Regression Modelling Strategies

Bjarke Hautop Kristensen

May 12, 2024

1 Introduction

These are my notes from reading Frank Harell's book Regression Modelling Strategies (`https://warin.ca/ressources/books/2015_Book_RegressionModelingStrategies.pdf`). The notes are based on things I found interesting and learned. R code and figures are included when relevant.

Contents

2 Issues with step-wise variable selection

Some issues with step-wise variable selection are

- It yields R^2 values that are biased high.
- The F and χ^2 test statistic doesn't have the correct distribution. They were only intended to test prespecified hypotheses.
- The method yields standard errors of regression coefficient estimates that are biased low and confidence intervals for effects and predicted values that are falsely narrow.
- P-values are too small and do not have the proper meaning.
- It provides regression coefficients that are biased high in absolute value and need shrinkage.
- The choice of the variables to be included depends on estimated regression coefficients rather than their true values, and so X_j is more likely to be included if its regression coefficient is over-estimated than if its regression coefficient is underestimated.
- Rather than solving problems caused by collinearity, variable selection is made arbitrary by collinearity.

Instead, one should fit a model and don't change it based on p-values, plots,....

3 Should one assume a linear relationship?

If there is no good reason to assume a relationship is linear simply just don't assume it. Instead, use restricted cubic splines when modeling. In the vast majority of studies, there is every reason to suppose that all relationships involving non-binary predictors are nonlinear. Deciding based on a scatter-plot (or similar methods) on how to represent variables will give confidence limits that are too narrow, P-values too small, R^2 too large, and calibration too good to be true. Since "phantom degrees of freedom" were dismissed during the subjective assessments and ignored when computing standard errors, P-values, and R^2_{adj}.

Instead, to avoid the above issues, prespecify a model and the complexity with which each predictor is represented in the model, without later simplification of the model.

Using restricted cubic splines

- The choice in general is to choose between 3, 4, or 5 knots. For many datasets, 4 knots is a good compromise between flexibility and loss of precision caused by overfitting a small sample.
  - For important variables or large datasets can use 5 knots while for less important variables or small datasets use 3 knots.

4 Missing values

- "Making up" data is better than discarding valuable data.
- Imputation.
  - Single imputation
    - Can use the "best guess", typically the expected values.
    - R function *transcan* does this as default
  - Multiple imputation
    - "approximate Bayesian bootstrap" samples n residuals replacement from the original n estimated residuals (from observations not missing on X_j), then sampling m residuals with replacement from the first sampled set.
    - Thus, more than one random predicted value can be generated for each missing value. To properly account for variability due to unknown values, the imputation is repeated M times, where $M \geq 3$. Each repetition results in a "completed" dataset that is analyzed using the standard method. Parameter estimates are averaged over these multiple imputations to obtain better estimates than those from single imputation.

```r
require(rms)
# Single imputing
w <- transcan(~ ..., imputed=TRUE,
              data=data, pl=FALSE, pr=FALSE)

# Multiple imputing using transcan
w <- transcan(~ ..., imputed=TRUE, n.impute=5
              data=data, pl=FALSE, pr=FALSE)

# Insert imputed values
attach(data)
# Repeat this code for each variable
variable <- impute(w, variable, data = data)

# Multiple imputing using aregImpute (recommended)
w <- aregImpute(~ ..., tlinear = FALSE, data = data)
# Replace fitter with a fit object from rms, such as cph
fit.mult.impute(S ~ ..., fitter, w, data=prostate)
```

Figure 1: Code for single and multiple imputation

A general guideline is if the proportion of observations having any variables missing is below 0.03 can safely use single imputation. Else use multiple imputations. Figure 1 provides code for imputation.

5 Prediction

If one wishes to do classification it's better to develop a good predictive model. It is far better to use the full information in the data to develop a probability model, and then develop classification rules based on estimated probabilities.

```r
set.seed (123)
n <- 50
y <- runif(20*n)
group <- rep(1:20,each=n)
ybar <- tapply (y, group , mean)
ybar <- sort(ybar)
plot(1:20, ybar, type='n', axes=FALSE, ylim=c(.3, .7),
    xlab='Group', ylab='Group Mean')
lines(1:20, ybar)
points(1:20, ybar, pch=20, cex=.5)
axis(2)
axis(1, at =1:20, labels=FALSE )
for(j in 1:20) axis(1, at=j, labels=names(ybar)[j])
abline(h=.5 , col= gray(.85))
```

Figure 2: Code for Figure 3

Shrinkage

When parameter estimates are derived from one dataset and then applied to predict outcomes on an independent dataset, overfitting will cause the slope of the calibration plot (i.e., the shrinkage factor) to be less than one, a result of *regression to the mean*. In ordinary linear regression, we know that all of the coefficient estimates are exactly unbiased estimates of the true effect when the model fits. Each separate coefficient has the desired expectation, however, we tend not to pick out coefficients at random for interpretation, but we tend to highlight very small and very large coefficients, so these will be biased. See Figure 3 for an example.

To counteract this problem we can shrink our estimates. A heuristic shrinkage estimate is given by

$$\hat{\gamma} = \frac{\text{model } \chi^2 - p}{\text{model } \chi^2}, \tag{1}$$

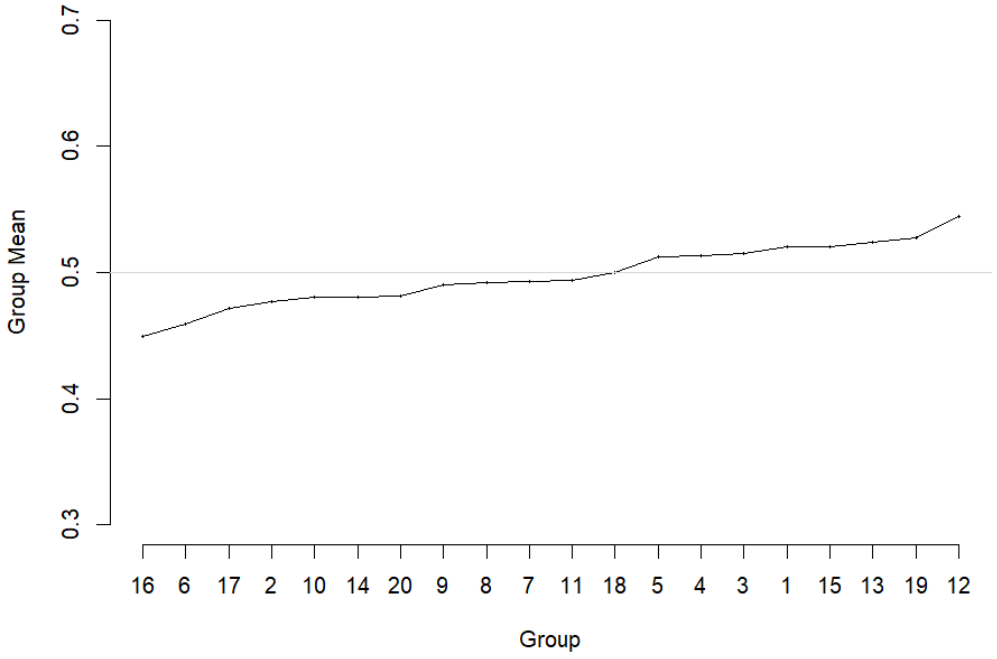

Figure 3: Means from 20 samples of size 50 from a uniform[0, 1] distribution. The reference line at 0.5 depicts the true population value of all of the means

where p is the total degrees of freedom and model χ^2 is the likelihood ratio χ^2 statistic for testing the joint influence of all predictors simultaneously. One can also estimate the shrinkage by bootstrapping as shown in Figure 6

Pre-shrinkage

Just as clothing is sometimes preshrunk so that it will not shrink further once it is purchased, one can in the first place incorporate the shrinkage in the estimation process. The object of shrinking the regression coefficient estimates is to obtain a shrinkage coefficient of $\gamma = 1$ on new data.

Examples of this include ridge regression, lasso regression, and in general penalized maximum likelihood estimation. A method such as cross-validation must be used to choose an optimal penalty factor. An advantage of penalized estimation is that one can penalize different components in the model differently. A drawback of ridge regression and penalized maximum likelihood is that the final model is difficult to validate unbiasedly since the optimal amount of shrinkage is usually determined by examining the entire dataset. Penalization is one of the best ways to approach the "too many variables, too little data" problem.

Data reduction

A good standard requirement on the number of predictors one can allocate for a model to likely be reliable is $p < m/15$, where m is the limiting sample size, for example, n if the response is continuous. If this is not satisfied, and shrinkage is not incorporated into parameter estimation, then data-reduction techniques are recommended.

Data reduction is aimed at reducing the number of parameters to estimate in the model, without distorting statistical inference for the parameters. This is accomplished by ignoring Y during data reduction. Manipulations of X in unsupervised learning may result in a loss of information for predicting Y, but when the information loss is small, the gain in power and reduction of overfitting more than offset the loss.

The preferred way to do data reduction is using domain knowledge. Other methods include

- Redundancy Analysis
  - Remove predictors that are easily predicted from other predictors, using flexible parametric additive regression models. The function *redun* does this.
- Variable Clustering, for example, PCA.

A guideline on how much data reduction is necessary is to fit a full model with all candidate variables, non-linear terms, and hypothesized interactions. Then using 1 if this falls below for example 0.9 we may be concerned with the lack of calibration the model may experience on new data. Either a shrunken estimator or data reduction is needed in this case.

6 Simplifying the final model by approximating it

A model that contains all prespecified terms will usually be the one that predicts the most accurately on new data. It is also a model for which confidence limits and statistical tests have the claimed properties. However, the model includes more predictors than the researchers care to collect in future samples.

Let the final model be the *gold standard* which is specifically used for formal inference. We then proceed to approximate this full model to any desired degree of accuracy. We calculate the accuracy with which it approximates the best model for any approximate model. We can then use the client's desired number of degrees of freedom in this model to approximate the gold standard. This approach has several advantages including

- The client will know how much she loses by dropping predictors.
- If the gold standard model used shrinkage the approximate model will inherit this property.
- Can be used to decode black boxes.
- We can use backwards step-down from the gold standard. So start with the model with $(R^2 = 1)$ to approximate the model.

Code for model approximation is given in Figure 4. For visualizing the result of the approximate model compared to the full model use the code given in Figure 5.

```r
require(rms)
# Assuming model is f
Z <- predict(f) # Compute linear predictor from full model
# Replace ... with the predictors in the model f. Set sigma=1 to
# avoid issue since perfect fit.
a <- ols(Z ~ ..., sigma =1)
# Do fast backward step-down
fastbw (a, aics =10000)
```

Figure 4: Code for model approximation

7 Validation

Validating and calibrating a model is important to measure predictive performance. Code to do this is given in Figure 6. Validate returns a bootstrap estimate of the slope shrinkage. If the model should be used for prediction one should multiply the coefficients of the model by this estimate.

Measure for predictive performance

Somers' D_{xy} is a general measure of predictive discrimination that can handle binary, ordinal, and censored time-to-event outcome variables. In the binary Y case it is just $2(c - 0.5)$ where c is the concordance probability, that is, the area under the ROC curve. It is a very interpretable measure of predictive discrimination but like AUROC is not sensitive enough for choosing or comparing models. Predictive discrimination is the degree to which predictive signals can separate those with good outcomes from those with worse outcomes.

If one wishes to compare models use log-likelihood, including the likelihood ratio test (LR) and AIC.

```
s <- fastbw(a, aics =10000)
betas <- s$Coefficients
X <- cbind(1, f$x) # design matrix
# Compute the series of approximations to lp
ap <- X %*% t(betas )
# For each approx. compute approximation R^2 and ratio of
# likelihood ratio chi-square for the approximate model to that
# of the full model
m <- ncol(ap) - 1 # all but intercept-only model
r2 <- frac <- numeric(m)
fullchisq <- f$stats['Model L.R.']
for (i in 1:m) {
    lpa <- ap[, i]
    r2[i] <- cor(lpa, Z)^2
    fapprox <- lrm(... ~ lpa, data=...)
    frac[i] <- fapprox$stats['Model L.R.']/fullchisq
}
plot(r2, frac, type = 'b',
    xlab = expression(paste('Approximation', R^2)),
    ylab = expression(paste('Fraction of',
    chi^2, 'Preserved')))
abline(h=.95 , col=gray(.83)); abline(v=.95, col=gray(.83))
abline(a=0, b=1, col=gray(.83))
```

Figure 5: Code for plot of approximate model compared to full model

```r
# Assuming the model is f
v <- validate(f, B=200)
cal <- calibrate(f, B = 200)
plot(cal, subtitles= FALSE)
```

Figure 6: Code for plot validation and calibration

Choosing between cross-validation and bootstrapping

Based on a simulation study done by Frank Harrell at `https://hbiostat.org/doc/simval.html` ordinary bootstrapping performs the best in validating Binary Logistic Regression Models. It is almost always better than .632 bootstrap and is better or equal to cross-validation for all strategies tried.

So general guideline is to use ordinary bootstrapping for validation.

8 Nomograms

Nomograms are the best way to visualize complicated models. Figure 7 gives an example of a nomogram. It is interpreted the following way:

1. Pick a value for each predictor and note the number of points they correspond to.
2. Sum up the points.
3. Read the measure you want according to the total number of points you have.

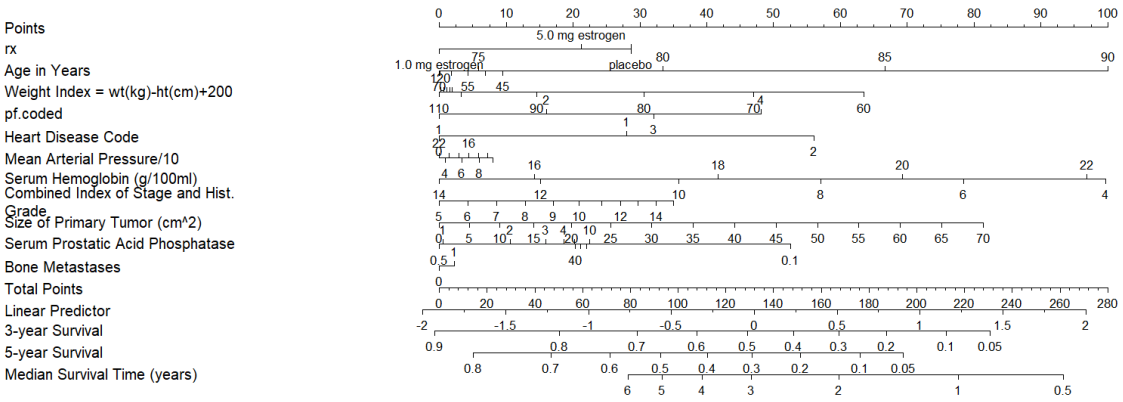

Figure 7: Nomogram