	0.62859	1.59087
rstpulse	resting pulse rate	1	0.7473	0.70642	1.41559
runpulse		1	0.0051	0.11852	8.43727
maxpulse		1	0.0360	0.11437	8.74385

Collinearity Diagnostics

Number	Eigenvalue	Condition Index	Proportion of Variation: Intercept	age	weight
1	2.57492	1.00000	0	0.02557	0.01888
2	1.32776	1.39258	0	0.15710	0.02452
3	1.00000	1.60466	1.00000	0	0
4	0.92509	1.66836	0	0.00413	0.79356
5	0.74322	1.86133	0	0.25359	0.00043484
6	0.36868	2.64277	0	0.52060	0.13021
7	0.06033	6.53301	0	0.03902	0.03240

	Proportion of Variation			
Number	runtime	rstpulse	runpulse	maxpulse
1	0.01903	0.03467	0.01468	0.01426
2	0.21064	0.09347	0.00015260	0.00211
3	0	0	0	0
4	0.03816	0.01391	0.00575	0.00332
5	0.00453	0.44155	0.02167	0.01673
6	0.70484	0.41148	0.00429	0.00393
7	0.02280	0.00492	0.95346	0.95965

Is there a problem?

5.5 Smoothers

We have used smoothers in residual plots to look for curvature (in residuals versus fitted values) and trends (in spread-location plots). Smoothers can also be used in y versus x plots to look for curvature and trend and to visually compare a line or parametric curve to a flexible fit.

Many variations are available in R, and each has a way to make it more or less flexible. We'll first consider the case of a single predictor, x used to build a model for y.

Polynomials:
We use higher orders to attain more flexibility. All exhibit "tail flop" in that they go to $\pm\infty$ in the extremes. Venables and Ripley use an artificial data set to illustrate different smoothers. It consists of two columns: time in milliseconds, and acceleration – supposedly on a dummy's head as the motorcycle hits a barrier.

```
library(MASS); data (mcycle)
attach(mcycle)
par(mfrow=c(2,3)) ## first use lm to fit a linear (polynomial) model
plot(times,accel,main="Polynomial Regression")
lines(times,fitted(lm(accel~poly(times,3))))
lines(times,fitted(lm(accel ~ poly(times,6))),lty=3)
legend(35,-100,c("degree=3","degree=6"),lty=c(1,3),bty="n")
```

None of these fit this data well, which is part of the reason for using the dataset.

Kernel Smoothers
Imagine moving a window from left to right across the scatterplot plot. As each x value becomes the center of window, we could take the average y for points within the window as the value of the smoother. This "moving average" is an example of a kernel smoother, where the "kernel" used is just a flat uniform density over the entire window. By choosing the width of the window, one can vary the amount of smoothing. A triangular kernel could be used to down-weight points near the edge of the window and emphasize points close to the center. The kernel provides weights which are used to compute $\widehat{y}$ as a weighted average. Instead of using a window, one can pick a kernel like the Gaussian which has support on the entire real line. Choice of scale, σ, gives the desired degree of smoothing. In general, let $K(\cdot)$ be the kernel function.

$$\widehat{y}_i = \frac{\sum_{j=1}^{n} y_i K(\frac{x_i - x_j}{b})}{\sum_{j=1}^{n} K(\frac{x_i - x_j}{b})}$$

```
library(mgcv)
plot(times,accel,main="Gaussian Kernel Smoothers")
lines(ksmooth(times,accel,"normal",bandwidth=5))  ## use "box" for moving average
lines(ksmooth(times,accel,"normal",bandwidth=2),lty=3)
legend(35,-100,c("bandwidth=5","bandwidth=2"),lty=c(1,3),bty="n")
```

Lowess
Again we have a window which is moved across the x values, but this time the y values are used in a locally weighted regression which puts more weight on points with x near the center of the window, less weight on those further away. The default is to make $f = 2/3$ of the data fall into the window, and to use quadratic regression with "tricube" weights:

$$w_i = [1 - (\frac{c - x_i}{b})^3]^3$$

where c is the center of the window which is of width $2b$. Just to add a little more complexity, in the `loess()` version, one can also specify `family = "symmetric"` to use robust local regression, which down-weights points with large $|e_i|$. In R the `lowess(x,y)` function is used to add a fit to an existing plot. It returns an object with x and y attributes, so the `lines` function can use it directly. The `loess(y ~ x)` function takes a formula argument, and works more like `lm` and can work with more than one predictor. It is in the R library "mgcv".

```
plot(times,accel,main="Lowess Smoothers")
lines(lowess(times,accel))
lines(lowess(times,accel,f=.2),lty=3)
loess.fit <- loess( accel ~ times, data = mcycle, span = .2)
lines(times, fitted(loess.fit), lty=5)
legend(35,-100,c("default span","span = .2", "loess"),lty=c(1,3,5),bty="n")
```

The function `gam()` uses the abbreviation `lo()` for lowess, and allows one to input several predictors. In R, a `gam` function is available in the mgcv package, but uses smoothing splines, not lowess. (More on these later.)

Regression Splines

A natural spline or basis spline is just a regression model built on some special predictors. We divide the x axis up into pieces separated at "knot points". Within each piece we fit a cubic regression under the constraint that the functions must agree (no discontinuities) at the knots and the first and second derivatives must also agree. These constraints mean that we only have one free parameter between each pair of knots. Flexibility is varied by setting the number of knot points.

```
plot(times,accel,main="Regression Splines")
library(splines)
lines(times,fitted(lm(accel ~ ns(times,df=5))))
lines(times,fitted(lm(accel ~ bs(times,df=10))),lty=3)
```

Note that these are just fits using `lm`, but with a special $\boldsymbol{X}$ matrix built by the `bs` or `ns` functions.

Smoothing Splines

Instead of minimizing some sum of squares, smoothing splines minimize a more complex function, a compromise between flexibility (small SSE) and smoothness. The function is

$$\sum w_i[y_i - f(x_i)]^2 + \lambda \int [f''(x)]^2 dx.$$

The degree of smoothness is controlled by λ with smaller values meaning a more flexible fit. Instead of setting the value of λ, with penalty = .5, one may instead set the approximate degrees of freedom. With smoothing splines, each unique x_i is considered a knot point.

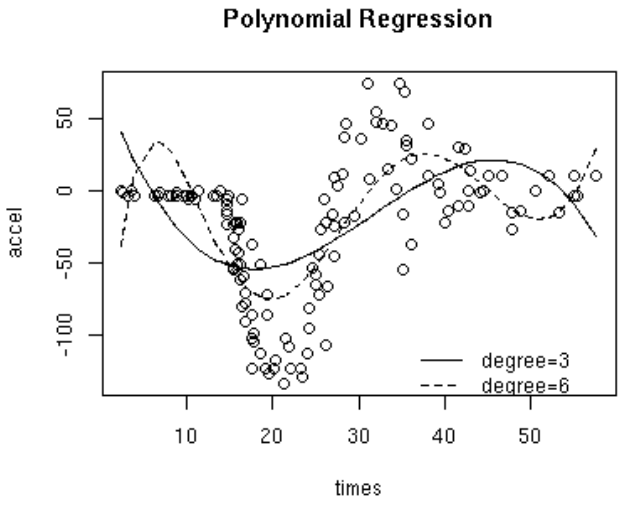
```
plot(times,accel,main="Smoothing Splines")
lines(smooth.spline(times,accel, df=5))
lines(smooth.spline(times,accel, df=10), lty=3)
```

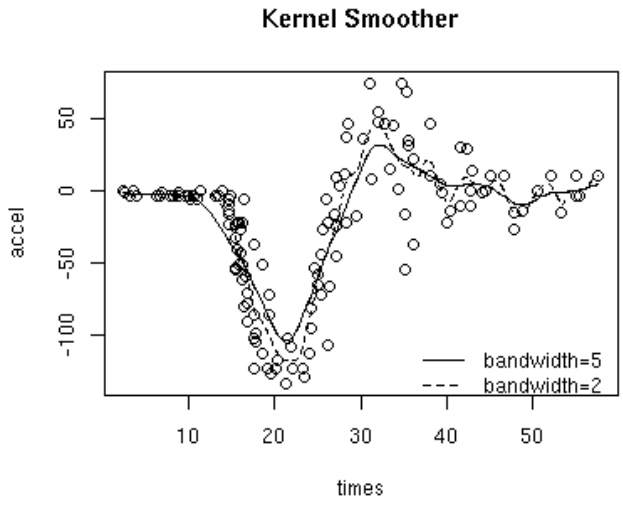
Super-smoother

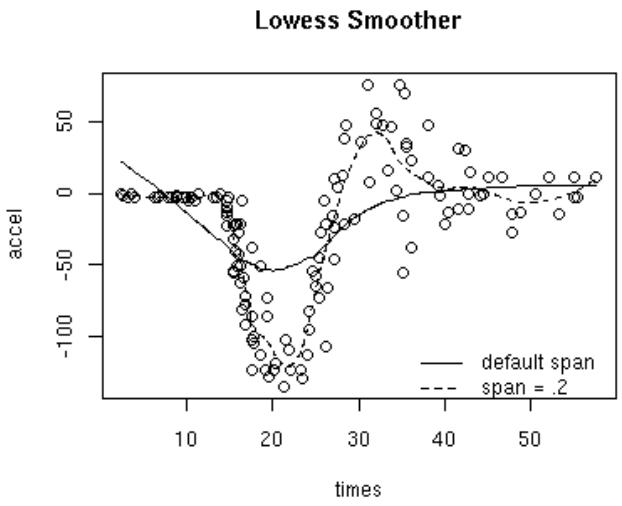
This technique also uses a window, now of $k/2$ points nearest x_i on each side – for a total of k points – and fits a line by least squares to that point. Three values of k are used, $n/2$, $n/5$, and $n/20$, and the best k is chosen for each point based on cross-validation. Cross-validation means that we take out one point at a time and look at the sum of squares residual for all points based on regressions which omit one point at a time. (PRESS) The advantage is that it is adaptive to the amount of smoothing needed. It can use a very smooth fit in one region, and a very flexible fit in another area.

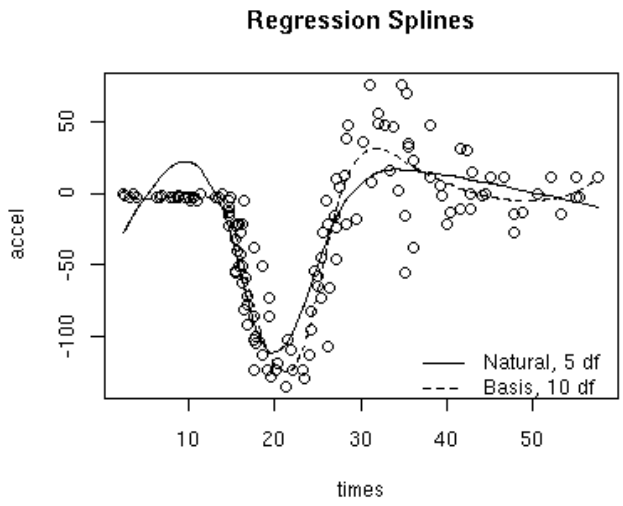
```
plot(times,accel,main="Super-Smoother")
lines(supsmu(times,accel), col=4)       ## in the stats package
lines(supsmu(times,accel,bass=3),lty=3)
```

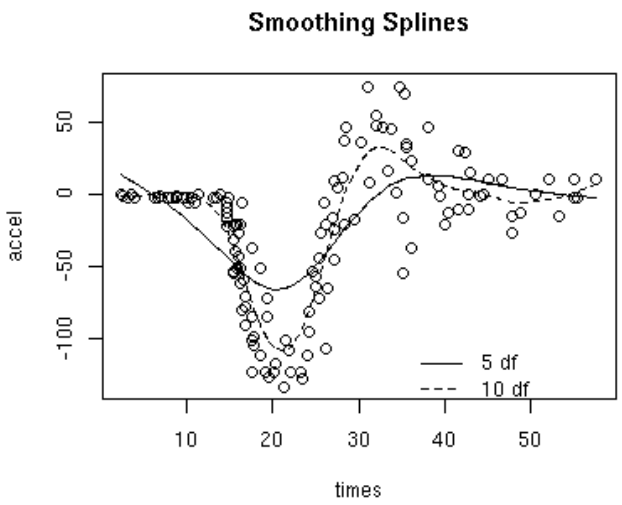
This data seems to have been chosen to show off supersmoother.

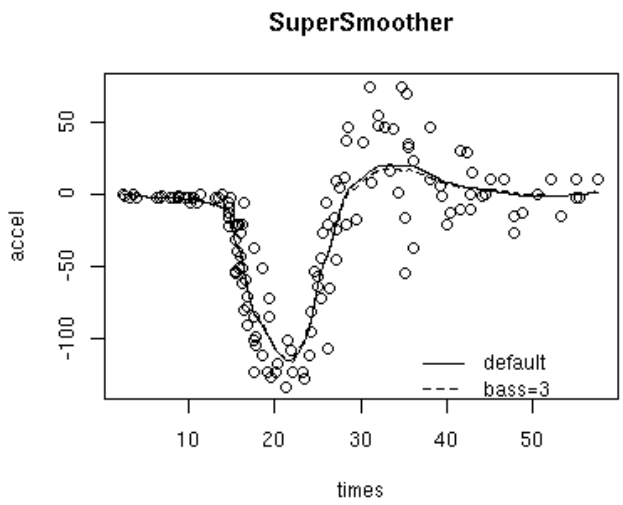












Problems with smoothers

There are several reasons why smoothers do not replace linear models.

- With the possible exception of regression splines, we do not get a concise representation of the fitted curve from the smoothing routines. It's hard to find the value of $\widehat{y}_h$ for an x_h which was not in the original data.
- These methods are sometimes called "nonparametric" smoothers. A better description is "highly parametric" because they, in effect, estimate a $\hat{y}$ for every x_i.
- We cannot use the smoother to answer simple questions like, "Is there an increasing trend?" Again regression splines are an exception, in that we can obtain their coefficient estimates. However, the coefficient estimates are hard to interpret, because the spline basis is quite artificial. One way to get predictions and standard errors would be to specify the knots in the initial call to `bs()` so that the same knots are used for the predictions.
- One can use the predict function with a `newdata =` argument. When `bs` or `poly` were used in earlier versions of R, predictions used to get thrown off because it tried to recompute $\boldsymbol{X}$ using the new points as well as the old ones. Now "safe prediction" routines handle those problems.
- **The curse of dimensionality**. When fitting a smooth curve to a single predictor using lowess, kernel smoothing or the super smoother, the computer has to estimate 100 or so $\widehat{y}$'s, but if we want the same accuracy with two predictors it will take $100^2 = 10000$ estimates. The number of evaluations rises exponentially with the number of predictors, and we should have that many more data points to estimate with good precision.
- The `gam()` functions in Splus and in R allow us to estimate surfaces and higher dimensions using loess or spline methods with commands like `gam(y ~ s(x , z))` which include interactions (but the interactions are not separable from the main effects). A potential problem is that some people use `gam` to build very fancy main-effects models without looking for interactions. If interactions are present, then we have to interpret main effects in light of the interaction, so the `gam` fit is suspect.

6 Other Topics in Modeling

6.1 Validation

When the purpose of model building is prediction, we need to look at how well the model we select does at predicting the response for new rows, $\boldsymbol{x}_{new}^{\mathsf{T}}$, with the same variables as the original data.

As an example, consider the `Boston` dataset contained in the MASS library in R. The data represent 506 towns around Boston, and for each we have 14 variables:

- `crim` per capita crime rate by town
- `zn` proportion of residential land zoned for lots over 25,000 sq.ft.
- `indus` proportion of non-retail business acres per town
- `chas` Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox` nitrogen oxides concentration (parts per 10 million)
- `rm` average number of rooms per dwelling
- `age` proportion of owner-occupied units built prior to 1940
- `dis` weighted mean of distances to five Boston employment centres
- `rad` index of accessibility to radial highways
- `tax` full-value property-tax rate per $10,000
- `ptratio` pupil-teacher ratio by town
- `black` 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- `lstat` lower status of the population (percent)
- `medv` median value of owner-occupied homes in $1000 – the response.

We can leave out 10% of the rows and fit a model, then see how well the model predicts the missing values:

```
> library(MASS)
> data(Boston)
> sampled.rows = sample(506,51)
> Boston1 = Boston[sampled.rows,]
> Boston2 = Boston[-sampled.rows,]
> fit1 = lm(medv ~ ., data = Boston2)
> summary(fit1)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  31.178952   5.085765   6.131 1.95e-09
crim         -0.114222   0.032955  -3.466 0.000580
zn            0.035328   0.013969   2.529 0.011786
indus         0.026162   0.060268   0.434 0.664428
chas          2.652470   0.836140   3.172 0.001618
```

```
nox         -16.133857   3.866116  -4.173 3.62e-05
rm            4.509385   0.422200  10.681  < 2e-16
age          -0.012469   0.013284  -0.939 0.348397
dis          -1.434154   0.199662  -7.183 2.93e-12
rad           0.296327   0.067178   4.411 1.29e-05
tax          -0.013631   0.003790  -3.596 0.000359
ptratio      -0.900220   0.130701  -6.888 1.96e-11
black         0.008291   0.002787   2.974 0.003096
lstat        -0.465713   0.052107  -8.938  < 2e-16

Residual standard error: 4.509 on 441 degrees of freedom
Multiple R-Squared: 0.7636,     Adjusted R-squared: 0.7566
F-statistic: 109.6 on 13 and 441 DF,  p-value: < 2.2e-16

> predictive.rsquared = function(fit, newdata, response.col)
    1- var(newdata[,response.col]-predict(fit,newdata))/
         var(newdata[,response.col])

> predictive.rsquared(fit1,Boston1,14)
[1] 0.5425014
```

Note that the percent of variation explained is 76% for the data used in fitting, but 54% for the withheld data. It is expected (though not true for every withheld sample) that predictive power is lower. Let's see what happens if we reduce the model.

```
> fit2 = step(fit1)
> summary(fit2)
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  31.519969   5.044470   6.248 9.75e-10
crim         -0.115500   0.032895  -3.511 0.000492
zn            0.036408   0.013786   2.641 0.008560
chas          2.660053   0.829944   3.205 0.001448
nox         -16.695679   3.575989  -4.669 4.02e-06
rm            4.401925   0.408641  10.772  < 2e-16
dis          -1.400334   0.187022  -7.488 3.83e-13
rad           0.293253   0.063981   4.583 5.96e-06
tax          -0.013024   0.003430  -3.797 0.000167
ptratio      -0.901832   0.128700  -7.007 9.10e-12
black         0.008123   0.002780   2.923 0.003649
lstat        -0.480189   0.048881  -9.824  < 2e-16

Residual standard error: 4.504 on 443 degrees of freedom
Multiple R-Squared: 0.763,      Adjusted R-squared: 0.7571
F-statistic: 129.7 on 11 and 443 DF,  p-value: < 2.2e-16

> predictive.rsquared(fit2,Boston1,14)
[1]  0.5515261
```

So R^2 still 76%, and the proportion of the variance of the withheld responses increased slightly to 55%.

There are problems in this model with multicollinearity.

```
>  round(collin(fit2, center=T),4)
Eignvls    cond    crim     zn   chas    nox     rm    dis    rad    tax ptratio  black  lstat
 7.0362 1.000000 0.4278 0.0330 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000  0.0000 0.0000 0.0019
 3.7097 1.377200 0.1486 0.4035 0.0002 0.0000 0.0000 0.0000 0.0000 0.0000  0.0000 0.0000 0.0012
 0.6529 3.282800 0.0594 0.0206 0.5918 0.0000 0.0000 0.0000 0.0000 0.0000  0.0000 0.0000 0.1623
 0.5558 3.558000 0.0725 0.0908 0.3519 0.0000 0.0000 0.0001 0.0000 0.0000  0.0000 0.0000 0.3710
 0.0339 14.40500 0.0554 0.2839 0.0268 0.0000 0.0000 0.5867 0.0004 0.0006  0.0000 0.0000 0.0435
 0.0081 29.50350 0.1953 0.0095 0.0002 0.0000 0.0000 0.0774 0.0811 0.0662  0.0000 0.0039 0.0454
 0.0020 59.67330 0.0118 0.0008 0.0006 0.0000 0.0008 0.0005 0.0041 0.0066  0.0000 0.9770 0.0040
 0.0008 94.89840 0.0199 0.0012 0.0078 0.0006 0.0668 0.0050 0.7299 0.6782  0.0000 0.0039 0.0629
 0.0004 125.5285 0.0048 0.0457 0.0004 0.0001 0.9002 0.0576 0.1626 0.1571  0.0010 0.0114 0.2671
 0.0001 225.1005 0.0023 0.0168 0.0199 0.5970 0.0042 0.2483 0.0081 0.0315  0.1479 0.0034 0.0186
 0.0001 316.1836 0.0021 0.0942 0.0005 0.4023 0.0279 0.0243 0.0137 0.0599  0.8511 0.0004 0.0222
```

I removed one variable at a time until I had removed all of `ptratio, nox, rm, black, rad, tax`. The model is then:

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 38.94521    1.03597  37.593  < 2e-16
crim        -0.14314    0.03585  -3.993 7.61e-05
zn           0.08778    0.01593   5.509 6.09e-08
chas         3.72381    1.03124   3.611 0.000339
dis         -1.33882    0.18617  -7.191 2.70e-12
lstat       -0.95324    0.04674 -20.395  < 2e-16

Residual standard error: 5.68 on 449 degrees of freedom
Multiple R-Squared: 0.618,      Adjusted R-squared: 0.6138
F-statistic: 145.3 on 5 and 449 DF,  p-value: < 2.2e-16
>  predictive.rsquared(fit3,Boston1,14)
[1] 0.5843242
```

Interesting: R^2 dropped to 62%, but the predictive R^2 increased to 58%! However, reducing the model does not always increase R^2. Let's try doing the sampling several times to see how much these values change:

```
for(i in 1:10){
       sampled.rows = sample(506,51)
       B1 = Boston[-sampled.rows,]; B2 = Boston[sampled.rows,]
       fitlm.AIC = step(lm(medv ~ ., data = B1),trace=0)
       print(c(summary(fitlm.AIC)$r.squared, predictive.rsquared(fitlm.AIC,B2,14)))}
[1] 0.728 0.812
[1] 0.770 0.482
[1] 0.738 0.754
[1] 0.745 0.665
[1] 0.734 0.797
[1] 0.776 0.227
[1] 0.747 0.688
[1] 0.740 0.748
[1] 0.744 0.712
[1] 0.731 0.832
```

In some cases predictive R^2 is 10% points higher than R^2 computed on the data used to fit the model. But overall, the mean of column 1 is 0.745, and the mean for column 2 is 0.671. The standard deviations are even more different: 0.016 for column one, and 0.185 for column two. Let's compare three different methods of model selection:

- Full model with 11 predictors
- Using AIC to stepwise reduce the full model
- Always using the 5 predictors which have little multicollinearity.

Ten random withdrawals yield:

	Full		AIC		5 predictors	
	mean	s.d.	mean	s.d.	mean	s.d.
R^2	0.746	0.011	0.746	0.011	0.613	0.007
pred.R^2	0.684	0.100	0.688	0.098	0.618	0.047

I ran the `for` loop again, reserving **half** the data for validation and got results:

	Full		AIC		5 predictors	
	mean	s.d.	mean	s.d.	mean	s.d.
R^2	0.745	0.037	0.744	0.037	0.606	0.027
pred.R^2	0.718	0.049	0.716	0.051	0.614	0.026

Lessons to be learned:

1. When the goal is prediction for new data, R^2 (for the data used to fit the model) is not a good indicator of the ability of the model to explain variability of new data. To really figure out the predictive precision of a model, you have to withhold some data.

2. If you really care **only** about prediction, and not about describing relationships between predictors and response, then collinearity is not a big problem, we might even include variables which have non-significant P-values and do not improve AIC.

6.2 Power Analysis

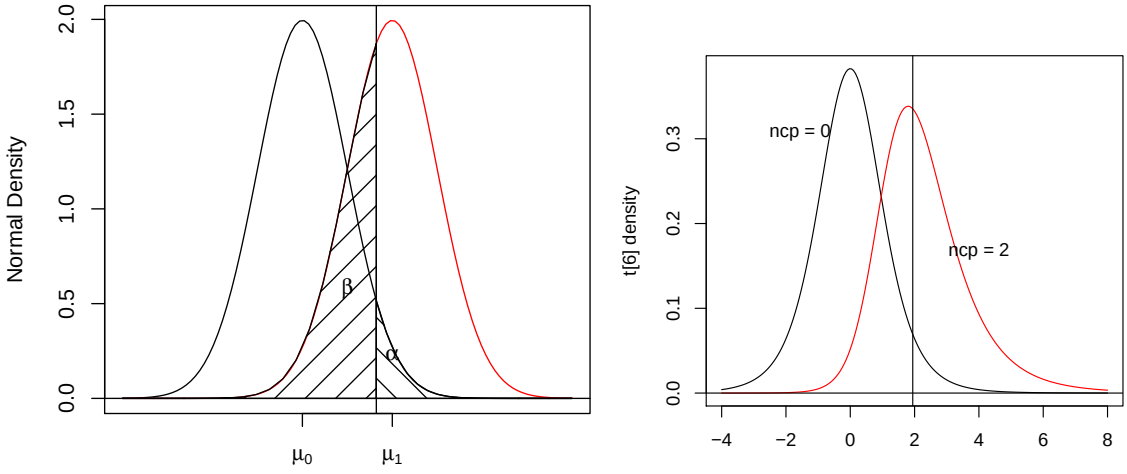
The Problem:
In planning a study which will use linear models, you are asked to answer one or both of two questions:

1. What is the probability that the study will find an important deviation from the null hypothesis to be statistically significant? (Compute the power.)

2. How large a sample size and what settings are needed to obtain a $1-\beta$ chance that the important deviation from H_0 will be found significant? (Given power, find sample size and X settings.)

Setting:
We are in the planning stage of a study, which can be viewed as a hypothesis testing situation (or building of a confidence interval if only one parameter is of interest) with fixed α, the significance level.

Simplest Case:
Linear model is $\boldsymbol{y} = \mu\mathbf{1} + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\boldsymbol{I})$ and we will test $H_0: \ \mu = \mu_0$ versus the $H_a: \ \mu > \mu_0$ alternative at significance level α. There is another value $\mu_1 > \mu_0$ which is important for some scientific reason. We want to be the study to detect the difference $\mu_1 - \mu_0$ with high probability. The plot indicates the rejection region to the right of the critical value with probability α under H_0, the probability of type II error, β to the left under $H_1: \ \mu = \mu_1$, and power $= 1-\beta$, the area to the right of the line under the H_1 curve.



Assume that we have data from a pilot study, which provides an estimate of σ. We will use $\widehat{\mu} = \overline{y}$ as our estimator and its sampling distribution, $\sqrt{n}(\overline{y} - \mu)/s \sim t(n-1)$. If n is given, we compute the power under $H_1: \ \mu = \mu_1$ as $P(\overline{y} > \mu_0 + t(.95, n-1)s/\sqrt{n} | \mu = \mu_1)$ $= P(T_{n-1} > t(.95, n-1) + (\mu_0 - \mu_1)/(s/\sqrt{n}))$. In R we can compute this as:

```
> s <- 2; n <- 100
>  power1 <-  1- pt( qt(.95,n-1), n-1, ncp =  .4/(s/sqrt(n)) )
[1]  0.6336
```

Note: We use the non-central t and function `pt` to compute the probability under the alternative. The rejection region is computed (with `qt`) under the null with $\sqrt{\phi} = 0$. Interpretation: The probability is 0.634 that we will be able to "detect" a difference of $\mu_1 - \mu_0 = .4$ when $s = 2$ and $n = 100$ using the $\alpha = .05$ significance level.

- What is the effect on power of changing α to 0.10? to 0.01?
- How does power change with the difference $\mu_0 - \mu_1$?
- How is power affected by s?

The other question, "How big a sample?", assumes that a power value is given and n must be computed. It's easiest to use the same computations as above, but to vary n. Suppose we want power = .90 when $\mu_1 - \mu_0 = .4$ and $s = .2$ in the above setting. Then take a range of n's from 110 to 200 and compute power for each:

```
> n <- seq(110,230,10)
> power2 <-  1 - pt( qt(.95,n-1), n-1, -(.8 - 1.2)/(s/sqrt(n)))
> round(rbind(n,power2*100),1) ## power in percentage now
   [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
n 110.0 120.0 130.0 140.0 150.0 160.0 170.0 180.0 190.0 200.0 210.0 220.0 230.0
     67  70.3  73.4  76.1  78.6  80.9   83  84.8  86.5    88  89.3  90.5  91.6
```

We have to use n of almost 220 to get 90% power. More precisely:

```
> n <- 211:220
> power2 <-  1- pt( qt(.95,n-1), n-1, -(.8 - 1.2)/(s/sqrt(n)))
> round(rbind(n,power2*100),1)
   [,1]  [,2]  [,3]  [,4]  [,5] [,6]  [,7]  [,8]  [,9] [,10]
n 211.0 212.0 213.0 214.0 215.0  216  217.0 218.0 219.0 220.0
   89.5  89.6  89.7  89.8  89.9  90.1  90.2  90.3  90.4  90.5
## make a plot:
> xx = seq(-4,6.5,len=100)
> plot(xx,dt(xx,215),type="l")
> lines(xx,dt(xx,215, ncp = .4 * sqrt(216)/s),type="l", col=2)
> abline(v = qt(.95,215))  ## with large df, both curves look symmetric
> abline(h=0)
```

We see that 216 or more units are required. There is now a function in R to do this for us.

```
> power.t.test(n=216,delta = .4, sd=2, power=NULL,type="one.sample",alt="one.sided")
     One-sample t test power calculation
              n = 216                    ## n was given,
          delta = 0.4
             sd = 2
      sig.level = 0.05
          power = 0.9006503              ## power was computed
    alternative = one.sided

> power.t.test(n=NULL, power = 0.9,delta = .4, sd=2, sig.level = 0.05,type="one",alt="one")
     One-sample t test power calculation
              n = 215.4562               ## n was computed
          delta = 0.4
             sd = 2
      sig.level = 0.05
          power = 0.9                    ## for specific power
    alternative = one.sided
```

To do the same in SAS, use the Analyst by choosing from the menus: Solutions, Analysis, Analyst. That pops up a new window where you choose menu options: Statistics, Sample Size, One sample t-test. In the window that pops up you can choose either N (given power) or Power (given N). Choose "Calculate N" at the top, and to mimic my analysis, set null

mean to .8, alternate mean to 1.2, standard deviation to 2, and alpha to .05. Click OK, and it will print 216 as the sample size needed. If you want to see how SAS does this, look at the "code" at the left side of the Analyst window. It has a "macro" of commands, much like a function in S.

Back to Linear Models:

The model we fit above **is** a linear model with a single parameter, μ. You should confirm that if $\boldsymbol{X} = \mathbf{1}$, then $\boldsymbol{X}^\mathsf{T}\boldsymbol{X} = n$, that $(\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y} = \sum y_i/n$, and that $\text{var}(\widehat{\mu}) = \sigma^2/n$ using the linear models formulae.

In simple linear regression we are often interested in the slope, β_1, often testing to see if $\beta_1 = 0$. Can we compute the power to detect a non-zero slope in this setting? We know that $\text{var}(\widehat{\beta}_1) = \sigma^2/\sum(x_i - \overline{x})^2$, which depends more on the range of x than on the sample size (and of course on the variance of the residuals about the line). Imagine, as an extreme case, that all x values are equal. What power would we have to detect a non-zero slope?

If we design an experiment where n and $SSX = \sum(x_i - \overline{x})^2$ are given, and an estimate of $\sigma^2 = \text{var}(\epsilon_i)$ is available, then the computation is no more difficult than the single mean case. We just need null and alternative values for β_1 and the α level.

```
power <- 1 - pt(qt(1-alpha, n-2), n-2, ncp=(null.beta1 - alt.beta1)*sqrt(SSX)/s)
```

Assuming the test is one-sided, $H_0: \beta_1 = \beta_{1(0)}$ versus $H_1: \beta_1 = \beta_{1(1)} > \beta_{1(0)}$. If you're wondering how we would ever know SSX ahead of time for an observational study, that is a very good question. There is a way around this obstacle we'll see later under multiple regression.

Two sample t:

Another linear model would be the ANOVA model with two or more groups:

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

If there are only two groups with equal variance and we are testing $H_0 : \tau_1 = \tau_2$ versus $H_a : \tau_1 \neq \tau_2$, then we have both t-test and F tests available: $t = (\overline{y}_{\cdot 1} - \overline{y}_{\cdot 2})/(s_p\sqrt{1/n_1 + 1/n_2})$ which has a $t(n_1 + n_2 - 2, \sqrt{\phi})$ distribution with $\phi = 0$ under H_0, and $\sqrt{\phi} = (\delta_1 - 0)/(s_p\sqrt{1/n_1 + 1/n_2})$ under a particular alternative: $H_a : |\tau_1 - \tau_2| = \delta_1$. The F test statistic is just the square of the t and has $F(1, n_1 + n_2 - 2, \phi)$ distribution. It's a special case of the distribution of an estimable contrast test statistic:

$$F^* = \frac{(\boldsymbol{K}^\mathsf{T}\hat{\boldsymbol{\beta}} - \boldsymbol{m})^\mathsf{T}[\boldsymbol{K}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{K}]^{-1}(\boldsymbol{K}\hat{\boldsymbol{\beta}} - \boldsymbol{m})/s}{MSE}$$

and has a $F(s, N-r, \phi)$ distribution, where $\phi = \frac{(\boldsymbol{K}^\mathsf{T}\boldsymbol{\beta} - \boldsymbol{m})^\mathsf{T}[\boldsymbol{K}^\mathsf{T}(\boldsymbol{X}^\mathsf{T}\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{K}]^{-1}(\boldsymbol{K}^\mathsf{T}\boldsymbol{\beta} - \boldsymbol{m})}{2\sigma^2}$ and $\phi = 0$ iff $H_0 : K^\mathsf{T}\boldsymbol{\beta} = \boldsymbol{m}$ is true. (Monahan equation 6.9 p 132). For the simple case of two groups, the contrast is only 1-dimensional ($s = 1$), and researchers should be able to come up with an estimate of a δ_1, which would be an "interesting" difference in means which we want to detect. Power computation in R assuming $\delta_1 > 0$:

```
> sqrt.phi <- delta1/(s * sqrt(1/n1 + 1/n2))## noncentrality parameter for t

> power <- 1 - pt( qt(1-alpha/2, n1 + n2 - 2), n1 + n2 - 2, ncp = sqrt.phi) +
                pt( qt(alpha/2, n1 + n2 - 2), n1 + n2 - 2, ncp = sqrt.phi)
```

```
   ## or use the non-central F distribution:
> power <- 1 - pf( qf(1-alpha, 1,  n1 + n2 - 2),  ## qf uses central F under H0
               1,  n1 + n2 - 2, ncp = sqrt.phi^2  )
Example:                        #^^^# ncp is 2*phi, Monahan's non-centrality parameter.
> delta1=.4;  n1=n2=200;  sqrt.phi <- delta1/(s * sqrt(1/n1 + 1/n2))
>  1 - pt( qt(1-alpha/2, n1 + n2 - 2), n1 + n2 - 2, ncp = sqrt.phi)
[1] 0.5140434
>  pt( qt(alpha/2, n1 + n2 - 2), n1 + n2 - 2, ncp = sqrt.phi)
[1] 3.824291e-05 ## added to  0.5140434 gives 0.5140816
>  1 - pf( qf(1-alpha, 1,  n1 + n2 - 2), 1,  n1 + n2 - 2, ncp = sqrt.phi^2  )
[1] 0.5140816
  ## the easy way -- seems to require equal sample sizes:
>  power.t.test(n=200, power = NULL,delta = .4, sd=2, sig.level = 0.05,type="two.sample",alt="two.sided")

     Two-sample t test power calculation
              n = 200
          delta = 0.4
             sd = 2
      sig.level = 0.05
          power = 0.5140434
    alternative = two.sided

 NOTE: n is number in *each* group
>  power.anova.test(groups=2, between.var=0.04,within.var=2,n=200)
     Balanced one-way analysis of variance power calculation
         groups = 2
              n = 200
    between.var = 0.04
     within.var = 2
      sig.level = 0.05
          power = 0.5140816       ## same computation using F test
 NOTE: n is number in each group
## Why use 0.04 for delta1 = 0.4?  They compute ncp as:
#  phi <- (groups - 1) * n * (between.var/within.var)
## var(c(0,delta1)) = 2(delta1/2)^2  = delta1^2/2
## so phi is  delta1^2/(s^2 * 2/n) as above
```

When we have more than two groups, there are many ways in which the group means can differ. For instance with 3 means, two could be equal and the third different (3 combinations there), or all three could differ in various orderings and amounts. It's just about impossible to elicit ahead of time, for an observational study, at least, what researchers want to detect.

For ANOVA (3 or more groups) and other cases of multiple regression, an approach which I have found useful, is to talk to the researchers in terms of R^2. In this setting, we might have some background variables, which are not of interest to the study, as well as the additional variables which are of interest. Example: Researchers in the MSU College of Nursing are studying people with chronic diseases (multiple sclerosis, diabetes, arthritis, cancer, disabilities). From previous studies of theirs and others, they know that "Social Support" has an effect on several responses of interest, such as "Quality of Life" or "Ability to Manage Symptoms". They wonder if there is an additional effect of "Spirituality" on the response(s). We have to ask:

1. What percentage of the variation in y could be explained by "background" variables, which all experts would agree have some effect on the response.

2. On top of the "background" variables in (1), what additional percentage of the variation in y is explained by the variable(s) of interest?

The researchers could tell me that, based on previous studies, that "Social Support" and demographic variables (gender in particular) will explain 30% of the variation in y, and if "Spirituality" explains an additional 3%, then the study will be noteworthy. They then want to know how large a sample size is needed to obtain power = 0.90 with $\alpha = 0.05$ significance level. In these cases, I've used methods from Cohen (1988) to compute power by defining the noncentrality parameter in terms of the R^2 values as

$$\phi = \frac{R^2(Y|A,B) - R^2(Y|A)}{1 - R^2(Y|A,B)}(u + v + 1)$$

where A denotes the background variables and B the variables of interest. In the "Spirituality" case, $R^2(Y|A,B) = 0.33$ and $R^2(Y|A) = 0.30$. We also will need u = numerator and v = denominator degrees of freedom for the F test. In this example, $u = 1$ (for the Spirituality effect) and $v = n - 6$ (assuming 5 df for the A variables, so $\phi = (.33 - .30)/(1 - .33) \times (1 + n - 6 + 1) = 0.04478(n - 4)$. I computed power for given n using the following function.

```
> powerf <- function(alpha,u,v,r2initial, r2final)
   1-pf(qf(1-alpha,u,v),u,v,(u+v+1)*(r2final - r2initial)/(1-r2final))
> n <- seq(160,300,10)
> rbind(n, round( powerf(.05, 1, n - 6, .30, .33),2))

n 160.0 170.0 180.0 190.0 200.00 210.00 220.00 230.00 240.0 250.00 260.00 270.00
   0.75   0.77   0.8   0.82   0.84   0.86   0.87   0.89   0.9   0.91   0.92

>  rbind(235:244, round( powerf(.05, 1, 235:244 - 6, .30, .33),3) )

 235.000 236.000 237.000 238.000 239.000 240.0 241.0 242.000 243.000
   0.893   0.894   0.896   0.897   0.898 0.899   0.9   0.902   0.903
```

They need to use at least 241 subjects to get the desired power.
The R package `pwr` Stéphane Champely, University of Lyon, France, takes this approach. Arguments to his `pwr.anova.test` function are k= number of groups, n = number of observations (per group), and f = effect size. For my example, $f = sqrt(0.04478) = .2116$.

```
> pwr.anova.test(f=sqrt(.04478),k=2,power=.9,sig.level=.05)
     Balanced one-way analysis of variance power calculation
              k = 2
              n = 118.2913         ## times 2 to get my answer (well, close)
              f = 0.2116129
      sig.level = 0.05
          power = 0.9
 NOTE: n is number in each group
```

Recommended reference: Cohen's book has been "panned" in Lenth (2001). Russell Lenth does not dispute the computations above, but objects vigorously to the use of what Cohen calls "small", "medium" and "large" effects sizes, saying that Cohen has done a disservice by pretending that there are universal sizes to effects. He also has an interesting web site with power computation applets at:
`http://www.stat.uiowa.edu/~rlenth/Power/`.

Other Approaches:

- **Simulation** One very good way to compute power, perhaps the only way in a study with multiple endpoints, (e.g. a clinical trial in which outcomes are checked at several time points. If one treatment is far superior to the others, the trial is stopped, and all subjects are switched to the preferred treatment.) is to run a simulation. Generate data from the distribution under H_a, compute a test statistic and a p-value. Repeat several hundred times, and check: in what proportion of the runs is the null hypothesis rejected. The process forces us to really think about the settings of the problem. An example of the anova model from above: $H_a: \ |\tau_1 - \tau_2| = .4, n_1 = n_2 = 200, \sigma = 2$.

```
Ftests = rep(0,500)
group = gl(2,200)   ## factor with 2 levels, each used 200 times
for(i in 1:500){
  y = rnorm(400, mu=rep(c(0,.4),each=200), 2)
  Ftests[i] = summary(lm(y ~ group))$fstatistic[1]
 }
table(cut(Ftests,c(0.0,qf(.95,1,398),100)))
  (0,3.86] (3.86,100]
       228        272##  54.4% of the 500 runs have rejected H_0.  #$
```

- **Power Computation on the Web**? A google search for "statistical power calculator" gets 67,000 hits.

- The Russel Lenth site includes a link to UnifyPow – a set of macros for SAS by Ralph O'Brien of the Department of Biostatistics and Epidemiology, Cleveland Clinic Foundation. These are included in SAS 9.1 as procedures POWER and GLMPOWER. I've not looked at how they work. SAS now also provides PSS (power and sample size) as a web applet. We don't have it installed.

- In R the following functions are available:

  `package:stats` power.t.test, power.prop.test, power.anova.test

  `package:pwr` pwr.t.test, pwr.2p.test (proportions), pwr.2p2n.test ($n_1 \neq n_2$), pwr.anova.test

  `package:Hmisc` by Frank Harrell: bpower (binomial), ciapower (exponential surval models), cpower (Cox survival models), popower (Poisson), spower (general survival models)

  `package:powerpkg` for some specific genetic tests.

  and many more. (2396 hits in searching for power within functions)

Danger:

It is not uncommon in some fields to see "post hoc" power analyses. These are **bad**, don't ever do them! Apparently researchers use them when they fail to reject a null hypothesis hoping that they can salvage some meaning. Observed power will be high when the null is rejected, and low otherwise. As Lenth says, "The automotive analogy is that if your car made it to the top of the hill, it was powerful enough; if it didn't it was not powerful enough." Observed power does not award super-significance as, "The results were significant AND the power was high." And cannot be used as an excuse, "The results were not significant, but that was because the power of the test was so small." It is fine

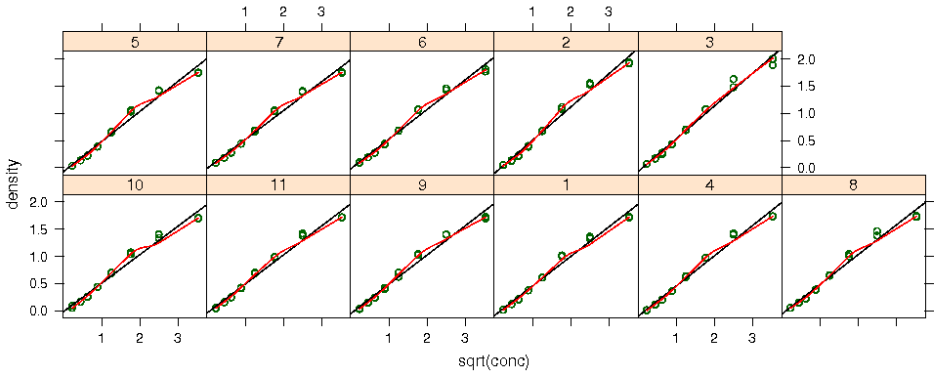
to use estimates of variance from one study to estimate sample size and power for a new study, but one cannot snoop at the data in midstream in order to compute the number of additional subjects needed to obtain significance.

Testing for Equivalence:
A hot topic in the pharmaceutical industry is, "How do we know if the new formulation is equivalent in efficacy to the old?" Generic drugs do not have to pass the same hurdles that a new drug must jump over, they only have to be shown equivalent. This turns the usual hypothesis test for significant differences on its head, because the researcher hopes to accept the null hypothesis of no difference in true mean efficacy. To untangle this one, we certainly need to consider power (by collecting only a few data points we would always expect to fail to reject the null) and often the problem of measurement, since differences could be so small as to be obscured by the limits of detection.

Thomas Oakberg did his writing project on this topic in 2003. There is certainly more that could be done with this area. R `package:MBESS` has `power.equivalence.md` function.

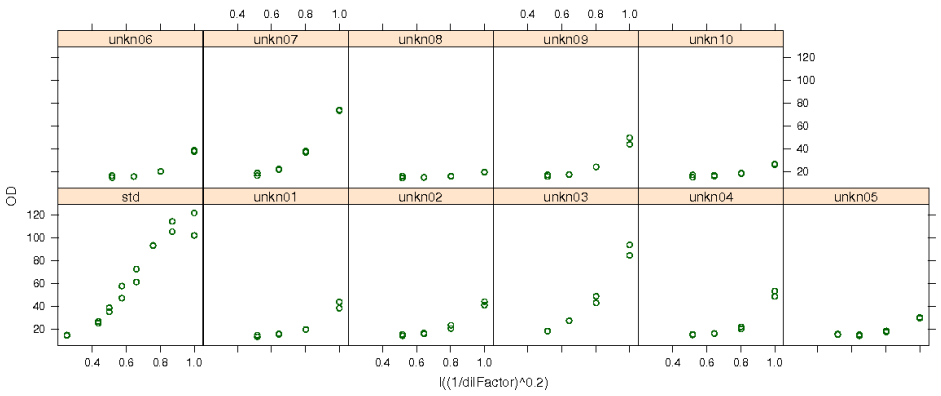
6.3 Calibration

In tests to determine the unknown concentration of a substance (ELISA and others), scientists use small quantities of "Standards" of known concentration, along with the quantities of interest. The built-in R dataset `DNase` is an example of typical output for the standards.




```
xyplot(density ~ sqrt(conc)|Run, DNase,panel=function(x,y){
                    panel.xyplot(x,y)
                    panel.lmline(x,y)
                    panel.loess(x,y,col=2)})
```

The samples are typically diluted (as well as the standards) in order to provide some responses which are in the same range as the densities of the standards. Gelman et al. (2003) show data which includes both standards and unknowns.



These standards include two samples with true zeroes for concentration, which show up in the bottom left corner. They will not be used for determining the line of best fit. The other panels show an x variable which is really unknown, and could slide in either direction. The values `1/dilFactor` just give relative concentration within each panel.

Frequentist Approach

Analyses typically use nonlinear models, for instance the four-parameter logistic regression model which estimates two horizontal asymptotes ($\beta_1,\ \beta_2$), a concentration for which the response is halfway between the asymptotes (β_3), and a "slope" (β_4) which determines how quickly the curve changes (at the midpoint).

$$g(x, \boldsymbol{\beta}) = \beta_1 + \frac{\beta_2}{1 + \exp[-\beta_4(x - \beta_3)]}$$

Nonlinear models are more difficult to fit than linear ones, and we don't cover them in this course. Instead we'll "suppose" that some transformation on x, the concentration, linearizes the relationship between the density and concentration. For the DNase data above, square root does fairly well, though we can still see lack of fit in the middle and upper areas. For Gelman's dilution data the standard concentrations to the .2 power (fifth root) are linearly related to response.

In calibration, we use a new response value, y_{new} to estimate an x value through the equation:

$$y_{new} = \widehat{\beta}_0 + \widehat{\beta}_1\widehat{x}. \text{ Solving for } x \text{ gives: } \widehat{x} = (y_{new} - \widehat{\beta}_0)/\widehat{\beta}_1$$

Using the multivariate delta method,

$$\text{Var}(\widehat{x}_n) \approx \text{Var}(y_{new})\widehat{\beta}_1^{-2} + \left[\frac{\partial g}{\partial \hat{\beta}_0} \ \frac{\partial g}{\partial \hat{\beta}_1}\right] \text{Var}(\widehat{\boldsymbol{\beta}}) \begin{bmatrix} \frac{\partial g}{\partial \hat{\beta}_0} \\ \frac{\partial g}{\partial \hat{\beta}_1} \end{bmatrix}$$

Where $g = \widehat{\beta}_0/\widehat{\beta}_1$, which is easy to differentiate: $\frac{\partial g}{\partial \hat{\beta}_0} = \widehat{\beta}_1^{-1}$, and $\frac{\partial g}{\partial \hat{\beta}_1} = -\widehat{\beta}_0\widehat{\beta}_1^{-2}$. $\text{Var}(\widehat{x})$ then simplifies to

$$\sigma^2 \left(1 + \begin{bmatrix} 1 & -\widehat{\beta}_0/\widehat{\beta}_1 \end{bmatrix} \text{Var}(\widehat{\boldsymbol{\beta}}) \begin{bmatrix} 1 \\ -\widehat{\beta}_0/\widehat{\beta}_1 \end{bmatrix}\right) / \widehat{\beta}_1^2$$

```
> dilution.fit <-  lm(OD ~ I(dilFactor^-.2), data = dilution,
                          subset = sample=="std" & dilFactor < 100)
> summary(dilution.fit)$coef
                  Estimate Std. Error   t value     Pr(>|t|)
```

```
(Intercept)       -43.30312   9.216014 -4.698682 5.154701e-04
I(dilFactor^-0.2) 166.51954  12.965384 12.843394 2.260852e-08
> MASS::boxcox(dilution.fit)
> y.new <-  dilution$OD[1:80]
> beta.hat <- coef(dilution.fit)
> sigma.hat <-  summary(dilution.fit)$sigma
> Var.hat <-  summary(dilution.fit)$cov.unscaled
> Var.hat
                  (Intercept) I(dilFactor^-0.2)
(Intercept)          1.016909         -1.379461
I(dilFactor^-0.2)   -1.379461          2.012642
> xhat <- (y.new - beta.hat[1])/ beta.hat[2]
>  added.var <- c(1,   beta.hat[1]/ beta.hat[2]) %*% Var.hat %*%  c(1,  - beta.hat[1]/ beta.hat[2])
>  se.xhat <- sigma.hat*sqrt( 1 + added.var)/ beta.hat[2]
> se.xhat
   0.09298506

> xhat <-  xhat * dilution$dilFactor[1:80]        ## adjust for dilution rates
> se.xhat <-  se.xhat * dilution$dilFactor[1:80]  ## scale up
> weights <- se.xhat^-2
> weights
 [1] 115.6574506 115.6574506  12.8508278  12.8508278   1.4278698   1.4278698
 [7]   0.1586522   0.1586522 115.6574506 115.6574506  12.8508278  12.8508278
> conc <-  1:10                                   ##  placeholders
> for(i in 1:10) conc[i] <-  weighted.mean(xhat[1:8 +(i-1)*8], weights[1:8 ] )
> conc                                            ## in x^.2 scale
 [1] 0.4908453 0.5000541 0.7639471 0.5446567 0.4304163 0.4762094 0.6762513
 [8] 0.3738603 0.5247389 0.4124500
> se.conc <-  sqrt(tapply( weights,  dilution$sample[1:80],sum)) /
                  tapply( weights,  dilution$sample[1:80],sum)
> round(se.conc,5)
  std  unkn01  unkn02  unkn03  unkn04  unkn05  unkn06  unkn07  unkn08  unkn09  unkn10
   NA 0.06199 0.06199 0.06199 0.06199 0.06199 0.06199 0.06199 0.06199 0.06199 0.06199
> confIntervals <- rbind(lower=conc -2*se.conc[-1], conc, upper=conc +2 * se.conc[-1])^5
> round( confIntervals,4)
      unkn01 unkn02 unkn03 unkn04 unkn05 unkn06 unkn07 unkn08 unkn09 unkn10
lower 0.0066 0.0075 0.1073 0.0132 0.0027 0.0054 0.0514 0.0010 0.0103 0.0020
conc  0.0285 0.0313 0.2602 0.0479 0.0148 0.0245 0.1414 0.0073 0.0398 0.0119
upper 0.0879 0.0946 0.5520 0.1337 0.0524 0.0779 0.3282 0.0306 0.1149 0.0444
```

The above illustrates one way to do calibration, but I don't like the way using delta method makes all standard errors the same. I prefer to think of the Working-Hotelling band which spreads out more as we get away from the mean of the x's.

Problems:

- Fixed standard errors of new x values.
- Did not take into account the increasing variance as OD increases.
- detection limits.

7 Biased Estimation and Modern Methods

When we looked at problems of multicollinearity, we had to

- Diagnose the problem,
- Solve by removing or combining variables.
- Or: think in terms of building a reasonable model or comparing a set of reasonable models with some criterion.

Other approaches:

- Find a model with smaller variances of $\widehat{\boldsymbol{\beta}}$. Give up some bias to obtain smaller variances. (Principal Components Regression)
- Change the criterion from BLUE to minimizing MSE. (Ridge Regression and Shrinkage estimation)

These methods are controversial, and involve "art" as well as "science".

7.1 Principal Components Regression

The idea is to replace $\boldsymbol{X}$ with $\boldsymbol{U}$, of smaller dimension, but containing most of the information carried by $\boldsymbol{X}$. Start by scaling each column of $\boldsymbol{X}$, so that all are in "standard deviation" units. Decompose $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ into eigenvectors and eigenvalues:

$$\boldsymbol{X}^\mathsf{T}\boldsymbol{X} = \boldsymbol{\Gamma}\boldsymbol{D}_\lambda\boldsymbol{\Gamma}^\mathsf{T}$$

where the diagonal of $\boldsymbol{D}_\lambda$ is sorted from largest to smallest eigenvalue. Rewrite the model as

$$\boldsymbol{y} = \boldsymbol{X}_s\boldsymbol{\Gamma}\boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{\beta}_s + \boldsymbol{\epsilon} \text{ or } \boldsymbol{y} = \boldsymbol{U}\boldsymbol{\delta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{\Gamma}$ and $\boldsymbol{\delta} = \boldsymbol{\Gamma}^\mathsf{T}\boldsymbol{\beta}$. This has not really changed the model or the fitted values. The PC step is to use a reduced $\boldsymbol{U}$ matrix with fewer columns than in $\boldsymbol{X}$. Because $\boldsymbol{U}^\mathsf{T}\boldsymbol{U}$ is diagonal, its columns are LIN, and removing some columns does not affect the estimates in the other columns. If the information discarded is trivial, then we have a simpler model which gives about the same predictions, and variances of the coefficient estimates are reduced. The art of PC is in interpreting the columns of $\boldsymbol{U}$. Here are the eigenvectors of $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ in the supervisor ratings data.

```
> options(digits=3)
> XtXeigen <- eigen(crossprod(scale(as.matrix(supervis[,-6]),T,T)))
> XtXeigen
$values
[1] 87.0915 57.5612  0.1959  0.1298  0.0216


$vectors
        [,1]    [,2]     [,3]    [,4]    [,5]
[1,]  0.5746 -0.0579  0.43350  0.0675  0.6884
[2,] -0.0792 -0.7023  0.00565 -0.7037  0.0724
[3,]  0.5743 -0.0637  0.38149 -0.0719 -0.7179
[4,]  0.5741 -0.0537 -0.81629 -0.0142  0.0317
[5,] -0.0640 -0.7046 -0.01349  0.7034 -0.0663
```

The first column is a weighted average of $\boldsymbol{x}_1, \boldsymbol{x}_3$, and $\boldsymbol{x}_4$, while the second puts loadings on $\boldsymbol{x}_2$ and $\boldsymbol{x}_5$, then we have a nicely interpretable reduced model. Here is the LRT comparison of three models using 1, 2, or 3 columns of $\boldsymbol{X\Gamma}$.

```
> Xs = scale(supervis[,-6],T,T)  ## column 6 is y
> superPC1.fit = lm(y~ I(Xs %*% XtXeigen$vectors[, 1]))
> superPC2.fit = update(superPC1.fit, .~ .+ I(Xs %*% XtXeigen$vectors[, 2]))
> superPC3.fit = update(superPC2.fit, .~. + I(Xs %*% XtXeigen$vectors[, 3]))
> anova(superPC1.fit,superPC2.fit,superPC3.fit)
Analysis of Variance Table
  Res.Df     RSS Df Sum of Sq      F Pr(>F) ||  R^2     adj.R^2
1     28 25.0065                           ||   0.375    0.353
2     27 24.9816  1    0.0249 0.0260 0.8731 ||   0.376    0.330
3     26 24.9109  1    0.0706 0.0737 0.7881 ||   0.378    0.306
> c( summary( superPC3.fit)$r.squared,  summary( superPC3.fit)$adj.r.squared)
```

Conclusion: One principal component is really all we need. We could do about as well by using a sum of x1, x3, and x4:

```
> sum.X1.X3.X4 = supervis[,1] + supervis[,3] + supervis[,4]
> anova( update(superPC1.fit, .~ sum.X1.X3.X4),superPC1.fit)
Analysis of Variance Table

  Res.Df   RSS Df Sum of Sq F Pr(>F)
1     28 25.02
2     28 25.01  0      0.01
```

In general, principal component regression has one big drawback: There is no guarantee that the correlation between $\boldsymbol{X}$ and $\boldsymbol{y}$ shows up in the first principal components. It's possible that it appears in the columns associated with the smallest eigenvalues. For more details, see Cook (2007). The art involved is in interpreting the loadings.

7.2 Ridge Regression

Earlier we saw that the problem of multicollinearity is that $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ is "close" to singular. One way to remove a singularity is to add in a non-singular matrix. The sum of the two matrices will be non-singular. Ridge regression takes this approach, usually using the centered and scaled version of $\boldsymbol{X}$. An estimate of $\boldsymbol{\delta}$ in

$$\boldsymbol{y} = \boldsymbol{Z}_{(s)}\boldsymbol{\delta} + \boldsymbol{\epsilon}$$

is (for any $c > 0$ you want to pick):

$$\widehat{\boldsymbol{\delta}}_c = [\boldsymbol{Z}_{(s)}^\mathsf{T}\boldsymbol{Z}_{(s)} + c\boldsymbol{I}]^{-1}\boldsymbol{Z}_{(s)}^\mathsf{T}\boldsymbol{y}.$$

One typically chooses c to be close to 0.

I like §12.3.1 in Sen & Srivastava where they describe three ways to think about ridge regression:

1. Ridge regression adds a bit of noise to the original $\boldsymbol{X}$ matrix, spreading it out to give a bigger footprint in the parameter space, hence more stability. The noise is designed to be orthogonal to $\boldsymbol{X}$ and to $\boldsymbol{y}$.

2. We could perform augmented OLS regression, appending $\mathbf{0}$'s to $\boldsymbol{y}$ and $\boldsymbol{W}$ to $\boldsymbol{Z}_{(c)}$ such that $c\boldsymbol{I} = \boldsymbol{W}^\top\boldsymbol{W}$.

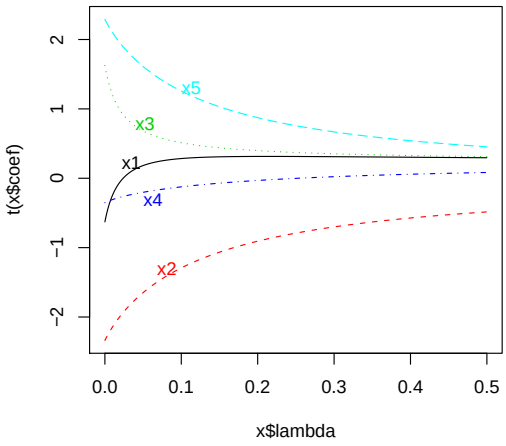
$$\widehat{\boldsymbol{\delta}} = \left[\begin{pmatrix}\boldsymbol{Z}^\top & \boldsymbol{W}^\top\end{pmatrix}\begin{pmatrix}\boldsymbol{Z}\\ \boldsymbol{W}\end{pmatrix}\right]^{-1}\begin{pmatrix}\boldsymbol{Z}^\top & \boldsymbol{W}^\top\end{pmatrix}\begin{pmatrix}\boldsymbol{y}\\ \mathbf{0}\end{pmatrix} = [\boldsymbol{Z}^\top\boldsymbol{Z} + \boldsymbol{W}^\top\boldsymbol{W}]^{-1}\boldsymbol{Z}^\top\boldsymbol{y}$$

3. The coefficient vector is constrained to lie on a sphere. When two columns of $\boldsymbol{Z}$ are highly collinear, one could have a huge positive coefficient, the other a hugely negative one, and they would cancel each other out. The constraint keeps odd things like huge coefficients from occurring.

Another interpretation: if we take a Bayesian approach, and use as prior: $\boldsymbol{\beta} \sim N(\mathbf{0}, c\boldsymbol{I})$, then the ridge regression solution is the mode (and mean) of the posterior.

The `lm.ridge` function in the MASS R library does ridge regression.

```
> Z <-  scale(supervis[,-6],T,T)
> require(MASS)
> lm.ridge( y ~ Z, data = supervis)
                  Zx1         Zx2         Zx3         Zx4         Zx5
16.2666667 -0.6397064 -2.3774774  1.6528936 -0.3578020  2.3296891
> plot(lm.ridge( y ~ Z, data = supervis,lambda = seq(0,0.5,0.001)))
> select(lm.ridge( y ~ Z, data = supervis,lambda = seq(0,0.1,0.0001)))
modified HKB estimator is 0.2120972
modified L-W estimator is 5.351978
smallest value of GCV  at 0.1
```



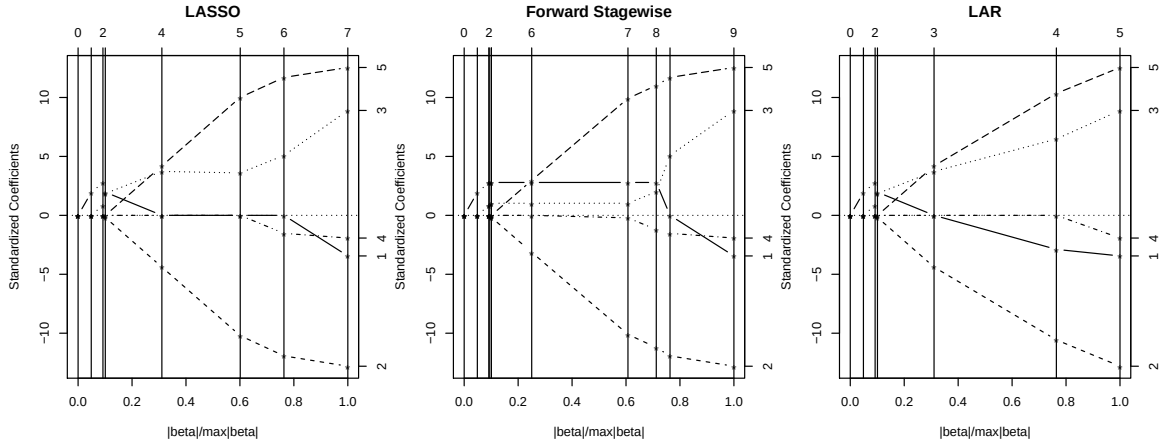
Note how the coefficient estimates change with c. To choose c, we want to take the smallest c for which components of the estimated coefficient vector are stable (for larger values of c. The HKB estimator above seems reasonable, the L-K estimator seems too large for this data.

7.3 LARS and Lasso

A good explanation is Efron et al. (2004), also available as a pdf:
`http://www-stat.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf`
Consider added variable plots where we projected potential predictors into the null space

of the current model matrix and looked to see which predictor was most highly correlated with $\boldsymbol{y}$. That predictor was added to the model. The idea behind LARS is that we should not be so "greedy", but rather take a step toward inclusion of such a variable.

How do you "partially" add a variable? They constrain the absolute sum of the coefficient estimates, so that the addition of another predictor means that coefficients of predictors already in the model must shrink toward zero. In the pictures below, each coefficient has a trace which is labeled on the right axis (e.g. $1 = x_1$). The x axis represents increasing complexity, culminating in the least squares fit at $\beta/\max(\beta) = 1$.




```
require(lars)
 y <- supervis[,6] - mean(supervis[,6])
 Z <-  scale(supervis[,-6],T,scale = apply(supervis[,1:5],2,sd)*sqrt(29) )
 sum(abs( coef(lm(y ~ Z - 1)))) ## 13.62
 par(mfrow=c(1,3))
 plot(lars(Z,y))
 plot(lars(Z,y,type="lar"))
 plot(lars(Z,y,type="for"))
```

Lasso

Standardize all predictors to have mean zero and length $1 = ||\boldsymbol{x}_i||$. Recenter response $\boldsymbol{y}$ to have mean 0. The goal is to minimize $Q(\hat{\boldsymbol{\beta}}) = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})$ subject to the constraint $T(\hat{\boldsymbol{\beta}}) = \sum |\hat{\beta}_j| < t$. In the leftmost plot, as we move to the right, t increases to 7.36, its value for the OLS coefficient estimate. When $t = 7.36$, there is effectively no constraint, and OLS is optimal. For smaller t, the coefficients are shrunk toward zero, which is intended to improve prediction accuracy. For values of $t < 7.36$, some subset of the predictors have non-zero coefficient estimates. For example, the vertical line at .6 indicates that 3 terms are in the model, x_2, x_3, x_5 and x_4 will join next (I don't know why it's labeled 5).

The stagewise algorithm, like the added variable plots, is based on the correlations between the columns of $\boldsymbol{X}$ and the residuals. Starting with $\hat{\boldsymbol{\mu}} = \boldsymbol{0}$, and proceeding by many small increments, they find the column of $\boldsymbol{X}$ with the greatest absolute correlation with the residuals, and increase that predictor's coefficient. The rightmost plot shows how that works for the supervisor data. Again, the right extreme is the OLS estimate.

LARS stands for "Least Angle Regression" and includes Lasso and stagewise as special cases. Start with all predictors having coefficient 0, and find the predictor with the most correlation with the residuals. Give this predictor a big enough coefficient that some other variable is just as correlated with the residuals as the first predictor. Put more weight on

each of these predictors in such a way as to make the residual vector have an equal angle with each predictor, increasing until a third predictor has the same absolute correlation as the first two. Give all three additional weight – again keeping the residual vector equal in angle to all three predictors – until a fourth variable is equally correlated. LARS was created to speed up the stagewise procedure which takes many small steps while LARS proceeds directly to the next splitting point.

Which of the models do we select in the end? This is an area for further research, but Efron et al. (2004) suggest using a C_p criterion.

Utility

These methods are intended to improve upon stepwise regression methods and are efficient computationally. They may not work well with highly collinear predictors. The authors suggest that they might be used as an exploration of the complexity of the data. After this exploration, statisticians and content area experts could decide on a reasonable model.

7.4 Boosting

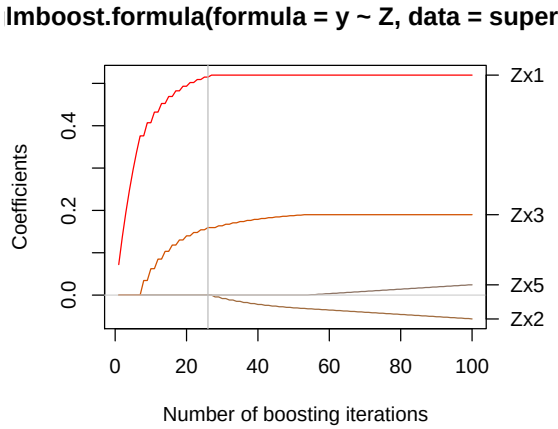
Boosting (see Buhlmann and Hothorn (2007)) is used heavily by the machine learning (data mining) community where often huge numbers of predictors are available. The goal is to reduce prediction SSE. We start with $\boldsymbol{X}_s$, the centered and scaled version of $\boldsymbol{X}$.
Algorithm:

1. Start with an initial function: $\hat{f}^{[0]}(\cdot) = \overline{y}$.
2. Compute residuals: $r_i = y_i - \hat{f}^{[0]}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$
3. Regress $r_1 \ldots, r_n$ on each column of $\boldsymbol{X}$ in turn. Take the column with smallest SSE as the only predictor, and call this model $\hat{g}(\boldsymbol{x})$.
4. Update the function: $\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu\hat{g}(\cdot)$ where ν is a small number, typically 0.1.
5. Repeat steps 2 to 4 for some specific number of iterations.

Note: some predictors may never enter the model, and will have coefficients of zero. Others may get chosen numerous times, and their coefficients are expected to approach the OLS solution coefficients.

```
> coef( lm(y ~ Z, data = supervis))
(Intercept)          Zx1          Zx2          Zx3          Zx4          Zx5
 16.2666667   -0.6397064   -2.3774774    1.6528936   -0.3578020    2.3296891
>  -0.6397064  +   1.6528936  -0.3578020
[1] 0.6553852
> require(mboost)  ## also loads  modeltools stats4 party survival splines
> supervis.boost <-  glmboost(y ~ Z, data = supervis)
> coef(supervis.boost)
(Intercept)          Zx1          Zx2          Zx3          Zx4          Zx5
 0.00000000   0.51953015 -0.05635068   0.18988396   0.00000000   0.02423444
attr(,"offset")
[1] 16.26667
> AIC(supervis.boost)
[1] 0.9616505
Optimal number of boosting iterations: 26
```

```
Degrees of freedom (for mstop = 26): 0.9363657

> coef(supervis.boost[26])
(Intercept)         Zx1         Zx2         Zx3         Zx4         Zx5
  0.0000000   0.5148691   0.0000000   0.1594324   0.0000000   0.0000000
attr(,"offset")
[1] 16.26667
>  0.5148691  + 0.1594324
[1] 0.6743015
> par(mai = par("mai") * c(1, 1, 1, 2.5))
> plot(supervis.boost)
> abline(v=26,col="grey")
```



The fourth column was never used, columns 2 and 5 entered after the optimal stopping point (26 iterations).

A problem: I don't know how to get error estimates for the parameter estimates (nor for predictions).

8 Random and Mixed Effects Models

Definitions:

Fixed effects: Only the levels of the factor which are used in this experiment (or observed in this study) are of interest. Questions for this factor could be: "Does mean response differ between levels of this factor?" or "What is the coefficient for the effect of x on y?"

Random Effects: The levels used (observed) are representative of some larger population of levels. Questions focus on how variable the random effect is across the population (not on individual estimates.)

A Mixed Effects Model uses at least one random and one fixed effect.

Which effects are "random"?

- A random effect is associated with some sort of a unit (an animal, a plot, a transect, a classroom, ...) or a collection of units, and the units (or collections) measured are representative of some broader population of units in the universe. We do not care about the estimated coefficient for any single unit, but rather wish to know how much variation there is between units in the population. (Estimation concerns population parameters. An individual random quantity can be "predicted", but would not be "estimated" because it does not describe an attribute of the population.)

- Random effects are those which are not repeatable. Consider clinical trials for a medication to lower blood pressure. The new medication and the older "standard treatment" will be used on samples of patients at 5 different centers around the country. For each person we measure the response: "change in BP". A second clinical trial would use different subjects and possibly different medical centers, so centers are random effects. The treatment effect is repeatable; it is a fixed effect. Similarly in an observational study, units of observation are always random, years are random, climatological and physiological effects are typically fixed.

Often we have to grapple with the choice of designating an effect as fixed or random. For example, a typical project for a masters level student in Ecology, LRES, or similar fields requires two years of data collection and we expect there to be a difference between years. To say that the particular years observed are representative of a broader population of years is a big assumption, which I would not want to make. The variance estimates (needed for random effects) based on two data points are very unstable, so I would prefer to use fixed effects when the sample size for random effects is so small. In either case, caution is needed in interpreting the effect of years.

Mixed Effects Model (formulation of Laird and Ware (1982)):

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{\epsilon}_i;\ \boldsymbol{b}_i \sim \text{ iid } N(\boldsymbol{0}, \boldsymbol{\Psi});\ \text{indep of } \boldsymbol{\epsilon}_i \sim \text{ iid } N(\boldsymbol{0}, \sigma^2\boldsymbol{V}),\ i = 1, \ldots, k \quad (6)$$

where $\boldsymbol{y}_i$ is a vector of observations from the ith level of some random factor, $\boldsymbol{\beta}$ is the population average coefficient vector, and $\boldsymbol{b}_i$ is the random effect. Typically, $\boldsymbol{Z}_i$ is a subset of the columns of $\boldsymbol{X}_i$ and $\boldsymbol{b}_i$ is an adjustment to the corresponding coefficients $\boldsymbol{\beta}_i$ for the

ith random level. We commonly use roman letters like b_i to remind ourselves that these are **random** effects.

Inference

With fixed effects, the primary emphasis is on estimating the coefficients, $\boldsymbol{\beta}$. With random effects the interest is in the variance-covariance parameters, so we are modeling $\sigma^2\boldsymbol{V}$ and $\boldsymbol{\Psi}$. Note the differences in typical inferences:

- Fixed effects: Estimate the coefficient (or a linear combination of coefficients), or test to see if a coefficient (or a linear combination of them) is zero.
- Random effects: How much do coefficients vary from one unit to another? How much of the overall variation is due to the random effect?

If a factor has k levels, notice the difference in degrees of freedom for estimation:

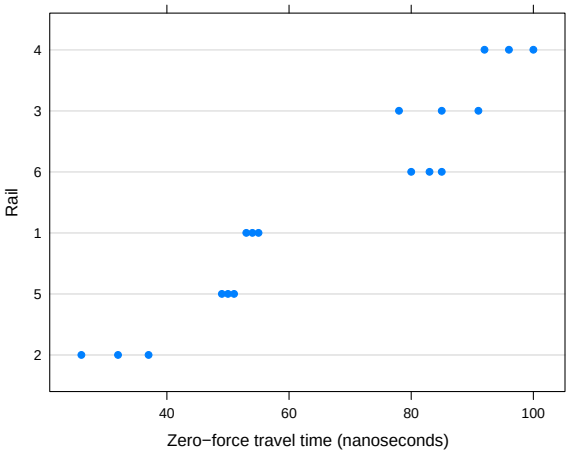
- Fixed effects use $k-1$ df for the individual effects.
- Random effects use 1 df for the variance from unit to unit.

The interaction between a fixed predictor and a random one will have to be random.

8.1 Simple Mixed Models

The Rails example (Pinheiro and Bates 2000)

To test rails made for railroad tracks non-destructively, a ultra-sound signal is sent from one end of the rail to the other. Imperfections in the steel increase the travel time. A sample of six rails is drawn at random from the production line, and each rail is measured three times. The response is in nanoseconds after subtracting off 36,100 ns.
Note that there is variation from one rail to the next, and within each rail we have variation as well.



The model for this data is

$$y_{ij} = \mu + b_i + \epsilon_{ij}; \quad i = 1, \ldots, 6, \quad j = 1, 2, 3$$

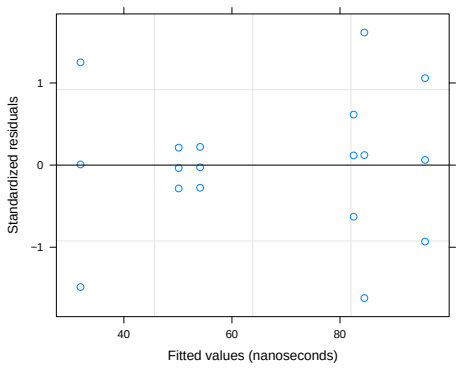
where $\epsilon_{ij} \sim \ iid\ N(0, \sigma^2)$ and $b_i \sim \ N(0, \sigma_b^2)$, independent of the ϵ's. Combining responses together for each rail, and using model (6), $\boldsymbol{X}_i$ and $\boldsymbol{Z}_i$ are columns of three ones, $\boldsymbol{\beta} = \mu$, and $\boldsymbol{b}_i$ is just a scalar, b_i. We assume independence between rails of the b_i's and of the ϵ_{ij}'s between rails. For this example, it is appropriate to assume $\boldsymbol{V} = \boldsymbol{I}$, so that the errors within a rail are also independent. We must assume that the two levels of random variation (b_i and $\boldsymbol{\epsilon}_i$) are independent in order to make variance parameters identifiable.
Alternative model:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \mathbf{r}_i; \ \mathbf{r}_i = \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{\epsilon}_i; \ \mathbf{r}_i \sim \ \text{iid}\ N(\mathbf{0}, \boldsymbol{Z}_i\boldsymbol{\Psi}\boldsymbol{Z}_i^{\mathsf{T}} + \sigma^2\boldsymbol{V}); \ i = 1, \ldots, k \qquad (7)$$

Using model (7) on the Rails data, the two error terms are combined giving a block diagonal variance-covariance matrix with $\sigma^2 + \sigma_b^2$ on the diagonal and covariance of σ_b^2 for two measurements on the same rail, 0 for measurements on different rails.

We can fit a model with a random rail effect in R using the `lme` function:

```
> require(nlme)
> data(Rail)
>  rail.lme1 = lme(travel ~ 1, data = Rail, random = ~1|Rail)
> rail.lme1
Linear mixed-effects model fit by REML
  Data: Rail
  Log-restricted-likelihood: -61.0885
  Fixed: travel ~ 1
(Intercept)
       66.5
Random effects:
 Formula: ~1 | Rail
        (Intercept) Residual
StdDev:    24.80547 4.020779

Number of Observations: 18
Number of Groups: 6
> plot(rail.lme1)
```



Using `lme`:

If you are familiar with the `lm` function in R, then you have seen most of the `lme` arguments before. The formula defines the fixed effects which here is just a 1 to represent the intercept. The `random` argument is new, and I read it as: fit an individual intercept (represented by a 1) for each (vertical bar) Rail. You might want to print the summary of the object to see t-tests for each coefficient. Above the `print` method is used which is less detailed. After `Random effects:` we see two estimates. The one labeled `Intercept` is the estimate of spread between rails, and the residual estimate is for within rail standard deviation.

Interpretation:

The estimates are:

$$\widehat{\mu} = 66.5 \text{ ns.} \qquad \widehat{\sigma}_b = 24.806 \text{ ns.} \qquad \widehat{\sigma} = 4.021 \text{ ns.}$$

Note that the the second and third estimates are in the standard deviation scale, not in variance scale. By default, the program used REML, that is, restricted maximum likelihood to estimate the parameters which gives unbiased estimates of variance parameters. If you want to use maximum likelihood instead (later we'll need this when comparing models with different fixed effects), include the argument `method = "ML"`.

The `intervals` command prints approximate 95% (by default) intervals for all parameters. Applying it to this data we see that mean μ is not close to zero, and neither between rail standard deviation nor within rail standard deviation is near zero. The width of the intervals tells us how precise the estimates are. Note that the estimate of σ is much more precise than that of σ_b. I strongly recommend examining the output of `intervals` because it gives ridiculously large intervals for parameters when the model is over-specified. An interval for a standard deviation which goes from 2e-10 to 2e+10 is an indication that the parameter is very poorly estimated, and the model should be simplified.

```
> intervals(rail.lme1)
Approximate 95% confidence intervals

 Fixed effects:
              lower est.    upper
(Intercept) 44.33921 66.5 88.66079
 Random Effects:
  Level: Rail
                   lower     est.    upper
sd((Intercept)) 13.27431 24.80547 46.35352
 Within-group standard error:
   lower     est.    upper
2.695007 4.020779 5.998747
```

In SAS use PROC MIXED. The model statement is the same as for PROC GLM. To fit only an intercept, use no predictor on the right. The random effect is specified in its own line with the term `subject` to identify what the subscript i refers to – here a rail.

```
options ps=76 ls=76;
data rails;
 rail = _n_ ;
 do i = 1 to 3;
     input travel @;
     output;
  end;
drop i;
datalines;
 55 53 54
 26 37 32
 78 91 85
 92 100 96
 49 51 50
 80 85 83
;
 /*  proc print;*/
run;
proc mixed data= rails;
  class  rail;
  model travel = ;
  random intercept / subject = rail;
run;
  /*  ############################  Output  ######################## */
                      Covariance Parameter Estimates

                    Cov Parm      Subject    Estimate

                    Intercept     rail         615.31
                    Residual                  16.1667
```

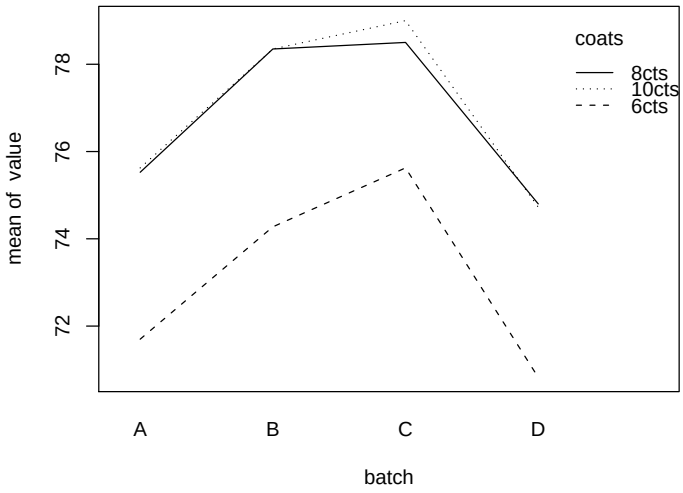
Estimates are $\widehat{\sigma}_b^2 = 615.31$ and $\widehat{\sigma}^2 = 16.167$, which agree with the (squares of the) estimates from R.

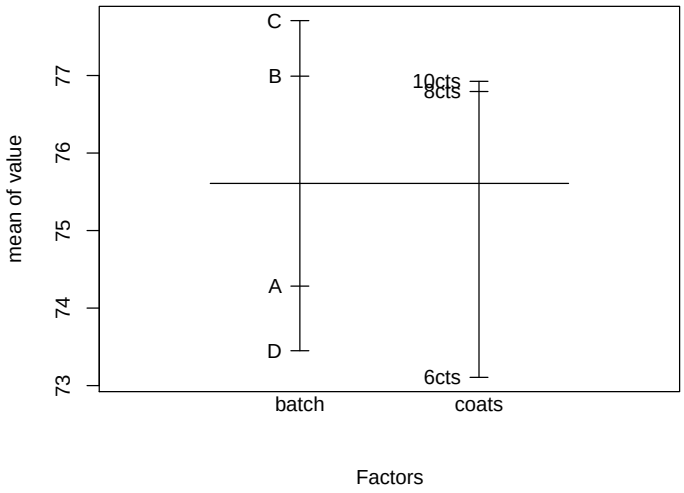
Randomized Block Example

Another very simple mixed effects model is that of a randomized block design (RBD) where we have a treatment factor (fixed) randomly applied within each of several blocks (random). RBD's are very common in agricultural, medical, and industrial experiments –

wherever a blocking variable is available which partitions the available units into groups which are similar.

A study was done to look at the effects of number of coats of lacquer (fixed at 6, 8, or 10) on the value of imitation pearls (exercise 25.17 in Kutner et al. (2004)). Four batches (considered random) were used.






```
>  pearls <- read.table("data/pearls",head=T)
> with(pearls,   interaction.plot(batch, coats, value))
> plot.design( value ~ batch + coats, data = pearls)
```

In the first plot, I am looking for interactions. If the lines were **not** generally parallel, then it would be a mistake to ignore interactions and focus only on the main effects. When interactions are present, one can interpret the effect of one variable (say `coats`) only if one picks a particular level of the other factor (`batch`). The second plot focuses on main effects, showing the means for each treatment and for each block.

The lines in the first plot look quite parallel. I see that batches B and C have higher mean value than batches A and D. Also, it seems 8 and 10 coats are quite similar, and have greater mean value than the 6-coat treatment.

Because there is no interaction, it makes sense to look at means by batch and by coats, as shown in the second plot. I see that the spread over batch is comparable to the spread across treatment, neither factor dominates the other by being much more spread out. The rankings I saw in the interaction plot (B-C better than A-D and 8 or 10 better than 6) are substantiated.

```
> require(nlme)
> pearls.lmefit <- lme(value ~ coats, data = pearls, random = ~1 | batch)
> summary(pearls.lmefit)
Linear mixed-effects model fit by REML
 Data: pearls
       AIC      BIC    logLik
  217.8680 226.9013 -103.9340

Random effects:
```

```
 Formula: ~1 | batch
        (Intercept) Residual
StdDev:    1.974262 2.044022

Fixed effects: value ~ coats
               Value Std.Error DF  t-value p-value
(Intercept) 76.92500 1.1115549 42 69.20486  0.0000
coats6cts   -3.81875 0.7226709 42 -5.28422  0.0000
coats8cts   -0.13125 0.7226709 42 -0.18162  0.8568
 Correlation:
          (Intr) cts6ct
coats6cts -0.325
coats8cts -0.325  0.500

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-2.1990458 -0.6281479  0.1033012  0.6554653  1.3637802

Number of Observations: 48
Number of Groups: 4
```

The call to `lme` is much like that in the rails example, but we have added a fixed effect for coats on the right-hand side. The output from `lme` gives coefficient estimates for the fixed effect (coats) and reports standard deviation estimates for the random effects. The first `anova` command below checks the "significance" of the fixed effect. The second one compares to a plain `lm` model which assumes no variance between blocks (You cannot fit a model with no random effect using `lme`, so `lm` was needed).

```
> anova( pearls.lmefit)
            numDF denDF  F-value p-value
(Intercept)     1    42 5385.570  <.0001
coats           2    42   17.997  <.0001

> pearls.lmfit <-  lm(value ~ coats, data = pearls)
> anova( pearls.lmefit, pearls.lmfit )
              Model df      AIC      BIC    logLik   Test  L.Ratio p-value
pearls.lmefit     1  5 217.8680 226.9013 -103.9340
pearls.lmfit      2  4 233.4531 240.6798 -112.7266 1 vs 2 17.58508  <.0001
```

SAS commands and output follow. The difference again is just in the model line where coats is added as a predictor.

```
data pearls;
   infile "../data/pearls" firstobs=2;
   input  value coats$ batch$;
run;
proc mixed data= pearls;
  class  coats batch;
  model value = coats;
  random intercept / subject = batch;
run;
  /*  ##########################  Output  ###################### */
                     Covariance Parameter Estimates

                    Cov Parm      Subject    Estimate
                    Intercept     batch        3.8977
                    Residual                   4.1780
```

```
                    Fit Statistics

        -2 Res Log Likelihood           207.9

            Type 3 Tests of Fixed Effects

                     Num     Den
Effect                DF      DF    F Value    Pr > F
coats                  2      42      18.00    <.0001
```

Estimates and results agree with those of R.

Variance-Covariance

Adding a random effect changes the variance-covariance structure of the responses as is shown in equation (7). For the two simple cases above, the random "intercept" is the same for observations within the same rail or batch, which generates a non-zero covariance and correlation for two measurements within the same rail or batch. For the pearls example, each of the four blocks contains 12 observations, so $\boldsymbol{X}_i$ is a 12 by 3 matrix with a column of ones for the intercept followed by two columns of dummy variables: a contrast between 10 and 6 coats and a contrast for 10 versus 8 coats. The $\boldsymbol{Z}_i$ matrix is again a column of twelve ones, and b_i is the random batch effect. The $\boldsymbol{X}_i$ matrix is identical for each i, or batch. In unbalanced designs or when there are covariates, it need not be the same for each i.

The variance of $\boldsymbol{y}_i$ comes from two sources – the underlying variance of the $\boldsymbol{\epsilon}_i$ (assume $\text{Var}(\boldsymbol{\epsilon}_i) = \boldsymbol{\Sigma}$) plus the variance of the random effects. Because the two sources are independent, we do not have to worry about a covariance term – it is zero. $\text{Var}(\boldsymbol{y}_i) = \boldsymbol{Z}_i\boldsymbol{\Psi}\boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{\Sigma}$ For the two cases we've looked at so far, $\boldsymbol{Z}_i\boldsymbol{\Psi}\boldsymbol{Z}_i^{\mathsf{T}}$ is a matrix of ones times σ_b^2, and $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$, so this structure is quite simple: the diagonal of $\text{Var}(\boldsymbol{y}_i)$ contains $\sigma^2 + \sigma_b^2$, and the off-diagonal is just σ_b^2. This is the compound symmetric structure we worked with in §4.1, and can be described with two parameters: either variance $(\sigma_b^2 + \sigma^2)$ and covariance (σ_b^2), or variance and correlation $(\rho = \sigma_b^2/[\sigma_b^2 + \sigma^2])$.

Next consider a vector of all the responses, $\boldsymbol{y} = [\boldsymbol{y}_1^{\mathsf{T}} \ \cdots \boldsymbol{y}_k^{\mathsf{T}}]^{\mathsf{T}}$. For $i \neq j$, the covariance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ is $\boldsymbol{0}$ due to independence of b_i from b_j and the iid nature of the residuals, which implies that the variance-covariance matrix of the complete response vector is block diagonal, all zeroes except for blocks $\boldsymbol{\Sigma}_i = \boldsymbol{Z}_i\boldsymbol{\Psi}\boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{\Sigma}$. If the variance parameters were known, we could use GLS to find the BLUEs of $\boldsymbol{\beta}$.

In the pearls example, there are 12 observations within each of four batches, so $\boldsymbol{\Sigma}_i$ is 12 by 12. From the output above, $\widehat{\sigma}_b^2 = 3.8977$ and $\widehat{\sigma}^2 = 4.1780$

```
  ## estimated  intra-batch correlation is:
>  3.8977/ (4.1780 + 3.8977 )
[1] 0.4826446
```

Another way to fit this model is by specifying a compound symmetric correlation structure in the `gls` function of the R library `nlme`, or by using a repeated statement in SAS PROC GLM.

```
> require(nlme)
> pearls.glsfit = gls(value ~ coats, correlation = corCompSymm(form=~1|batch),data=pearls)
> pearls.glsfit
Generalized least squares fit by REML
  Log-restricted-likelihood: -103.9340
```

```
Coefficients:
(Intercept)   coats6cts   coats8cts
   76.92500    -3.81875    -0.13125

Correlation Structure: Compound symmetry
 Formula: ~1 | batch
 Parameter estimate(s):
      Rho
0.4827887
Degrees of freedom: 48 total; 45 residual
Residual standard error: 2.842130

  /*  ########################## SAS CODE  ###################### */
proc mixed data= pearls;
  class  coats batch;
  model value = coats;
  repeated  / subject = batch type = CS ;
run;

  /*  ########################## Output    ###################### */
                     Covariance Parameter Estimates

                    Cov Parm     Subject    Estimate
                    CS           batch        3.8977
                    Residual                  4.1780

                            Fit Statistics
                 -2 Res Log Likelihood           207.9
```

Note that Log-restricted-likelihoods for the two models are identical, as are the coefficient estimates, and estimated correlation is the same to at least 3 significant figures (0.4826 versus 0.4828).

Interpretation is slightly different. When we use random effects we view batches as a source of added variation which explain about 48% of the variation in value (after adjusting out the effect of coats). When we use correlation, we are thinking of batches as being measured repeatedly and are asking "How similar are measurements within the same batch?"

8.2 Growth Curves – Random Coefficient Models

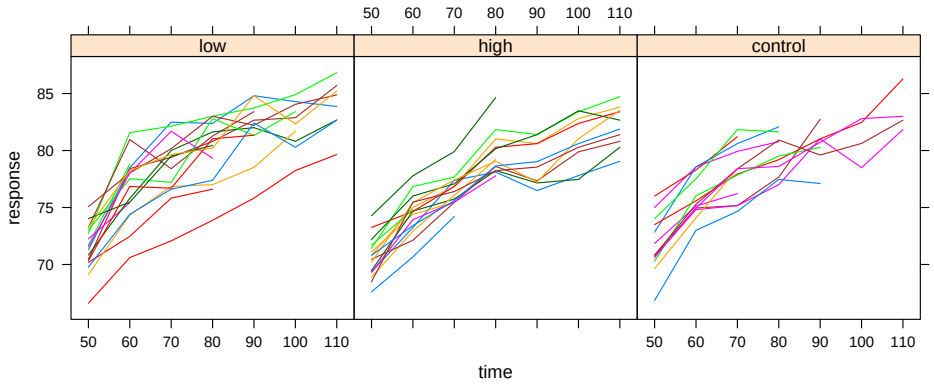
An example of a model with "random coefficients" is a growth curve model where individuals each have their own growth curve over time, and there is also a "population average" growth curve. In an study of the effect of testosterone inhibition on male rats, treatments were applied when the rats were 45 days old and repeated measurements were made on 50 animals over days 50 to 115 (data from Verdonck et al. (1998) as cited in Verbeke and Molenberghs (2000)). The three treatments used were

1. control
2. low dose of Decapeptyl (a testosterone inhibitor)
3. high dose of Decapeptyl.

The question of interest is whether there are differences in the response (pixel distance between two well-defined facial features) due to treatment. These are fixed effects for the population of male Wistar rats, so we are thinking about the "population average" growth curves for the three treatments.

We begin by looking at the data for each individual.

```
> rats = read.table("data/rats01.txt", head=T, sep="\t")
> summary(rats)
       ID              group          response         time      
 Min.   : 1.00   Min.   :1.00   Min.   :66.60   Min.   : 50  
 1st Qu.:16.00   1st Qu.:1.00   1st Qu.:74.62   1st Qu.: 60  
 Median :32.50   Median :2.00   Median :77.88   Median : 80  
 Mean   :32.58   Mean   :1.94   Mean   :77.53   Mean   : 80  
 3rd Qu.:49.00   3rd Qu.:3.00   3rd Qu.:81.02   3rd Qu.:100  
 Max.   :64.00   Max.   :3.00   Max.   :86.83   Max.   :110  
                                NA's   :98.00                
> rats$ID = factor(rats$ID)
> rats$group = factor(rats$group, labels=c("low","high","control"))
> require(lattice)
> xyplot(response ~ time|group, group = ID, data = rats, type="l")
  ## the group = subject command links together points for the same rat
```



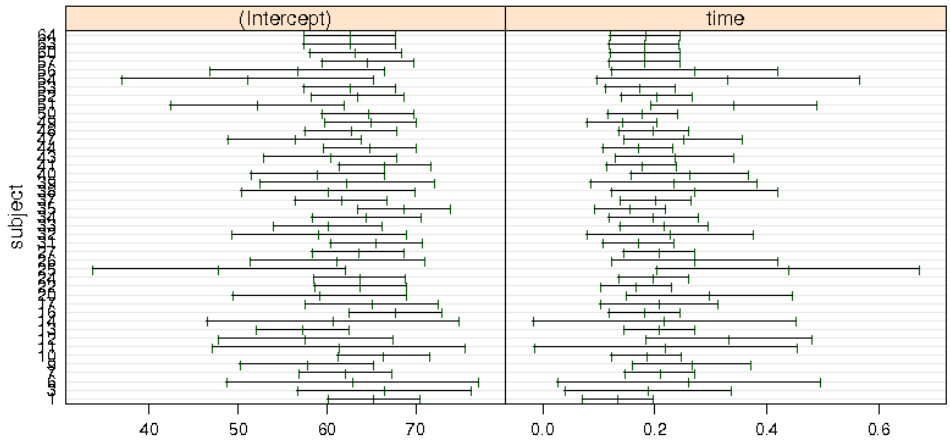
We see a general increase in the measured response which perhaps has a quadratic component because growth rate seems to slow down with age. There are missing data points due to deaths because the response is measured while rats are anesthetized, and not all animals survive. In fact, for 4 rats there is only one measurement, for 3 others only two. We cannot build a quadratic based on only two data points, so I will drop the animals who did not survive to day 70.

```
> table(is.na(rats$response),rats$time)
        50 60 70 80 90 100 110
  FALSE 50 46 43 38 29  24  22
  TRUE   0  4  7 12 21  26  28
> (lost = unique(rats$ID[rats$time==70 & is.na(rats$response)]))
[1]  8 28 30 59 62  5 23
>  rats2 = subset(rats, !(ID %in% lost))
```

When considering random effects, it helps to fit a model to each unit, in this case a linear growth model for response over time for each ID. The `lmList` function fits the same model for each unit using a common variance for all residuals. Plotting `intervals` applied to this fit allows us to compare confidence intervals for the intercepts and for the slopes.

```
> require(nlme)
```

```
> rat.listfit1 = lmList(response ~ time|subject, data = rats2,
+                       subset = !is.na(response))
> plot(intervals(rat.listfit1))
```



I look to see if all intervals overlap, that is, "Can we find some vertical line which intersects almost all of the 95% CI's?" If so, a random component may be unnecessary for that coefficient. In the plot above, the a value of 62 falls within almost all of the intercept intervals, and a slope of .2 is consistent with almost all of the slope intervals, suggesting that random effects are not terribly strong. To obtain a test, fit a model with is random slope and intercept, and a model with only random intercept. When looking at different random effects, it's best to start with a full (or even over-specified) model, so we'll include a quadratic term as a fixed effect.

```
> rats.lme2 = lme( response ~ 1+ group * poly(time,2),data=rats2, random = ~1 + time|ID,
  subset = !is.na(response))
> rats.lme1 = update(rats.lme2, random = ~1|ID)
> rats.lm0 = lm( response ~ 1+ group * poly(time,2),data=rats2, subset = !is.na(response))
> anova(rats.lme2,rats.lme1,rats.lm0)
          Model df       AIC       BIC    logLik   Test  L.Ratio p-value
rats.lme2     1 13  910.0681  954.9316 -442.0341
rats.lme1     2 11  906.0681  944.0296 -442.0341 1 vs 2   0.0000       1
rats.lm0      3 10 1059.4667 1093.9771 -519.7334 2 vs 3 155.3986  <.0001
```

In each fixed formula a group effect is specified, a quadratic curve over time, and the interaction (so curves might differ between groups). In the first random formula, a random intercept and a random slope is fit for each rat. The form of the $\mathbf{\Psi}$ matrix is

$$\mathbf{\Psi} = \begin{bmatrix} \sigma^2_{b_0} & \rho_{0,1}\sigma_{b_0}\sigma_{b_1} \\ \rho_{0,1}\sigma_{b_0}\sigma_{b_1} & \sigma^2_{b_1} \end{bmatrix}$$

The estimate of these values is given in the summary.

```
> summary(rats.lme2)
Linear mixed-effects model fit by REML
 Data: rats2
  Subset: !is.na(response)
       AIC      BIC    logLik
  910.0681 954.9316 -442.0341

Random effects:
 Formula: ~time | ID
```

```
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev       Corr
(Intercept) 1.904892e+00 (Intr)
time        3.601499e-06 0
```

meaning: $\widehat{\sigma}_{b_0} = 1.905,\ \widehat{\sigma}_{b_1} = 3.60 \times 10^{-06},\ \widehat{\rho} = 0$. Or with the extracting function:

```
>  getVarCov(rats.lme2)
Random effects variance covariance matrix
            (Intercept)        time
(Intercept)  3.6286e+00 -5.2085e-10
time        -5.2085e-10  1.2971e-11
  Standard Deviations: 1.9049 3.6015e-06
```

The second model uses only a random intercept, not a random slope, so the $\mathbf{\Psi}$ matrix is just a scalar.

```
>  getVarCov(rats.lme1)
Random effects variance covariance matrix
            (Intercept)
(Intercept)      3.6286
  Standard Deviations: 1.9049
```

The above `anova` command compares the three models. We see that the random slope does not improve upon the random intercept (only) model because logLikelihood stayed the same and the p-value is very high. Both random effects models improve substantially over the `lm` model with no random effects.

The final model is:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\mathbf{b}_i + \boldsymbol{\epsilon}_i$$

where $\boldsymbol{y}_i$ is the vector of responses for the ith rat, $\boldsymbol{X}_i$ has five columns: a column of all ones for the intercept, two columns which separate low from high and low from controls (for a given rat each is either all 0's or all 1's), a column of times, and finally a column of squared times. The $\boldsymbol{\beta}$ vector contains the five fixed (population averaged) effects: intercept, group differences, slope across time, and quadratic coefficient.

The random effects, $\mathbf{b}_i$ are just random intercepts for each rat, so $\boldsymbol{Z}_i$ is simply a vector of ones.

The covariate `time` makes the variance matrix for $\boldsymbol{y}_i$ much more complex than the simple compound symmetric structure we saw with a randomized block design. $\text{Var}(\boldsymbol{y}_i) = \boldsymbol{Z}_i\text{Var}(\boldsymbol{b}_i)\boldsymbol{Z}_i^{\mathsf{T}} + \text{Var}(\boldsymbol{\epsilon}_i)$ has elements $\sigma^2_{b_0} + (t_j + t_k)\sigma_{b_0b_1} + \sigma^2_{b_1}t_jt_k + \sigma^2$ in position (j, k). The entire variance-covariance matrix for all observations is block diagonal with ith block $\text{Var}(\boldsymbol{y}_i)$.

SAS Proc Mixed code for the final model:

```
DATA rats;
   infile "../data/rats01.csv" firstobs=2 dlm=',' ;
   input ID  group   response  time ;
   timeSq = time * time;
PROC MIXED;
  class ID group;
  model response = time group time timeSq / s;
  random intercept / subject = ID ;
run;
```

```
 Covariance Parameter Estimates
Cov Parm      ID    Estimate
Intercept     ID          3.6177
Residual                  1.6155

    Solution for Fixed Effects
                        Standard
Effect        group    Estimate       Error      DF    t Value    Pr > |t|
Intercept              47.1927      1.5398      40      30.65      <.0001
group         1         0.2891      0.7527     197       0.38      0.7013
group         2        -1.3576      0.7536     197      -1.80      0.0732
time                    0.6419     0.03835     197      16.74      <.0001
timeSq                -0.00288    0.000244     197     -11.80      <.0001

    Type 3 Tests of Fixed Effects
          Num     Den
Effect     DF      DF    F Value    Pr > F
group       2     197       2.92    0.0560
time        1     197     280.09    <.0001
timeSq      1     197     139.34    <.0001
```

8.3 Estimation in Mixed Effects Models

Variance Components

Before computing became cheap, methods based on ANOVA were used to estimate variance components in balanced mixed effects models. For example, Kutner et al. (2004) p 1052 give a table of expected mean squares for balanced two-factor ANOVA models when 0, 1, or 2 of the factors are considered random. Under the model:

$$y_{ijk} = \mu + \alpha_i + b_j + \epsilon_{ijk},\ k = 1, \ldots, n$$

assuming $\alpha_i,\ i = 1, \ldots, a$ are fixed effects and $b_j \sim N(0, \sigma_b^2)$, independent of $\epsilon_{ijk} \sim N(0, \sigma^2)$, the MOM estimator of σ_b^2 is

$$\tilde{\sigma}_b^2 = \frac{\text{MSB - MSE}}{na}$$

because $E(\text{MSB}) = \sigma^2 + na\sigma_b^2$ and $E(\text{MSE}) = \sigma^2$. This type of estimator has several problems:

1. It can take on negative values when MSE is larger than MSB.
2. It depends on having n observations in each cell. When counts are not balanced, there is not a unique ANOVA decomposition of the sums of squares.
3. The MOM estimators are available only for simple models.

Maximum Likelihood

Another approach is to write out the likelihood of the mixed model and then maximize it simultaneously in $\boldsymbol{\beta}$ and in variance parameters, $\boldsymbol{\theta}$. One can only use maximum likelihood when a particular distribution is assumed, so we typically assume the Gaussian distribution. We'll examine the mixed model with one level of random effect,

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta}_i + \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{\epsilon}_i,\ \text{for } i = 1, \ldots, M$$

$\mathbf{b}_i \sim N(\mathbf{0}, \boldsymbol{\Psi})$ independent of $\boldsymbol{\epsilon}_i \sim N(\mathbf{0}, \sigma^2 \boldsymbol{V}_i)$. We can combine all responses into one big $N = \sum_{i=1}^{M} n_i$ vector and look at the total likelihood:

$$L(\boldsymbol{\beta}, \boldsymbol{\theta}\sigma^2 | \boldsymbol{y}) = \prod_{i-1}^{M} p(\boldsymbol{y}_i | \boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2)$$

(assumes independence between the units with random effects). The random $\boldsymbol{b}_i$ terms are part of the mean vector, but are not of interest in themselves, and may be integrated out to get the marginal distribution for the data:

$$L(\boldsymbol{\beta}, \boldsymbol{\theta}\sigma^2 | \boldsymbol{y}) = \prod_{i-1}^{M} \int p(\boldsymbol{y}_i | \boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2) d\boldsymbol{b}_i$$

Pinheiro and Bates (2000) describe in detail various techniques used to efficiently compute this likelihood and maximize it. Assuming for the moment, $\boldsymbol{V} = \boldsymbol{I}$, the essential points are:

1. Work with the precision matrix Δ instead of $\boldsymbol{\Psi}$ where
$$\Delta^{\mathsf{T}} \Delta = \sigma^2 \boldsymbol{\Psi}^{-1}$$

2. Augment the data vectors and model matrices:
$$\tilde{\boldsymbol{y}}_i = \begin{bmatrix} \boldsymbol{y}_i \\ \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{X}}_i = \begin{bmatrix} \boldsymbol{x}_i \\ \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{Z}}_i = \begin{bmatrix} \boldsymbol{Z}_i \\ \Delta \end{bmatrix}$$
using a "pseudo-data" approach. This gives a simplified expression for the likelihood:
$$L(\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2 | \boldsymbol{y}) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-\sum_{i=1}^{M} ||\tilde{\boldsymbol{y}}_i - \tilde{\boldsymbol{X}}_i \boldsymbol{\beta} - \tilde{\boldsymbol{Z}}_i \widehat{\boldsymbol{b}}_i||^2}{2\sigma^2} \right) \prod_{i=1}^{M} \frac{\text{abs}|\Delta|}{\sqrt{|\tilde{\boldsymbol{Z}}_i^{\mathsf{T}} \tilde{\boldsymbol{Z}}_i|}}$$

3. Profile the likelihood as a function only of $\boldsymbol{\theta}$ using:
$$\widehat{\sigma}^2(\boldsymbol{\theta}) = \frac{1}{N} ||\boldsymbol{y}_e - \boldsymbol{X}_e (\widehat{\boldsymbol{b}}_1^{\mathsf{T}}, \ldots, \widehat{\boldsymbol{b}}_M^{\mathsf{T}}, \widehat{\boldsymbol{\beta}}^{\mathsf{T}})^{\mathsf{T}}||^2$$
where $\boldsymbol{y}_e$ and $\boldsymbol{X}_e$ are larger augmented systems. The denominator is N, not $N - p$, because this is a maximum likelihood estimator. Then substituting into the previous equation gives
$$L(\boldsymbol{\theta}) = L(\widehat{\boldsymbol{\beta}}(\boldsymbol{\theta}), \boldsymbol{\theta}, \widehat{\sigma}^2(\boldsymbol{\theta})) = \frac{\exp(-N/2)}{[2\pi\widehat{\sigma}^2(\boldsymbol{\theta})]^{N/2}} \prod_{i=1}^{M} \frac{\text{abs}|\Delta|}{\sqrt{|\tilde{\boldsymbol{Z}}_i^{\mathsf{T}} \tilde{\boldsymbol{Z}}_i|}}.$$

4. Solving any linear system, especially the large sparse systems generated above is done using QR decompositions to avoid matrix inversion.

5. Optimization is iterative using Newton-Raphson or EM (Expectation-Maximization) algorithms. Newton-Raphson methods converge very quickly once they are in the neighborhood of the solution, but can have trouble getting started. Pinheiro and Bates use EM for initial iterations to get close to a solution, then N-R to obtain the final solution. SAS PROC MIXED was written mainly by Russel Wolfinger. It uses ridge-stabilized Newton-Raphson followed by Fisher Scoring in its iterations (Littell et al. 2006).

Take Home Message:
The addition of random effects greatly increases the computational burden over that of OLS or GLS. When $\boldsymbol{V} = \sigma^2\boldsymbol{I}$, or even when $\boldsymbol{V}$ was complex, but known, solutions are relatively easy to obtain. Having to estimate variance/covariance parameters introduces many opportunities for ill-defined parameters and flat likelihoods.

Restricted Maximum Likelihood

Maximum likelihood estimates of variance components tend to be biased toward zero. An alternative which does not have this problem is to use residual (or restricted) maximum likelihood estimation. One definition of the REML method is restricted likelihood is integrated over the coefficient parameter space:

$$L_R(\boldsymbol{\theta}, \sigma^2|\boldsymbol{y}) = \int L(\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2|\boldsymbol{y}) d\boldsymbol{\beta}.$$

From a Bayesian perspective, one assumes a locally uniform prior on $\boldsymbol{\beta}$ and integrates $\boldsymbol{\beta}$ out of the likelihood. The integrated form is useful for computation.

Another definition is that one finds a linear combination of responses, $\boldsymbol{C}^\mathsf{T}\boldsymbol{y}$ such that $\boldsymbol{C}^\mathsf{T}\boldsymbol{X} = \boldsymbol{0}$, and then minimizes likelihood for $\boldsymbol{C}^\mathsf{T}\boldsymbol{y}$. Note the implication: if the $\boldsymbol{X}$ matrix changes, even to an equivalent reparameterization, then $\boldsymbol{C}$ will change and the REML estimated variances as well. **One cannot use comparisons of log-restricted-likelihood to compare models with different fixed effects.** We will use REML estimated models to compare models with different variance structures, different correlations, and different random effects, but will use ML estimates to compare models with different fixed effects.

The distribution of interest is

$$\boldsymbol{C}^\mathsf{T}\boldsymbol{y} \sim N(\boldsymbol{0}, \boldsymbol{C}^\mathsf{T}\mathrm{Var}(\boldsymbol{y})\boldsymbol{C})$$

which has dimension $N - p$ rather than N.

LRT P-values

When comparing likelihoods or restricted likelihoods of two models, one would like to say that twice the difference in log-likelihoods has a χ^2 distribution. Unfortunately the asymptotic argument needed is not valid at the boundary of the parameter space, as when we set one variance component to zero. Simulations can be used to find better estimates of true p-values, but the general result is that REML-based tests at the parameter space boundary using χ^2 distributions overstate the p-value (are conservative).

With fixed effects we will use ML-based tests, which again assume that a χ^2 distribution holds. Simulations show that these tests can be anticonservative – that reported p-value may be smaller than "true" p-value. (This problem occurs when the number of fixed effects being estimated is large compared to the sample size.)

8.4 P-values in Mixed Effects Models

Who Can We Trust? – How do we get reliable p-values for testing?

So far we've seen that fixed effects are compared using ML fits, while random effects are compared using REML fits. With REML fits, p-values tend to be conservative. Take as an example the orthodontic growth curves we looked at previously, and we will compare the model with random intercept (only) to the model with random slope AND random intercept. The first model uses only one variance parameter for Ψ, where the second uses three. Restricting the second to obtain the first involves setting a variance to zero which is on the boundary of the parameter space (also setting a correlation to 0, but that's not on the boundary). Verbeke and Molenberghs (2000) in §6.3 discuss the problem of the LRT when the null hypothesis is on the boundary of the parameter space. For this case, rather than use the χ^2_2 reference distribution (the change in df is 2), they argue that the correct reference distribution is a .5/.5 mixture of χ^2_2 and χ^2_1. In practice, for test statistic X, the p-value would be computed as

$$\text{p-value } = \frac{1}{2}P(\chi^2_1 > X) + \frac{1}{2}P(\chi^2_2 > X).$$

In R the default LRT output is

```
> OrthFem.lme1 <-  lme(distance ~ age, data = Orthodont, random = ~1|Subject,
                       subset = Sex=="Female")
> OrthFem.lme2 <-  update(OrthFem.lme1, random = ~age|Subject)
> anova( OrthFem.lme1, OrthFem.lme2)
             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
OrthFem.lme1     1  4 149.2183 156.1690 -70.60916
OrthFem.lme2     2  6 149.4287 159.8547 -68.71435 1 vs 2 3.789622  0.1503
```

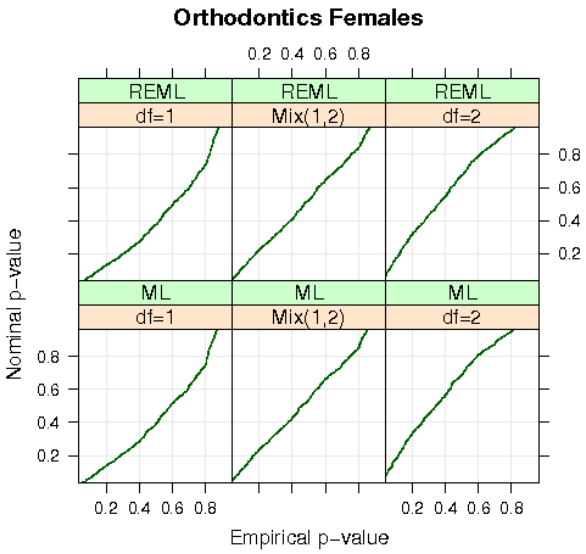
In fact, the p-value is better approximated by

```
>   .5 * (1- pchisq(3.789622,2) + 1 - pchisq( 3.789622, 1))
[1] 0.1009590
```

More generally, Verbeke and Molenberghs (2000) argue that when we start with q variance components and $\mathbf{\Psi}$ is $q \times q$, and consider dropping one random effect, the correct reference distribution is an equal mixture of a χ^2_q and a χ^2_{q-1}. This includes the special case of dropping a single random effect. In that case, we are comparing a model with one random effect to a model with only fixed effects. The correct reference distribution is a 50-50 mixture of a χ^2_1 and a χ^2_0 where the latter is an improper distribution placing all its mass at zero. For this case, we merely divide the the p-value output in half, because a χ^2_0 puts all its mass at 0.

We might wish to drop several variance components simultaneously. In that case, we must simulate the reference distribution. Pinheiro and Bates provide functions to do the simulation which we will illustrate using the orthodontics case discussed above.

```
>  orthLRTsim <-  simulate.lme(OrthFem.lme1,m2=OrthFem.lme2,
                               nsim=1000,seed=1234567890)
> plot(orthLRTsim,df=c(1,2),main ="Orthodontics Females" )
```



What are these plots? The function does 1000 iterations of

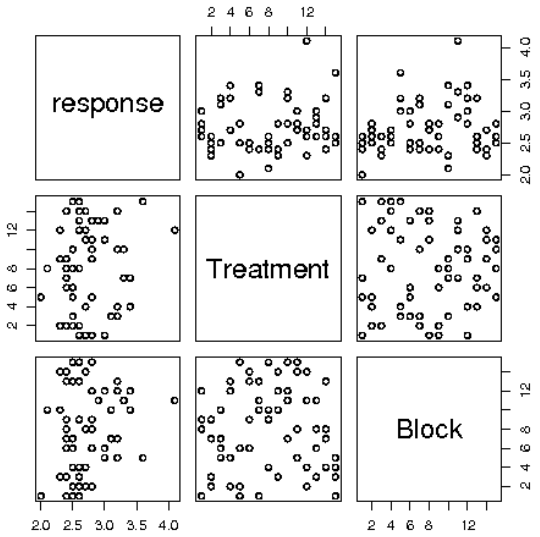
- generate a new data set under the null hypothesis that model 1 is correct.
- fit models 1 and 2, do a LRT between the models. Store the p-value.

The plots show the ordered p-values (y axis) versus quantiles of a known distribution: χ^2_1 in first column, χ^2_2 in third column, and, in the middle, a mixture distribution which is χ^2_1 half the time and χ^2_2 the other half. When the distribution of empirical p-values matches the reference distribution, the points will lie on the line $y = x$. Locate the $y = x$ line using intersections of gray lines where x and y values are equal. Starting on the right with the χ^2_2 distribution (used because we have two more parameters in model 2 than model 1), we see that nominal p-values are too high. The left-hand plots show that using χ^2_1 would over-correct giving p-values that are too small. In the middle we have a mixture distribution which works very well for both of these situations. **Note:** the usual reference distribution has, loosely speaking, too many degrees of freedom.

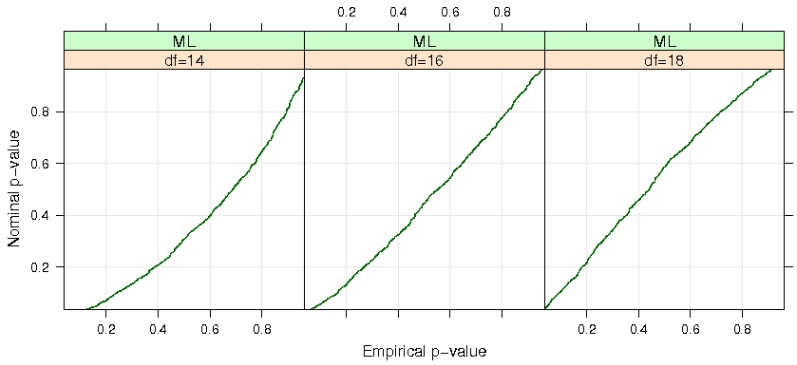
FIXED effects

When comparing fixed effects models, Pinheiro and Bates (2000) recommend NOT using the `anova` function with two arguments. They do allow it – it does not give a warning as when you try to make the same comparison of fixed effects but use REML fit models. The problem is that the p-values are again biased, but in the opposite direction. They are anti-conservative, giving p-values that are smaller than they should be. To illustrate, we look at the `PBIB` data which has 15 treatments (fixed effects) applied within 15 blocks (random effects) but each treatment was not used in each block, a partially balanced incomplete block design.

```
> require(SASmixed)
> plot(PBIB)
> with(PBIB,table(Treatment))
Treatment
 1 10 11 12 13 14 15  2  3  4  5  6  7  8  9
 4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  ## each Treatment is used 4 times
> with(PBIB,table(Block))
Block
 1 10 11 12 13 14 15  2  3  4  5  6  7  8  9
 4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  ## each Block is used 4 times
> with(PBIB,table(Block, Treatment))
  ## output not shown,
  ## but  it has 11 zeroes in each row and column
```




```
## Now to test treatment effect:
> PBIB.lme1 = lme(response ~ 1, data =PBIB, random = ~1|Block)
> PBIB.lme2 = lme(response ~ Treatment, data =PBIB, random = ~1|Block)
> PBIB.sim <-  simulate.lme(PBIB.lme1,m2=PBIB.lme2,nsim=1000,seed=1234567890)
> plot(PBIB.sim, df = c(14,16,18) )
```



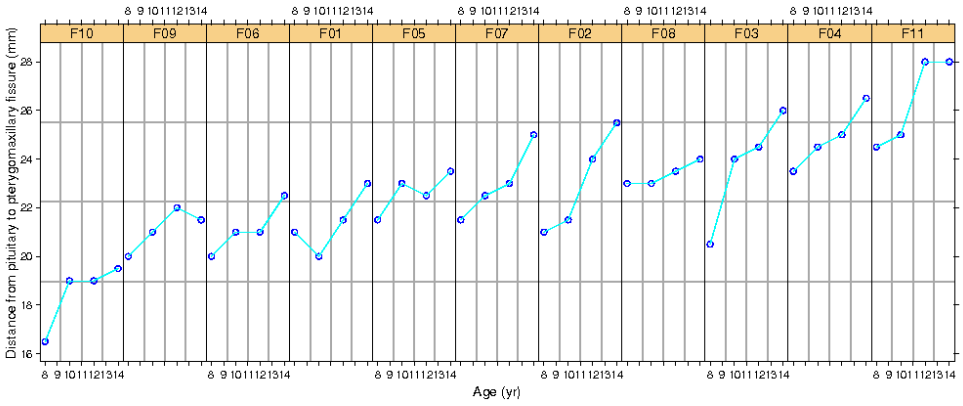
The LRT assumes 14 df, but the plot shows us that it should be more like 18 df. The first anova below overstates the importance of Treatment. Either the second or third seems more reasonable in light of the simulation results.

```
> anova( update(PBIB.lme1,meth="ML"),  update(PBIB.lme2,meth="ML"))
          Model df      AIC      BIC    logLik   Test  L.Ratio p-value
PBIB.lme1     1  3 52.15189 58.43493 -23.07595
PBIB.lme2     2 17 56.57058 92.17444 -11.28529 1 vs 2 23.58131  0.0514

> anova( PBIB.lme2)
            numDF denDF   F-value p-value
(Intercept)     1    31 1654.2066  <.0001
Treatment      14    31    1.5312  0.1576

> anova( update(PBIB.lme2,meth="ML"))
            numDF denDF   F-value p-value
(Intercept)     1    31 1401.3827  <.0001
Treatment      14    31    1.5552  0.1493
```

This data example is a good illustration of the problem because there are 15 estimated fixed effects and only 60 observations. When $n >> p$, this is not such a problem. Asymptotic theory does prove that the reference distributions are correct as $n \to \infty$, but we do have to be careful when using small samples.
Note: in contrast to the random effects case, with fixed effects the default reference distribution uses too few degrees of freedom.

8.5 SAS Proc Mixed

In SAS, Proc Mixed has similar capabilities to `lme` in S for fitting mixed effect models. As an example, we will look at the Orthodontics data on girls again. Here's the plot as a reminder:



And the R output from fitting a slope and intercept at the population level and

Model 1 a random intercept for each child

Model 2 a random slope and intercept for each child

```
> summary(OrthFem.lme1)
Linear mixed-effects model fit by REML
 Data: Orthodont
  Subset: Sex == "Female"
       AIC     BIC    logLik
  149.2183 156.169 -70.60916

Random effects:
 Formula: ~1 | Subject
        (Intercept)  Residual
StdDev:     2.06847 0.7800331

Fixed effects: distance ~ age
                Value Std.Error DF   t-value p-value
(Intercept) 17.372727 0.8587419 32 20.230440       0
age          0.479545 0.0525898 32  9.118598       0
Number of Observations: 44
Number of Groups: 11
>  summary(OrthFem.lme2)
Linear mixed-effects model fit by REML
 Data: Orthodont
```

```
  Subset: Sex == "Female"
       AIC      BIC    logLik
  149.4287 159.8547 -68.71435

Random effects:
 Formula: ~age | Subject
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.8841866 (Intr)
age         0.1609278 -0.354
Residual    0.6682746

Fixed effects: distance ~ age
                Value Std.Error DF   t-value p-value
(Intercept) 17.372727 0.7606027 32 22.840737       0
age          0.479545 0.0662140 32  7.242353       0
Number of Observations: 44
Number of Groups: 11
```

We have seen that the random slope is not really needed. In the output above the fact that the AIC's are almost the same is evidence of that.

Now in SAS, we'll also fit the two models. They both use the model statement `model distance = age` to give population slope and intercept. The difference is in the random specification.

```
 options ls=76 ps=66 nodate nonumber;
 dm "out;clear;log;clear;";
 data OrthoFem;
infile "data/OrthoFem" firstobs=2;
input   distance age Subject$ Sex$;
run;
title 'One Random Effect';
  proc mixed;
     class Subject;
     model distance = age;
     random Subject;
  run;
title 'Random Intercept and Slope';
proc mixed scoring=8;
     class Subject;
     model distance = age;
     random intercept age/type=un sub=Subject;
  run;
```

The first uses `random Subject` for a simple random adjustment to each subject. The second `random intercept age/type=un sub=Subject` sets a random intercept and a random slope on `age`. The `type = un` says to use an unstructured variance-covariance matrix for $\Psi = \text{Var}(\boldsymbol{b}_i)$ which is called $\mathbf{G}$ in SAS. Alternative to unstructured include VC = variance components which fits only the variances and leaves off-diagonals to be 0, or CS = compound symmetric, or several banded structures.

Output:

```
                          One Random Effect

                        The Mixed Procedure

                         Model Information

          Data Set                     WORK.ORTHOFEM
          Dependent Variable           distance
          Covariance Structure         Variance Components
          Estimation Method            REML
          Residual Variance Method     Profile
          Fixed Effects SE Method      Model-Based
          Degrees of Freedom Method    Containment

                      Covariance Parameter
                           Estimates

                      Cov Parm     Estimate
                      Subject        4.2786
                      Residual       0.6085
### Compare to squared estimates from 1st R model:  4.27856 and 0.60845

                          Fit Statistics

                -2 Res Log Likelihood           141.2
                AIC (smaller is better)         145.2  # 4 diff from R
                AICC (smaller is better)        145.5
                BIC (smaller is better)         146.0


                     Solution for Fixed Effects

                               Standard
      Effect       Estimate       Error      DF    t Value    Pr > |t|
      Intercept     17.3727      0.8587      10      20.23      <.0001
      age            0.4795     0.05259      32       9.12      <.0001
                  ## same as in R
;*************************************************************************;
                      Random Slope and Intercept

                         Estimated G Matrix

            Row    Effect       Subject        Col1        Col2

              1    Intercept    F01           3.5502     -0.1075
              2    age          F01          -0.1075     0.02590


                    Covariance Parameter Estimates

                     Cov Parm     Subject    Estimate   ## R:
                     UN(1,1)      Subject      3.5502   1.8841866^2 = 3.55016
                     UN(2,1)      Subject     -0.1075   -0.354 *1.8842 * 0.1609 = -0.1073
                     UN(2,2)      Subject     0.02590   0.1609278^2 = 0.025898
                     Residual                  0.4466   0.6682746^2 = 0.44659
```

```
           Fit Statistics
-2 Res Log Likelihood           137.4
AIC (smaller is better)         145.4  # ~4 diff from R
AICC (smaller is better)        146.5
BIC (smaller is better)         147.0


   Null Model Likelihood Ratio Test
     DF    Chi-Square      Pr > ChiSq
      3         55.79          <.0001

      Solution for Fixed Effects
                 Standard
Effect       Estimate       Error      DF    t Value    Pr > |t|
Intercept     17.3727      0.7606      10      22.84      <.0001
age            0.4795     0.06621      10       7.24      <.0001
```

Differences between Proc Mixed and `lme`:

- As with plain linear models, there is no easy way to compare two fits via a LRT.
- SAS prints variance estimates, R prints standard deviations
- Proc Mixed can be used without a random term, like `gls` in R. (`lme` requires a random term.) You then use a repeated statement to set up the correlation structure.
- SAS does not separate the variance model from the correlation model. For example, in R, you could use AR1 or compound symmetric structure for correlations, then add in a `weights = varPower()` for the variances. In SAS you use a repeated command with type = AR1() or type =CS to get just the correlations, and type = AR1H or type =CSH to get "heteroscedastic" variances on top of the AR1 or CS structure.
- SAS can handle some types of spatial AND temporal correlation structures. For example, if the data on zebra counts is correlated over space (nearby plots are more similar than those far away) and time (repeated measures on each plot with possible correlation), SAS can use a "direct" or Kronecker product, for example `type = sp(gaus) @ ar(1)` to cross Gaussian spatial with ar(1).

8.6 Nested Random Effects

Crossed versus Nested

When including two factors in a model, whether they are fixed or random effects, one factor could be nested within the other, or the two factors could be crossed.

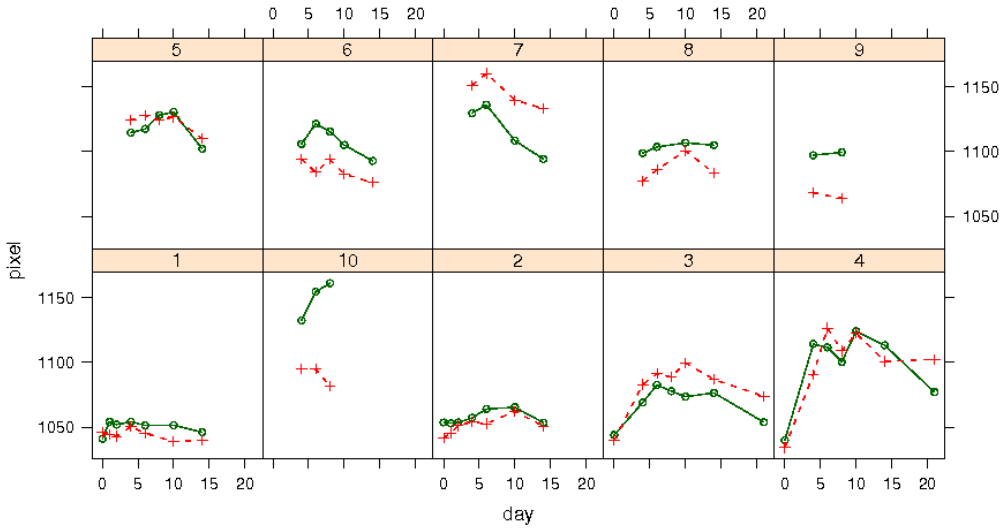
In a designed experiment on the rising of cake mixes, temperature and baking time are two (fixed) factors which each have meaning individually. Similarly for random effects, we could have a random effect for "mixer operator" and a random effect for "oven" where and combination of operator and oven is possible. These would be crossed random effects, and we can think about mixer-to-mixer variance, oven-to-oven variance and possible interaction effects.

An example of nesting is that counties are nested within states. Both Montana and Iowa (and, I'd guess, many other states) have a Jefferson county, but the county identifier

has meaning only when state is specified. There is no meaning to "Jefferson county" unless we know what state is referred to. With nested fixed effects, we can estimate the effect of the outer factor alone, and can look at the effect of the nested factor, which includes what would be called its main effect and the interaction under the crossed model.

Nested Random Effects Example

Pinheiro and Bates (2000) include the `Pixel` data in the `nlme` package which is an example of nested random effects. Dogs were measured with CT scans on both left and right sides of the brain and at several times up to 21 days after injection of a dye.



`Dog` is necessarily a random effect as these dogs represent some large population of dogs. The nature of the relationship with time changes markedly from dog to dog. Side within dog also exhibits variance, and the two sides are not responding exactly in parallel. Also, there is no inherent meaning to "left" or "right". Some dogs have generally higher pixel intensity on the left, some on the right, and that was to be expected. We do want to know how much intensity varies from one dog to another and from side-to-side within dogs. The model has the form:

$$\boldsymbol{y}_{ij} = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{Z}_{ij}\boldsymbol{b}_{ij} + \boldsymbol{\epsilon}_{ij},\ i = 1, \ldots, 10, j = 1, 2.$$

The authors used a quadratic model in time as the fixed effect ($\boldsymbol{\beta}$ contains intercept, slope, and quadratic effects) with random intercepts and slopes for each dog and random intercepts for side within dog. Hence, $\boldsymbol{b}_i$ is 2 by 1 and has a $N(\boldsymbol{0}, \boldsymbol{\Psi}_1)$ distribution where $\boldsymbol{\Psi}_1$ is 2 by 2, and $b_{ij} \sim N(0, \sigma_2^2)$. The assumption is needed that the three levels of randomness are all independent of each other.

```
> xyplot(pixel ~ day|Dog, group = Side, data =Pixel,type="o")
> DogPixel.lme1 = lme(pixel ~ day + I(day^2), data = Pixel, random = list(Dog =  ~day, Side = ~1))
> intervals(DogPixel.lme1)
Approximate 95% confidence intervals

 Fixed effects:
                   lower         est.        upper
(Intercept) 1053.0968335 1073.3391385 1093.5814435
day            4.3796921    6.1295970    7.8795018
I(day^2)      -0.4349038   -0.3673503   -0.2997967
```

```
 Random Effects:
  Level: Dog
                           lower       est.      upper
sd((Intercept))      15.9292331 28.3699236 50.5267617
sd(day)               1.0815600  1.8437506  3.1430677
cor((Intercept),day) -0.8944097 -0.5547222  0.1908261
  Level: Side
                   lower     est.    upper
sd((Intercept)) 10.41724 16.82427 27.17190

 Within-group standard error:
    lower      est.     upper
 7.634530  8.989608 10.585203
```

Note that there are now three sections of output within the random effects: one group for dog effects, one for side-within-dog, and the underlying residual error. We can use `anova` to examine the fixed effects.

```
> anova(DogPixel.lme1)
            numDF denDF   F-value p-value
(Intercept)     1    80 16771.709  <.0001
day             1    80     0.116  0.7345
I(day^2)        1    80   117.111  <.0001
```

To see specifically what $\boldsymbol{X}_i$, $\boldsymbol{Z}_i$, and $\boldsymbol{Z}_{ij}$ look like, take dog number 8 who was scanned on days 4, 6, 10, and 14. Note that days scanned are the same for left and right sides on any dog, therefore $\boldsymbol{X}_i$ and $\boldsymbol{Z}_i$ do not depend on $j = 1, 2$.

$$\boldsymbol{X}_8 = \begin{bmatrix} 1 & 4 & 16 \\ 1 & 6 & 36 \\ 1 & 10 & 100 \\ 1 & 14 & 196 \end{bmatrix}, \quad \boldsymbol{Z}_8 = \begin{bmatrix} 1 & 4 \\ 1 & 6 \\ 1 & 10 \\ 1 & 14 \end{bmatrix}, \quad \boldsymbol{Z}_{8,1} = \boldsymbol{Z}_{8,2} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Although $\boldsymbol{Z}_{8,1}$ and $\boldsymbol{Z}_{8,2}$ have the same appearance, they do get multiplied by different random coefficients, $b_{8,1}$ and $b_{8,2}$. The coefficients for $\boldsymbol{\beta}$ are the same for the whole population, and the coefficient $\boldsymbol{b}_i$ is the same for one dog.

Interpretation: For fixed effects we have significant curvature with a "frown" shape because the CI for `I(day^2)` is negative: (-0.43, -0.30). The slope may not be significant, but is needed for the quadratic to make sense, so we would not remove it. Under random effects, the confidence interval for the correlation between intercept and slope at the Dog level includes zero, so we might refit with correlation set to 0.

```
> DogPixel.lme2 = update(DogPixel.lme1, random = list(Dog = pdDiag(~day), Side = ~1))
> anova(DogPixel.lme1,DogPixel.lme2)
              Model df      AIC      BIC    logLik   Test  L.Ratio p-value
DogPixel.lme1     1  8 841.2102 861.9712 -412.6051
DogPixel.lme2     2  7 841.4547 859.6205 -413.7273 1 vs 2 2.244464  0.1341
```

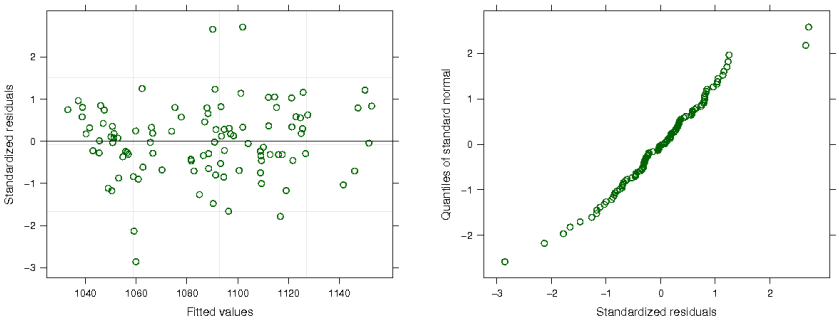
There seems to be no need for the off-diagonal term in $\boldsymbol{\Psi}_1$.

```
> intervals(DogPixel.lme2)
Approximate 95% confidence intervals
 Fixed effects:
                   lower         est.        upper
(Intercept) 1053.2366772 1072.6787874 1092.1208977
day            4.5030471    6.2248540    7.9466609
```

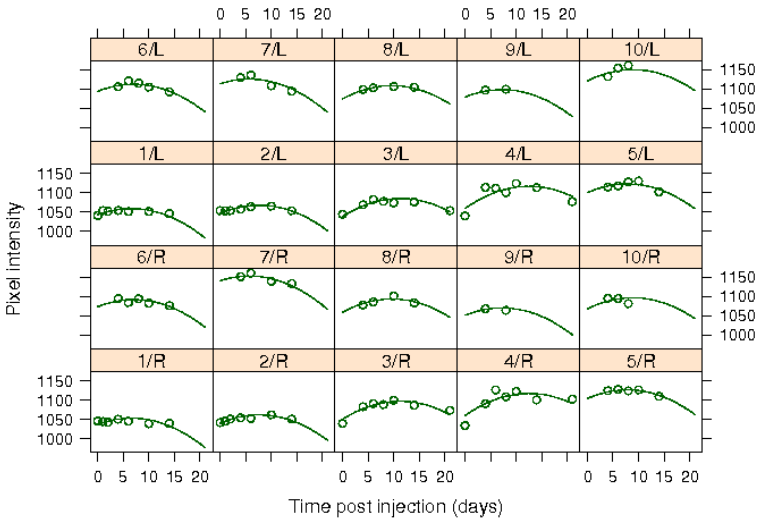
```
I(day^2)      -0.4362485   -0.3685759   -0.3009033
attr(,"label")
[1] "Fixed effects:"

 Random Effects:
  Level: Dog
                     lower      est.     upper
sd((Intercept)) 14.986197 26.947458 48.455622
sd(day)          1.039309  1.781721  3.054461
  Level: Side
                    lower     est.    upper
sd((Intercept)) 10.40949 16.81724 27.16940

 Within-group standard error:
    lower      est.     upper
 7.651286  9.020353 10.634390
```

Now we have a dog-to-dog standard deviation of intercept estimate of (14.99, 48.46), and slopes between dogs are estimated to vary with standard deviation in (1.04, 3.05). Within one dog, the standard deviation, σ_2, between sides is similar to the estimated standard deviation between dogs, being (10.41, 27.17), and the underlying variation has estimated σ in (7.65, 10.63).

A plot of residuals versus fitted values gives no surprises. The qqplot shows three large absolute value residuals.



The augmented prediction plot shows each estimated curve and the data points.
`plot(augPred(DogPixel.lme2))`



The estimated variance of the observations on one side of one dog is:

$$\text{Var}(\mathbf{y}_{ij}) = \boldsymbol{Z}_i \text{Var}(\boldsymbol{b}_i)\boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{Z}_{ij}\text{Var}(\boldsymbol{b}_{ij})\boldsymbol{Z}_{ij}^{\mathsf{T}} + \text{Var}(\epsilon_{ij}) \ = \boldsymbol{Z}_i\boldsymbol{\Psi}_1\boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{Z}_{ij}\boldsymbol{\Psi}_2\boldsymbol{Z}_{ij}^{\mathsf{T}} + \sigma^2\boldsymbol{I}$$

Estimate with:

$$\boldsymbol{Z}_i\widehat{\boldsymbol{\Psi}}_1\boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{Z}_{ij}\widehat{\boldsymbol{\Psi}}_2\boldsymbol{Z}_{ij}^{\mathsf{T}} + \widehat{\sigma}^2\boldsymbol{I}$$

Note the estimated values for $\text{Var}(\widehat{\boldsymbol{\Psi}}_i)$ appear as standard deviations (for the $\boldsymbol{D}$ matrix) and correlations (for $\mathbf{R}$) Some assembly is required. Proc Mixed code:

```
proc mixed;
  class Dog Side;
  model pixel =  day daySq  / s;
  random intercept day / subject = Dog type = VC ;
  random intercept /subject = Dog(Side);
run;
```

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
Intercept	Dog	726.17
day	Dog	3.1742
Intercept	Dog(Side)	282.82
Residual		81.3677

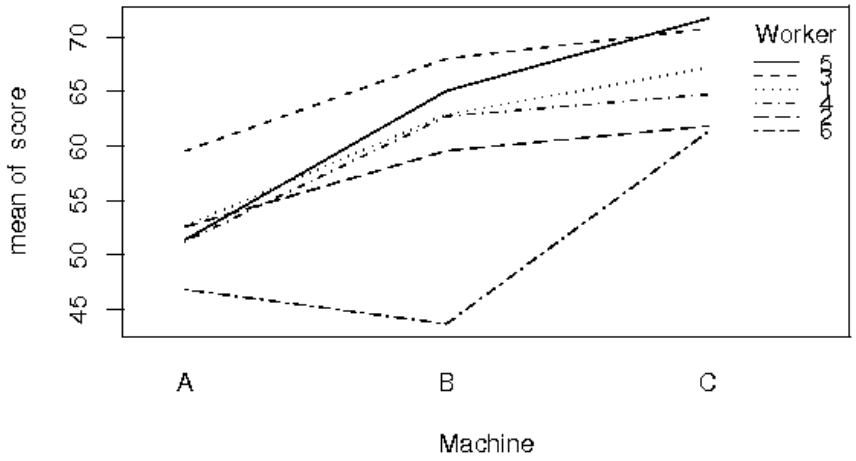
Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	1072.68	9.7696	9	109.80	<.0001
day	6.2248	0.8652	9	7.19	<.0001
daySq	-0.3686	0.03401	71	-10.84	<.0001

Further examples of nested versus crossed effects:

- To study of teaching methods, a random sample of elementary schools within a state was taken, and two teachers were selected within each school to take part in a comparison of methods. Within a school, the selected teachers are referred to as teachers A and B. Are teacher and school crossed or nested?
- In a study of the presence/absence of chronic wasting disease in wild deer populations takes place over many years. Each year 20 of 50 possible check stations are randomly selected for heavier scrutiny. Are year and station crossed or nested?

Random Interactions
Also in Pinheiro and Bates (2000) is data from a replicated randomized block experiment in which six different workers were each tested three times on three different machines (originally from Milliken and Johnson (1984)). Each worker is considered a "block" and random effects are used for workers. An interaction plot of means shows that one worker has a low score on machine B, whereas other workers all have a common pattern which is higher on B than on A.



Machine is a fixed effect, and the presence of an interaction certainly complicates the interpretation. Generally, most workers have a higher score on Machine B than Machine A, however one worker demonstrates that such is not always the case. We should check to see if the anomaly is due to one terrible score, or carried across worker 6's three attempts on Machine B.

```
> require(nlme)
> with( Machines, interaction.plot(Machine, Worker, score))

> with(Machines,Machines[Machine =="B" & Worker ==6,])
   Worker Machine score
34      6       B  43.7
35      6       B  44.2
36      6       B  43.0         ## the 3 scores are consistent
```

If there were no interaction, we could use a very simple mixed model:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i + \epsilon_i,\ i = 1, \ldots, 6$$

where β represents the fixed effects of Machines, and b_i (one-by-one) represents the random worker effect and $\boldsymbol{b}_i \sim\ N(0, \sigma_b^2)$, independent of $\epsilon_{ijk} \sim N(0, \sigma^2)$.

```
> machine.lme1 = lme(score ~ Machine-1, data = Machines, random = ~1|Worker)
> machine.lme1
Linear mixed-effects model fit by REML
  Data: Machines
  Log-restricted-likelihood: -143.4391
  Fixed: score ~ Machine - 1
MachineA MachineB MachineC
52.35556 60.32222 66.27222

Random effects:
 Formula: ~1 | Worker
        (Intercept) Residual
StdDev:    5.146552 3.161647

Number of Observations: 54
Number of Groups: 6
```

The interaction plot, however, makes us consider the interaction between the two effects. Will that be fixed or random? Random, because it tells us how the fixed Machine effect varies between the (random) Workers. We will not be able to "estimate" adjustments to the Machine effect for each Worker, though we could "predict" them for the 6 workers used in the study. This is really a new level of random effect, and requires a new model.

$$\boldsymbol{y}_{ij} = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{Z}_{ij}\boldsymbol{b}_{ij} + \boldsymbol{\epsilon}_{ij},\ i = 1, \ldots, 6,\ j = 1, 2, 3$$

where $b_i \sim N(0, \sigma_1^2)$, $b_{ij} \sim N(0, \sigma_2^2)$, both independent of each other and of $\epsilon_{ijk} \sim N(0, \sigma^2)$. The interaction random effect is nested within the first, so in the formula, we use `Worker/Machine` to indicate a Worker random effect (b_i) and a Machine within Worker effect, $\boldsymbol{b}_{ij}$. It is saying that even if you knew all about a particular worker's mean score (the first level of random effect) you still would have to account for how worker's scores vary from machine to another machine. (Bottom level variance is for a single worker operating one machine.) The nested model used only one additional parameter.

```
> machine.lme2 = update(machine.lme1, random = ~ 1|Worker/ Machine)
>  machine.lme2
Linear mixed-effects model fit by REML
  Data: Machines
  Log-restricted-likelihood: -107.8438
  Fixed: score ~ Machine - 1
MachineA MachineB MachineC
52.35556 60.32222 66.27222

Random effects:
 Formula: ~1 | Worker
        (Intercept)
StdDev:    4.781049

 Formula: ~1 | Machine %in% Worker
        (Intercept)  Residual
StdDev:    3.729536 0.9615768

Number of Observations: 54
Number of Groups:
             Worker Machine %in% Worker
                  6                  18
> anova(machine.lme1,machine.lme2)
             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
machine.lme1     1  5 296.8782 306.5373 -143.4391
machine.lme2     2  6 227.6876 239.2785 -107.8438 1 vs 2 71.19063  <.0001
```

Note that the fixed coefficients have not changed, the worker-to-worker StdDev is slightly smaller, but most of the Machine-in-worker StdDev has come from the residual, and that we get a tiny p-value, with a much smaller AIC. This is a better fit.

We could stick with one level of random effect by using the random effects formula `Machine|Worker` which builds a 3-by-1 vector $\boldsymbol{b}_i$ with a worker-to-worker variance for each machine and three correlation parameters.

```
>  machine.lme3 = update(machine.lme1, random = ~ Machine-1|Worker)
>  AIC(machine.lme3)
[1] 228.3112
```

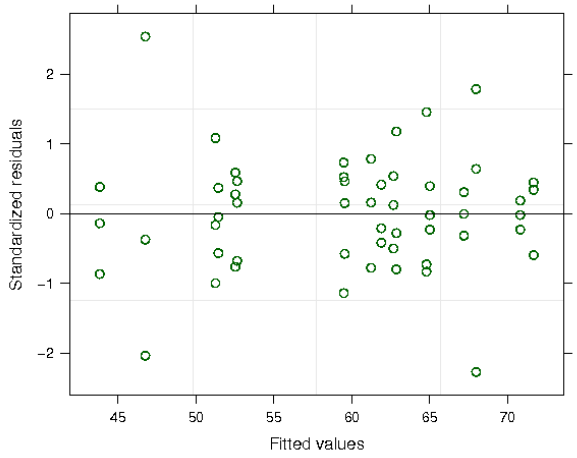
It's also possible to fit the three variances, one for each machine, while forcing machine-to-machine correlation to be zero using a `pdDiag` variance structure for $\boldsymbol{b}_i$.

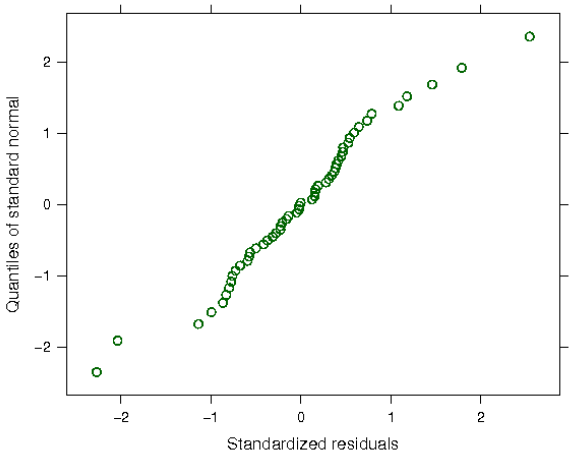
```
> machine.lme4 = update(machine.lme1, random = pdDiag(~ Machine-1))
> machine.lme4
Linear mixed-effects model fit by REML
  Data: Machines
  Log-restricted-likelihood: -108.8255
  Fixed: score ~ Machine - 1
MachineA MachineB MachineC
52.35556 60.32222 66.27222

Random effects:
 Formula: ~Machine - 1 | Worker
 Structure: Diagonal
        MachineA MachineB MachineC  Residual
StdDev: 4.079281 8.625291  4.38948 0.9615766

Number of Observations: 54
Number of Groups: 6
> anova(machine.lme2,machine.lme4)
             Model df      AIC      BIC    logLik   Test  L.Ratio p-value
machine.lme2     1  6 227.6876 239.2785 -107.8438
machine.lme4     2  7 231.6511 245.1739 -108.8255 1 vs 2 1.963511  0.1611
```

However, neither AIC nor p-value shows any improvement over our second (two-level) model.

Note the difference in interpretation. Model `machine.lme4` estimates a variance for each of the three machines over all workers, whereas model `machine.lme2` fits a variance for machine-within-worker which is not machine-specific.

The residuals exhibit no fan shape; there are two negative residuals rather far from zero, but otherwise normality is not a big issue.





Proc Mixed code:

```
title "No Interaction";
proc mixed;
  class Worker Machine;
  model score =  Machine   / s;
  random intercept  / subject = Worker  ;
run;

title "Random Interaction";
proc mixed;
```

```
  class Worker Machine;
  model score =  Machine   / s;
  random intercept  / subject = Worker ;
  random Worker(Machine);
run;
```

No Interaction
Covariance Parameter Estimates

Cov Parm	Subject	Estimate
Intercept	Worker	26.4870
Residual		9.9960

Fit Statistics

-2 Res Log Likelihood	286.9
AIC (smaller is better)	290.9
AICC (smaller is better)	291.1

```
################################################################################
```

Random Interaction
Covariance Parameter Estimates

Cov Parm	Subject	Estimate
Intercept	Worker	22.8584
Worker(Machine)		13.9095
Residual		0.9246

Fit Statistics

-2 Res Log Likelihood	215.7
AIC (smaller is better)	221.7
AICC (smaller is better)	222.2

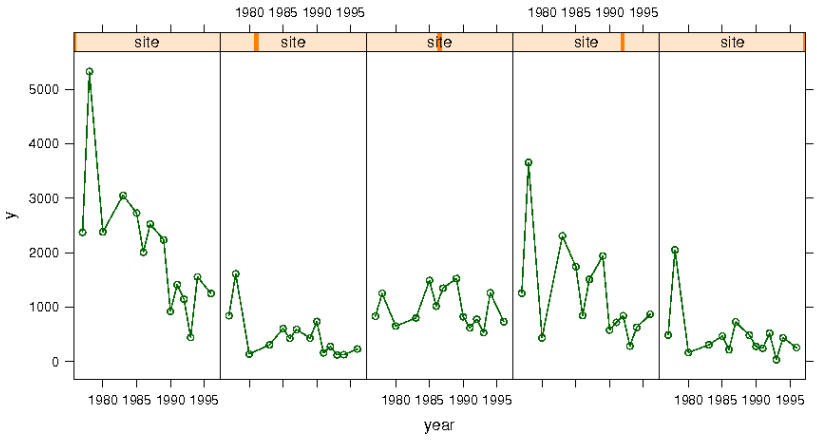
Solution for Fixed Effects

Effect	Machine	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept		66.2722	2.4858	5	26.66	<.0001
Machine	A	-13.9167	2.1770	10	-6.39	<.0001
Machine	B	-5.9500	2.1770	10	-2.73	0.0211

8.7 Crossed Random Factors and `lmer`

Example

In Piepho and Ogutu (2002), the authors examine count data from recurring aerial surveys of Thompson's Gazelle in the Serengeti. They wish to know if there is a significant trend over time (1977 – 1996). Years are considered as a random effect (a random adjustment for the entire region) and as a fixed effect (looking for an overall trend across years which are centered and labeled in the model as w_j). The data were gathered at individual sites, which introduce a separate random effect which is crossed with year. "Crossed" means that the site identification is the same in each year. If different sites were selected in each year, then site would have meaning only when year is specified, and site would be nested within year. In this study, however, sites are identifiable without considering year; they are the same sites measured repeatedly.



Two models were considered:

$$y_{ijk} = (\beta_0 + a_{0i} + b_{0j}) + (\beta_1 + a_{1i})w_j + \epsilon_{ijk} \quad (8)$$

$$y_{ijk} = (\beta_0 + \gamma_i + b_{0j}) + (\beta_1 + a_{1i})w_j + \epsilon_{ijk} \quad (9)$$

In model (1) we have

- fixed effects: the overall mean, β_0, and the overall slope or trend β_1 over years w_j
- random effects: b_{0j}, an adjustment to overall mean due to year j, a_{0i}, an adjustment to overall mean due to site i, and a_{1i}, an adjustment to trend over years due to site i

Model (2) takes out the random mean adjustment for site, a_{0i}, and puts in a fixed site effect, γ_i. Interpretation: the mean for each site across all years is of interest for only these 5 sites.

Using `lmer`

Doug Bates has developed another mixed effects package called `lme4` which contains a function `lmer`. It is explained well in Gałecki et al. (2013). The `lmer` function is similar to the `lme` function in `nlme`, but has several differences.

- `lmer` does not use a `random =` statement. There is just one formula, and a random effect is entered in parentheses like `+ (1 | Subject)` for random intercept or `+ (time|Subject)` for random slope and intercept with a correlation between them.
- `lmer` is optimized for crossed random effects.
- `lmer` handles responses which are not just normally distributed, but could be Poisson or binomial count data.
- `lmer` does not feel as finished, for instance `update` doesn't work right for me, there are no `intervals`, `predict` or `augPred` methods for `lmer` objects. Dr. Bates describes it as a research project.
- `lmer` and `Matrix` use object orientation under the "S4" system. Differences:
  - To access a part of an S4 object use the `@` operator rather than $. Compare extracting the call from an lme object and from an lmer object:

```
> library(nlme)              ### S3 example
> rails.lme <- lme(travel ~ 1, random = ~1|Rail, Rail)
> rails.lme$call
lme.formula(fixed = travel ~ 1, data = Rail, random = ~1 | Rail)
> detach("package:nlme")

> require(lme4)              ### S4 example
> xyplot(Reaction ~ Days|Subject,sleepstudy, layout=c(6,3))
> sleep.fit1 <- lmer( Reaction ~ Days + (Days|Subject), sleepstudy))
> sleep.fit1@call
lmer(formula = Reaction ~ Days + (Days | Subject), data = sleepstudy)
```

– With S3 object orientation, a generic function like `summary` or `anova` looks at the argument and determines which class of object it is, for example an object of class `lm` is created when you save a model fitted with the `lm` function. To summarize a linear model object, a special form of `summary` called `summary.lm` is used. Typing `summary.lm` shows how the summary function is defined for an object of class `lm`.

  Under S4, `summary` is a "method" defined when the class of object is defined. Invoking the method is more like sending a message to the object saying "summarize yourself". To see the definition for an object of class `lmer` type `selectMethod("summary",signature="lmer")`

We end up using similar commands in either system. For example, to compare two models:

```
> sleep.fit2 <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), sleepstudy)
> anova(sleep.fit1,sleep.fit2)
Data: sleepstudy
Models:
sleep.fit2: Reaction ~ Days + (1 | Subject) + (0 + Days | Subject)
sleep.fit1: Reaction ~ Days + (Days | Subject)
           Df     AIC     BIC  logLik  Chisq Chi Df Pr(>Chisq)
sleep.fit2  4 1760.05 1772.82 -876.02
sleep.fit1  5 1761.99 1777.95 -875.99 0.0606      1     0.8056    ## full model is last
```

Model `sleep.fit1` uses random intercept and slope for Subjects, fitting two variances and a correlation between the two effects. Using separate statements for intercept and slope in model `sleep.fit2` removes the correlation between slope and intercept, setting it to zero. We did the same with `lme` using the pdDiag structure.

```
> summary(sleep.fit1)
Linear mixed-effects model fit by REML
Formula: Reaction ~ Days + (Days | Subject)
   Data: sleepstudy
  AIC  BIC logLik MLdeviance REMLdeviance
 1754 1770 -871.8       1752         1744
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 Subject  (Intercept) 612.554  24.750
          Days         35.058   5.921   0.066
 Residual             654.916  25.591
number of obs: 180, groups: Subject, 18
```

```
Fixed effects:
            Estimate Std. Error t value
(Intercept)  251.405      6.826   36.83
Days          10.467      1.546    6.77

Correlation of Fixed Effects:
     (Intr)
Days -0.137
> summary(sleep.fit2)
Linear mixed-effects model fit by REML
Formula: Reaction ~ Days + (1 | Subject) + (0 + Days | Subject)
   Data: sleepstudy
  AIC  BIC logLik MLdeviance REMLdeviance
 1752 1764 -871.8       1752         1744
Random effects:
 Groups   Name        Variance Std.Dev.
 Subject  (Intercept) 627.508  25.0501
 Subject  Days         35.858   5.9881
 Residual             653.590  25.5654
number of obs: 180, groups: Subject, 18; Subject, 18

Fixed effects:
            Estimate Std. Error t value
(Intercept)  251.405      6.885   36.51
Days          10.467      1.560    6.71
```

Back to Piepho's Data:
Fitting Model 1

```
> library(lme4)
> peipho <- read.table("peipho.data",head=T)
> peipho$year = factor(peipho$year)
> peipho$site = factor(peipho$site)
> peipho.fit1 <- lmer2( y ~ wyr + (1|year) + (wyr|site),data=peipho)

> peipho.fit1
Linear mixed-effects model fit by REML
Formula: y ~ wyr + (1 | year) + (wyr | site)
   Data: peipho
  AIC  BIC logLik MLdeviance REMLdeviance
 1068 1081 -527.9       1077         1056
Random effects:
 Groups   Name        Variance  Std.Dev. Corr
 year     (Intercept)  204370.4  452.073
 site     (Intercept) 1504880.9 1226.736
          wyr            2768.3   52.615 -0.975
 Residual              140992.2  375.489
Number of obs: 70, groups: year, 14; site, 5

Fixed effects:
            Estimate Std. Error t value
(Intercept)  1771.51     616.26   2.875
wyr           -64.00      32.37  -1.977

Correlation of Fixed Effects:
```

```
    (Intr)
wyr -0.909
> anova(peipho.fit1,peipho.fit2)
Data: peipho
Models:
peipho.fit1: y ~ wyr + (1 | year) + (wyr | site)
peipho.fit2: y ~ wyr + site + (1 | year) + (wyr - 1 | site)
              Df     AIC     BIC  logLik  Chisq Chi Df Pr(>Chisq)
peipho.fit1.p  6 1089.46 1102.95 -538.73
peipho.fit2.p  8 1084.92 1102.91 -534.46 8.5401      2    0.01398
```

I used the following code to fit model (2).

```
> peipho.fit2 <-  lmer( y ~ wyr + site + (1|year) + (wyr-1|site), data = peipho)
> summary(peipho.fit2)
Linear mixed-effects model fit by REML
Formula: y ~ wyr + site + (1 | year) + (wyr - 1 | site)
   Data: peipho
      AIC      BIC    logLik MLdeviance REMLdeviance
 1014.018 1034.254 -498.0088   1068.920     996.0175
Random effects:
 Groups   Name        Variance Std.Dev.
 year     (Intercept) 204340.0 452.040
 site     wyr           2768.1  52.613
 Residual             140995.5 375.494
# of obs: 70, groups: year, 14; site, 5

Fixed effects:
             Estimate Std. Error DF t value  Pr(>|t|)
(Intercept)  3688.990    337.717 64 10.9233 2.898e-16
wyr           -64.000     32.373 64 -1.9769   0.05236
site2       -2732.492    296.937 64 -9.2023 2.512e-13
site3       -2563.621    296.937 64 -8.6335 2.476e-12
site4       -1520.514    296.937 64 -5.1207 3.010e-06
site5       -2770.761    296.937 64 -9.3311 1.500e-13

> anova(peipho.fit2)
Analysis of Variance Table
             Df   Sum Sq  Mean Sq    Denom F value    Pr(>F)
wyr           1   551028   551028       64  3.9082   0.05236
factor(site)  4 18051102  4512776       64 32.0074 1.199e-14
```

Note on degrees of freedom. Bates used to put in the number of observations minus the number of parameters as the df, and included p-values based on a $t(df)$ distribution. He qualifies his choice with the disclaimer that, "These df are an upper bound." The F ratios built for these comparisons do not have F distributions, even under the null hypothesis. Some statisticians (Kenward and Roger 1997) have gone searching for a good choice of denominator df for an F distribution which approximates the distribution of the ratio under the null, but others argue that it's overly optimistic to expect to find an easy approximation for small sample sizes, and we should focus on simulating the distributions or using a Baeysian approach.

Conclusions

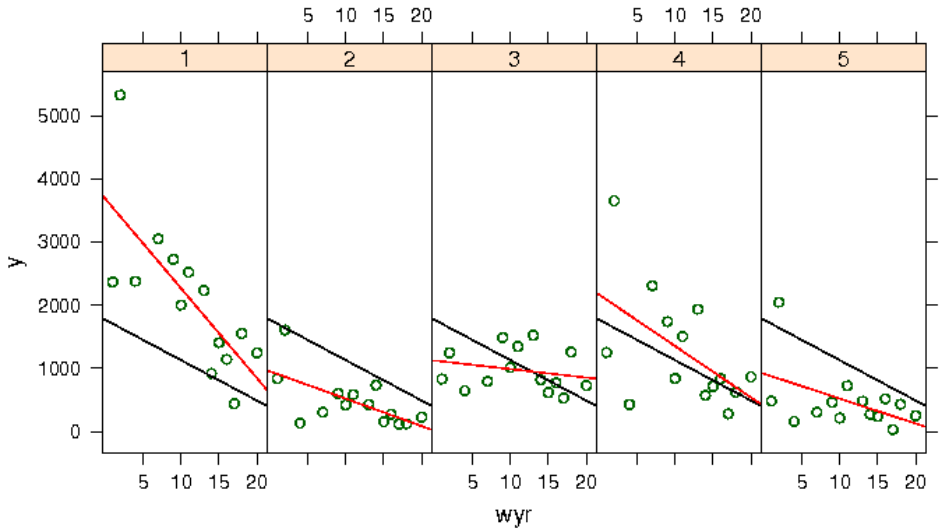
Under either model, we have a significant overall (negative) trend in counts, which is most interesting to game managers. The difference between sites is strong (and would be

expected, one site is the Mara game preserve). Sites do have different slopes, but I would argue that site intercepts and slopes are fixed effects because there are only a fixed number of these sites.

```
> peipho.fit3
Linear mixed-effects model fit by REML
Formula: y ~ wyr * site + (1 | year)
   Data: peipho
 AIC   BIC logLik MLdeviance REMLdeviance
 973 997.7 -475.5       1047          951
Random effects:
 Groups   Name        Variance Std.Dev.
 year     (Intercept) 204332   452.03
 Residual             140997   375.50
Number of obs: 70, groups: year, 14

Fixed effects:
            Estimate Std. Error t value
(Intercept)  3783.94     342.05  11.063
wyr          -150.65      27.10  -5.560
site2       -2852.45     309.10  -9.228
site3       -2719.90     309.10  -8.800
site4       -1594.33     309.10  -5.158
site5       -2895.44     309.10  -9.367
wyr:site2     109.47      24.49   4.471
wyr:site3     142.63      24.49   5.825
wyr:site4      67.37      24.49   2.751
wyr:site5     113.78      24.49   4.647

Correlation of Fixed Effects:
          (Intr) wyr    site2  site3  site4  site5  wyr:s2 wyr:s3 wyr:s4
wyr       -0.888
site2     -0.452  0.401
site3     -0.452  0.401  0.500
site4     -0.452  0.401  0.500  0.500
site5     -0.452  0.401  0.500  0.500  0.500
wyr:site2  0.401 -0.452 -0.888 -0.444 -0.444 -0.444
wyr:site3  0.401 -0.452 -0.444 -0.888 -0.444 -0.444  0.500
wyr:site4  0.401 -0.452 -0.444 -0.444 -0.888 -0.444  0.500  0.500
wyr:site5  0.401 -0.452 -0.444 -0.444 -0.444 -0.888  0.500  0.500  0.500
```

Note: you can suppress printing of the correlations with `print(peipho.fit3, corr=FALSE)`

The augmented prediction plot looks like this:



The black line is common to all sites, the red lines indicate site-specific lines.

Peipho did his analysis in SAS Proc Mixed using separate random statements for year and for site. Code:

```
proc mixed method=reml nobound;
class year site;
model y =wyr site/ddfm=satterth s;
random int/sub=year;
random wyr/sub=site;
run;
```

References

Arnold, S. *The Theory of Linear Models and Multivariate Analysis*. Wiley (1981).

Benjamini, Y. and Hochberg, Y. "Controlling the false discovery rate: a practical and powerful approach to multiple testing." *JRSS B*, 57 (1995).

Box, G. E. P. and Cox, D. R. "An Analysis of Transformations." *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252 (1964).

Buhlmann, P. and Hothorn, T. "Boosting Algorithms: Regularization, Prediction, and Model Fitting." *Statistical Science*, 22(4):477–505 (2007).

Burnham, K. P. and Anderson, D. R. *Model Selection and Multimodel Inference: a Practical Information-theoretic Approach*. Springer-Verlag Inc (2002).

Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. New York: Academic Press, 2nd edition (1988).

Cook, R. D. "Fisher Lecture: Dimension Reduction in Regression." *Statistical Science*, 22(1):1–26 (2007).

Davison, A. and Hinkley, D. *Bootstrap Methods and their Application*. Cambridge Press (1997).

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. "Least Angle Regression." *The Annals of Statistics*, 32(2):407–499 (2004).

Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap*. Chapman Hall (1993).

Gałecki, A., Galecki, A., and Burzykowski, T. *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. Springer Texts in Statistics. Springer London, Limited (2013).

Gelman, A., Carlin, J., Stern, H., and Rubin, D. *Bayesian Data Analysis*. Chapman Hall (2003).

Hjort, N. L. and Claeskens, G. "Frequentist Model Average Estimators." *JASA*, 98(464):879–916 (2004).

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. "Bayesian Model Averaging: A Tutorial (with Comments by M. Clyde, David Draper and E. I. George, and a Rejoinder by the Authors." *Statistical Science*, 14(4):382–417 (1999).

Holm, S. "A simple sequentially rejective multiple test procedure." *Scandinavian Journal of Statistics*, 6:65–70 (1979).

Kenward, M. and Roger, J. "Small sample inference for fixed effects from restricted maximum likelihood." *Biometrics*, 53(3):983–997 (1997).

Kutner, M. H., Nachtsheim, C. J., Neter, J., and Li, W. *Applied Linear Statistical Models, Fifth Edition*. McGraw-Hill (2004).

Laird, N. M. and Ware, J. H. "Random-Effects Models for Longitudinal Data." *Biometrics*, 38(4):pp. 963–974 (1982).
URL `http://www.jstor.org/stable/2529876`

Lenth, R. V. "Some Practical Guidelines for Effective Sample Size Determination." *The American Statistician*, 55(3):187–193 (2001).

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. *SAS System for Mixed Models, Second Edition*. SAS Institute, Inc. (2006).

Martell, R. F. and Leavitt, K. "Reducing the Performance-Cue Bias in Work Behavior Ratings: Can Groups Help?" *Journal of Applied Psychology* (2002).

Milliken, G. A. and Johnson, D. E. *Analysis of Messy Data, Volume 1: Designed Experiments*. Van Nostrand Reinhold Co. Inc. (1984).

Monahan, J. F. *A Primer on Linear Models*. Chapman and Hall/CRC (2008).

Piepho, H.-P. and Ogutu, J. O. "A Simple Mixed Model for Trend Analysis in Wildlife Populations." *Journal of Agricultural, Biological, and Environmental Statistics*, 7(3):350–360 (2002).

Pinheiro, J. C. and Bates, D. M. *Mixed-effects Models in S and S-PLUS*. Springer-Verlag Inc (2000).

Sen, A. K. and Srivastava, M. S. *Regression Analysis: Theory, Methods, and Applications*. Springer-Verlag Inc (1997).

Snedecor, G. W. and Cochran, W. G. *Statistical Methods, Seventh Edition*. Iowa State University (1980).

Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S*. Springer-Verlag (2002).

Verbeke, G. and Molenberghs, G. *Linear Mixed Models for Longitudinal Data*. Springer-Verlag Inc (2000).

Verdonck, A., De Ridder, L., Kuhn, R., Darras, V., Carels, C., and de Zegher, F. "Effect of testosterone replacement after neonatal castration on craniofacial growth in rats." *Archives of Oral Biology*, 43(7):551–557 (1998).

Wright, S. P. "Adjusted P-values for simultaneous inference." *Biometrics*, 48:1005–1013 (1992).

Xie, Y. *Dynamic Documents with R and Knitr*. Chapman & Hall/CRC the R series. Taylor & Francis Group (2013).