Model Comparison and Out of Sample Prediction

ISLR Chapter 5

Measuring Quality of Fit

- interactive building of models (transformations, interactions, etc)
- Measure of Quality of fit

$$\text{MSE} = \frac{1}{n}\sum_i (Y_i - \widehat{f(x)})^2$$

- OLS: training MSE can be driven smaller by adding more predictors
- How well does model predict for new data Y^*?

$$\text{MSE} = \frac{1}{n^*}\sum_i (Y_i^* - \widehat{f(x_i^*)})^2$$

No guarantee that model that has lowest training MSE will be the best out of sample MSE

Bias - Variance Trade Off

MSE for Prediction

$$\mathrm{E}[(Y^* - \widehat{f(x^*)})^2] = \mathrm{Var}(\widehat{f(x^*)}) + [\mathrm{Bias}(\widehat{f(x^*)})]^2 + \mathrm{Var}(\epsilon^*)$$

- Variance: amount $\widehat{f(x)}$ varies if we use a different training set
- Bias: error that is introduced by approximating model by a potentially simpler model
- Variance of error: irreducible error (Normal models)

Want a statistical procedure that has low variance and low bias

Need a test set

Evaluation on Test Set

- Split the data into 2 groups: training and test

```
set.seed(8675309)
n.train = floor(.75*n)   # 75% training
train = sample(1:n, size=n.train, rep=F)
hiv.train = hiv[train,]
hiv.test = hiv[-train,]
```

- Fit model to training data
- Predict on test data
- Evaluate MSE (or root MSE) on test data

```
rmse = function(y, ypred) {
  rmse = sqrt(mean((y - ypred)^2))
  return(rmse)
}
```

HIV Example Poisson Model

```
hiv.train.glm = glm(fupacts~ bs_hiv + log( bupacts + 1) +
                              sex + couples + women_alone,
                    data=hiv.train, family=poisson)
poi.yhat.train = predict(hiv.train.glm)
poi.yhat.test = predict(hiv.train.glm, newdata=hiv.test)
rmse(hiv.train$fupacts, poi.yhat.train)
```

```
## [1] 29.15457
```

```
rmse(hiv.test$fupacts, poi.yhat.test)
```

```
## [1] 32.08803
```

Negative Binomial Model

```
hiv.train.nb = glm.nb(fupacts ~ bs_hiv + log(bupacts + 1) +
                                  sex + couples + women_alone,
                      data=hiv.train)
nb.yhat.train = predict(hiv.train.nb)
nb.yhat.test = predict(hiv.train.nb, newdata=hiv.test)

rmse(hiv.train$fupacts, nb.yhat.train)
```

```
## [1] 29.16413
```

```
rmse(hiv.test$fupacts, nb.yhat.test)
```

```
## [1] 32.09507
```

Predictions are not that different between the two models

What about other test sets? K-fold cross validation

- Split data into K groups
- first fold is test (or validation) set
- remaining $K-1$ folds are training data
- MSE_1 obtained from prediction on test set

Repeat for each fold giving K MSE's

$$\text{CV}_K = \frac{1}{K}\sum_i \text{MSE}_i$$

Usually $K = 5$ or $K = 10$

Special Case is Leave-one-out- Cross Validation

HIV with 10 fold Cross-Validation

```
library(boot)

hiv.glm.poi = glm(fupacts~ bs_hiv + log( bupacts + 1) +
                            sex + couples + women_alone,
                  data=hiv, family=poisson)
cv.glm(hiv, hiv.glm.poi, cost=rmse, K=10)$delta
```

```
## [1] 23.30877 23.28011
```

```
hiv.glm.nb = glm.nb(fupacts~ bs_hiv + log( bupacts + 1) +
                            sex + couples + women_alone,
                  data=hiv)
cv.glm(hiv, hiv.glm.nb, cost=rmse, K=10)$delta
```

```
## [1] 23.02306 22.99372
```

NB slightly better

Coverage

Coverage is the nominal probability

$$P(Y^* \in (L, U)) \geq (1 - \alpha)$$

Does model achieve the desired coverage?

Estimate Predictive Coverage (empirical coverage)

```
coverage = function(y, pi) {
 mean(y >= pi[,1] & y <= pi[,2])
}
```

Use simulation to obtain the Prediction Interval

Prediction Interval Function for Negative Binomial

```
pi.nb = function(object, newdata, level=.95, nsim=10000) {
  require(mvtnorm)
  n = nrow(newdata)
  X = model.matrix(object, data=newdata)
  beta = rmvnorm(nsim, coef(object), vcov(object))  # use (
  theta = rnorm(nsim, object$theta, object$SE.theta)
  y.rep = matrix(NA, nsim, n)

  for (i in 1:nsim) {
    mu = exp(X %*% beta[i,])
    y.rep[i,] = rnegbin(n, mu=mu, theta=theta[i])
  }

  pi = t(apply(y.rep, 2, function(x) {
                          quantile(x, c((1 - level)/2,
                                        .5 + level/2))}))
  return(pi)
}
```

Negative Binomial Coverage

```
K = 10
f  = ceiling(n/K)  # number of samples in each fold
folds = sample(rep(1:K, f), n)
NB.coverage = rep(NA, K)

for (i in 1:K) {
 hiv.train = hiv[folds != i,]
 hiv.test  = hiv[folds == i,]
 hiv.train.NB =  glm.nb(fupacts ~ bs_hiv + log(bupacts + 1)
                              sex + couples + women_alone,
                    data=hiv.train)
 pi = pi.nb(hiv.train.NB, hiv.test)
 NB.coverage[i] = coverage(hiv.test$fupacts, pi)
}
mean(NB.coverage)
```

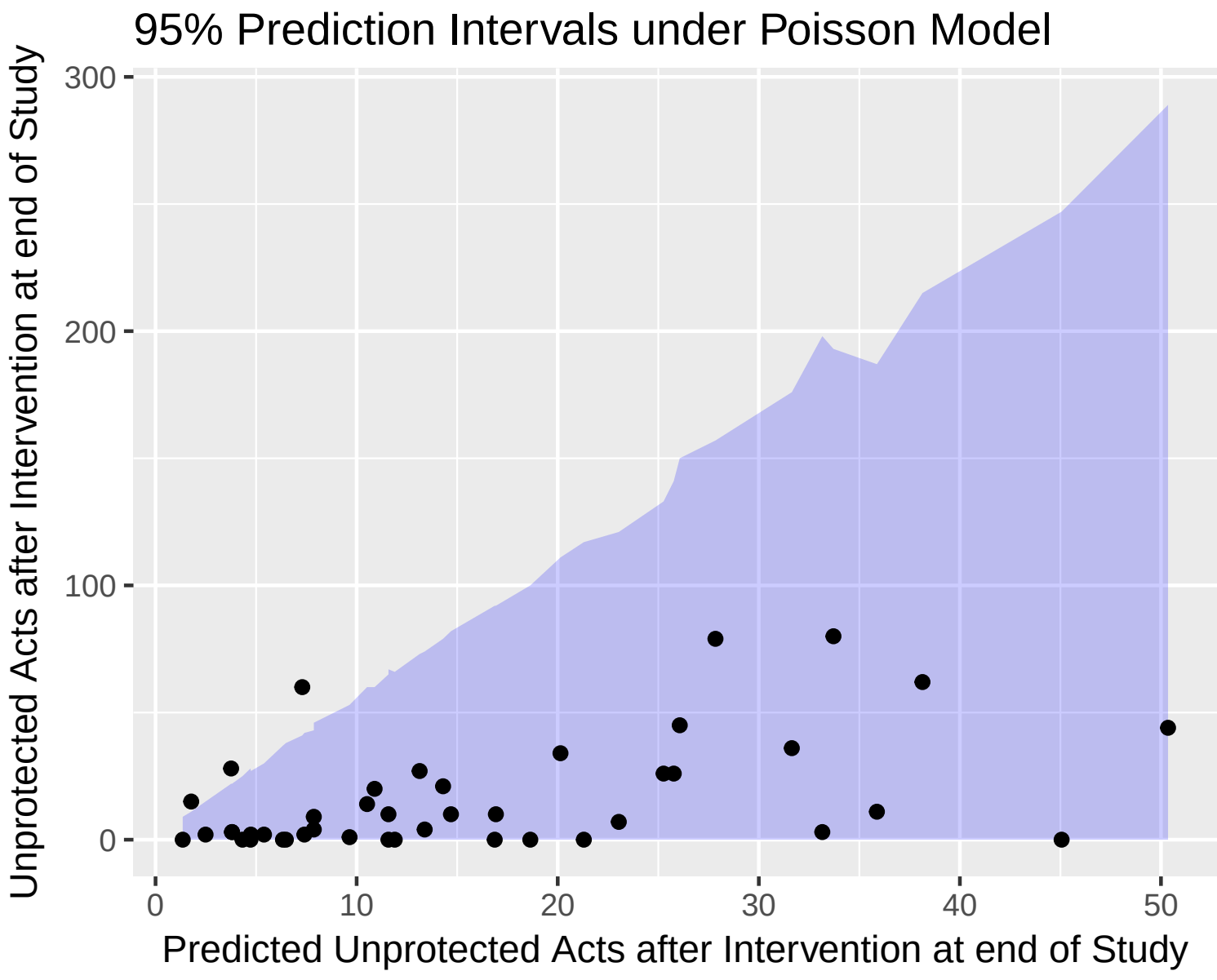
```
## [1] 0.9722466
```

Prediction interval from last fold NB model (code)

```
df = data.frame(fupacts = hiv.test$fupacts,
                pred = predict(hiv.train.NB,    # training m
                               hiv.test,        # test data
                               type="response"),  # type o
                lwr = pi[,1], upr=pi[,2])
df = df %>% arrange(pred)    # sort by prediction

# http://stackoverflow.com/questions/14033551/r-plotting-c
gp = ggplot(df, aes(x=pred, y=fupacts)) +
     geom_ribbon(aes(ymin = lwr, ymax = upr),
                 fill = "blue", alpha = 0.2) +
     geom_point(aes(y=fupacts)) +
 xlab("Predicted Unprotected Acts after Intervention at end
 ylab("Unprotected Acts after Intervention at end of Study"
 ggtitle("95% Prediction Intervals under Poisson Model")
```

Prediction Interval Plot



Poisson Coverage

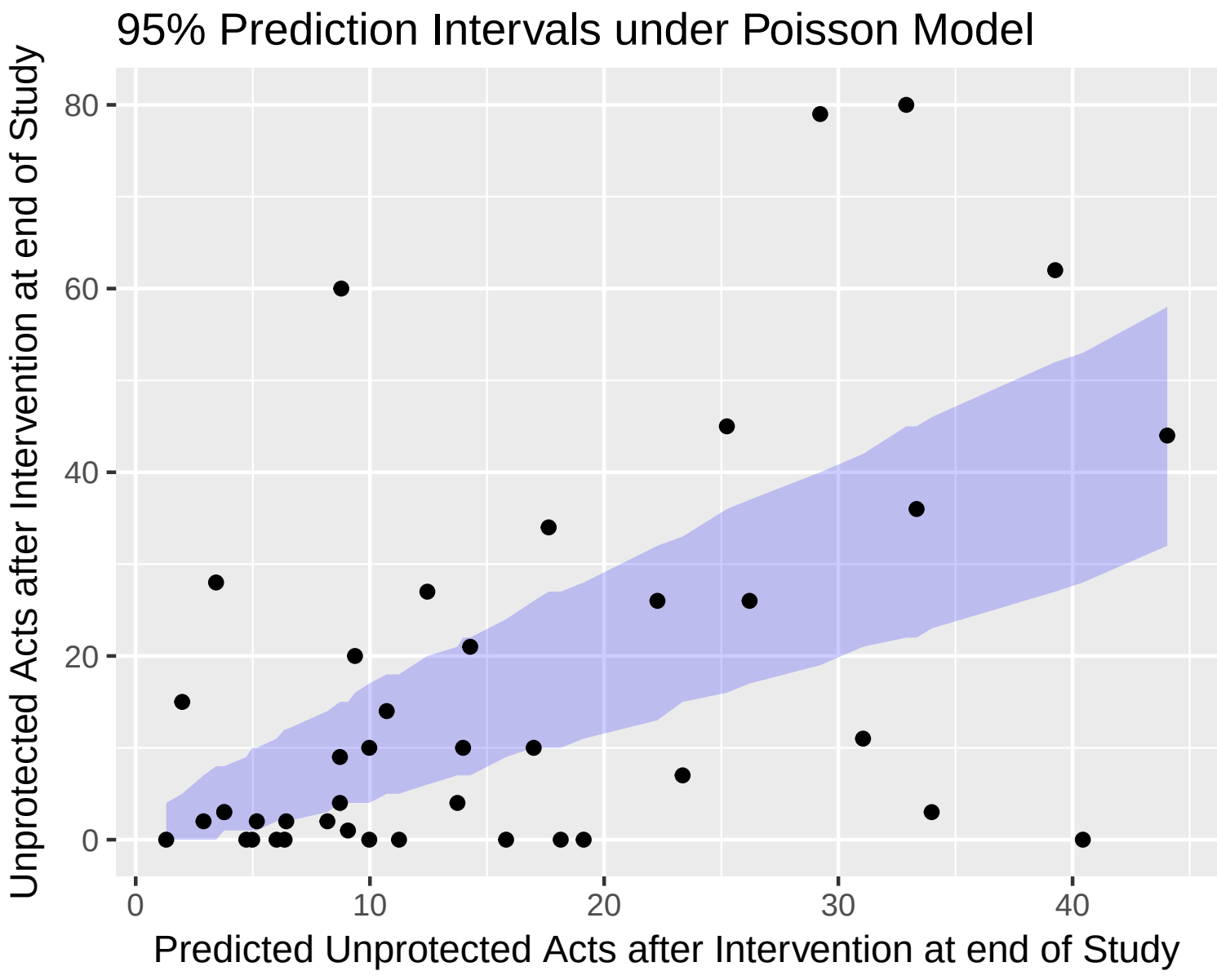
```
poi.coverage = rep(NA, K)

for (i in 1:K) {
 hiv.train = hiv[folds != i,]
 hiv.test  = hiv[folds == i,]
 hiv.train.poi = glm(fupacts ~ bs_hiv + log(bupacts + 1) +
                       sex + couples + women_alone,
                  data=hiv.train, family=poisson)
 pi = pi.poi(hiv.train.poi, hiv.test)
 poi.coverage[i] = coverage(hiv.test$fupacts, pi)
}

mean(poi.coverage)
```

```
## [1] 0.3711794
```

Prediction interval from last fold Poisson model



Summary

- predictions of Poisson and Negative Binomial are very similar - very little difference in RMSE (not impacted by overdispersion)
- Prediction Intervals and Coverage do take into account (over) dispersion
- Think about meaningful "cost" function for comparing models
- Can overdispersion be reduced by adding other predictors?
- a well calibrated method for model fitting should have in-sample and out-of-sample costs that are similar.

More on Predictive Checks:

- Define $T(y)$ or $T(y, \theta)$ a function to measure model adequacy (scalar summary) may be a function of y only
- Generate draws of $\theta^{(r)}$
- For each sample r, draw replicate $Y's$ from the model $Y^{rep\,r} \mid \theta^r$ and evaluate $T(y^{rep\,r}, \theta^r)$
- If discrepancy depends on θ, evaluate $T(y^{obs}, \theta^r)$ for each r
- plot the observed discrepancy $T(y^{obs}, \theta^r)$ versus the replicate discrepancy $T(y^{rep\,r}, \theta^r)$. Under the model the points should be scattered around the 45^o line or look at histogram of $T(y^{obs}, \theta^r) - T(y^{rep\,r}, \theta^r)$ (should include 0)
- Are these values comparable to the out of sample discrepancies? Think about MSE decomposition.