Supplemental Notes
HES 704: Biostatistical Modeling

Frank E Harrell Jr
Division of Biostatistics and Epidemiology
Department of Health Evaluation Sciences
University of Virginia School of Medicine
fharrell@virginia.edu

March 1, 1999

1 Robust Assessment of Association

If one does not need to make predictions, association can be assessed using rank correlation coefficients. The Spearman rank correlation (ρ) between x and y asks the question "to what extent does y increase (decrease) as x increases?" ρ is identical to the Pearson product–moment linear correlation coefficient r after the data are replaced by their ranks (using midpoints for ties). ρ does not assume linearity, only monotonicity.

When x is dichotomous, the Spearman test for association between x and y is equivalent to the Wilcoxon–Mann–Whitney two–sample test for differences in y by levels of the grouping variable x. The generalization of this test to $k > 2$ levels of x is the Kruskal–Wallis k–sample test. This test can be obtained by the `kruskal.test` function in S–Plus, although it uses a conservative χ^2 approximation. A better test statistic and more accurate P–value can be obtained by doing an ordinary ANOVA on the ranks of y. If there are no missing values, this can be obtained by either of the following two commands.

```
summary(aov(rank(y) ~ x))
summary(lm(rank(y) ~ x))
```

2 Inference for Correlation Coefficients

Rosner described how two correlation coefficients may be compared. This is rarely appropriate, because due to differences in variances, two correlations can be the same whereas slopes may differ significantly. It is usually more appropriate to compare two effects on a "real" scale. This is done by comparing two slopes.

3 Automatic Testing of Contrasts with Design `anova` Function

Two ways of testing hypotheses involving k degrees of freedom, where $k > 1$, are (1) fitting models with and without k parameters and (2) listing the variables describing the k parameters last and adding the sequential sums of squares for the variables in question. There is a third method that is faster if one is willing to use matrix manipulations (or let the computer do it). This method is based on looking at the $\hat{\beta}$s to be tested along with their standard errors and correlations. This approach uses a "contrast matrix" to pick off elements of β to test simultaneously. The `anova` function in the Design library automatically generates many contrasts automatically, to test multiple d.f. hypotheses. Among the automatic tests done by `anova.Design` are tests of total association for each predictor, tests of linearity for each predictor, tests for whether each predict is important either as a main effect or as an effect modifier, and combined tests of global nonlinearity and interaction for the entire model.

4 S-Plus Imputation Functions

See the notes by Alzola and Harrell for an overview of S-Plus imputation functions that are in the `Hmisc` library. Briefly, the `impute` function is used for imputing constant values (see *Predicting Outcomes* for conditions under which this is advisable), and the `transcan` function automatically develops customized imputation rules (multiple regression models using spline functions) to allow each predictor to be predicted from all the other predictors. Here are some simple examples of the `impute` function.

```
cholesterol ← impute(cholesterol)  # replaces NAs with median cholesterol
```

```
age          ← impute(age,mean)     # replace NAs with mean age
sys.bp       ← impute(sys.bp,120)   # replace NAs with constant
race         ← impute(race)         # replace NAs with most frequent category
f ← lrm(y ~ cholesterol + age + sys.bp + race,
         subset=!is.imputed(sys.bp))# exclude subjects w/ imputed sys.bp
```

From the last command you can see that `impute` defines an attribute that allows the user to easily detect which values were imputed. The `describe` function prints the number of imputed values (and the number of remain `NA`'s, if any) for a variable.

A statement such as `age ← impute(age,mean)` causes the original version of the `age` variable to be masked. Some users will want to use e.g. `age.i ← impute(age,mean)`. Whichever method is used, the `Data` area will be cluttered by many derived variables. It's often best to manage S–Plus project areas by making the source data frame be the only place where variables are stored. Free–standing vectors such as `age.i` or a new version of `age` can be removed at the end of the session using the `rm` or `remove` functions[1]. The `Hmisc store` function can facilitate the management of `Data` directories, by redirecting all new objects created during the session to a temporary directory that is deleted at the end of the session. See Alzola and Harrell for details. For example:

```
attach(mydataframe)
store()            # attaches a temporary directory in search pos. 1
age ← impute(age) # new age is in temp. directory,
                   # old one still in mydataframe
f ← lrm(y ~ age)
store(f)           # stores f permanently in _Data
store(f,'fit1')    # stores f permanently in _Data under the name fit1
stores(age, f)     # stores f and age perm. under names f and age
                   # store can deal only with a single variable
```

As a simple example of using the `Hmisc transcan` for developing customized imputation models, and applying these models to impute variables before fitting the response model, consider once again the coronary artery disease problem described above. First we develop 7 additive nonlinear imputation models, one for each predictor, with the other 6 predictors as independent variables:

[1] `remove(edit(objects()))` will let you edit the list of objects to remove from the list of all objects in search position one.

```
imp.models ← transcan(∼ age + sex + LDL + HDL + bp +
                      trig + smoke, imputed=T)
LDL  ← impute(imp.models, LDL)
HDL  ← impute(imp.models, HDL)
trig ← impute(imp.models, trig)
f    ← ols(y ~ age + sex + log(LDL) + log(HDL) + rcs(bp,4) +
           rcs(trig,4) + smoke)
anova(f)      # prints partial test statistics for all variables
              # statistics will be too large because imputation ignored
```

The models estimated by `transcan` (and stored in `imp.models` above) do not assume that any variable can be predicted from any other variable in a linear fashion. When developing a model to estimate, say, predictor 1 from predictors 2,...,7, `transcan` allows both sides of the multiple regression model to be transformed.

When developing the imputation models for, say, predictor 1, `transcan` accounts for the fact that one or more of the "current right hand side" predictors (predictors 2,...,7) may have `NA`'s. Even though subjects for whom the predictor 1 is `NA` must be deleted from the current imputation model fit, predictors 2,...,7 will be imputed using the current working imputation model for them. The whole process is an iterative one which usually takes about 5 iterations to converge.

5 Data Reduction

When the number of predictors is very large in relation to the effective sample size, some data reduction (e.g., using summary scores computed within a cluster of related variables) may be necessary to achieve reliable and interpretable models. An example of this is the ordinal logistic model in Chapter 4 of *Predicting Outcomes*. When the number of predictors is only moderately large, penalized estimation can be used to achieve reliability, and data reduction is probably not worth the effort.

6 Bootstrap

The bootstrap is most frequently used to compute variances, standard deviations, and confidence limits for statistical quantities. The bootstrap is a

resampling method that can (1) estimate the properties (e.g., standard deviation) of statistical estimates (e.g., median, regression coefficients) without having to derive complex mathematical theory and (2) estimate properties without assuming that the distribution or the model you selected is the one that actually fits the data. To estimate variances or standard deviations, 100 bootstrap resamples is usually adequate. To compute nonparametric confidence limits (i.e., without assuming that the statistical estimate is normally distributed) using the empirical quantile method can require 1000 resamples.

We also use the bootstrap to estimate the optimism (bias) in statistical indexes such as R^2. By subtracting the estimated optimism from the apparent R^2 we can obtain a nearly unbiased estimate of R^2. Thus we can estimate the likely performance of our model on a future series of subjects similar to those used in developing the model.

6.1 Bootstrapping in S–Plus

The `sample` function builtin to S–Plus forms the basis of bootstrapping, as it can quickly draw a sample with replacement from a given set of data or from a set of indexes into a full data set. See the notes by Alzola and Harrell for an example S–Plus program. In S–Plus 4.0 there are builtin functions to make bootstrapping even easier. For example, we can obtain a nonparametric 0.95 confidence interval for the population median `age` by typing `summary(bootstrap(age,median))`. To obtain a bootstrap estimate of the standard deviation of a sample mean use `bootstrap(age, mean)`. This standard deviation can be compared with `sqrt(var(age)/length(age))`[2].

7 Estimation and Testing When Not Using Least Squares

When one assumes a normal distribution for $Y|X$ and Y is a traditional continuous variable, the best estimates of β are obtained by minimizing the sum of squared errors. In this case the best test statistic for testing regression coefficients is the F–test. When the test involves a single parameter, this

[2]It is unfortunate that statisticians created the term *standard error* to refer to standard deviations of summary statistics. Note that the standard deviation of a mean is the standard deviation of the raw data divided by $\sqrt{n}$.

can reduce to a t–test (parameter estimate divided by its estimated standard error).

Least squares estimation is a special case of maximum likelihood estimation (MLE). MLE can also handle much more complex models as well as non–normal distributions. With MLE, estimation of unknown parameters is done by solving for the values of the parameters that make the observed data appear to have the highest probability of being observed. In other words, we solve for values of the parameters that are most consistent with the observed data. This requires a trial–and–error approach except when normality is assumed. Under normality the MLE are the least squares estimates.

With MLE the log–likelihood function is the analog of the sum of squared errors. Differences in log–likelihood between full and reduced models can be used to test whether the reduced model is adequate. The χ^2 distribution provides reasonable P–values from these *likelihood ratio* tests. We can also carry out t–like tests using approximate standard errors of estimates that are easily computed during the MLE process. The ratios of parameter estimates to standard errors here are z–statistics because we are assuming normal distributions for parameter estimates. instead of assuming a t distribution. The Wald χ^2 test statistic is the square of the z statistic. There is also a multiple degree of freedom Wald χ^2.

In the `Design` library the `anova` function, when run on a fit from `ols`, produces either F statistics and associated P–values (the default) or Wald χ^2 test statistics and their P–values. The χ^2 statistics equal F–statistics multiplied by their numerator degrees of freedom. The P–values arising from the χ^2 tests will be slightly smaller than had the F (or t) distribution been used. For non–least squares–based models, `anova` for `Design` relies on χ^2 statistics.