Reporting your results

Thomas Chadefaux

Step 1: Report your overall regression results

First let's import some example data. These data are about admission into graduate school.

```
data <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
head(data)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

Let us now estimate our model and report the regression results, export the table to a word file

```
# start with a simple model
logit1 <- glm(admit ~ gre, data = data, family='binomial')
# Add another variable
logit2 <- glm(admit ~ gre + gpa, data = data, family='binomial')
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(logit1, logit2,  type = 'html', out = 'mytable.doc')
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Thu, Feb 20, 2020 - 23:33:37

Step 2: Run some predictions

First I need to select a learning set and a test set. Here I will use the cross-validation approach, whereby I learn from half the sample and test on the other half

```
runSomePredictions <- function(mydata, reportPlots){ # Wrap this all into a function -- will prove usef

  mydata$predictions <- NA
  mydata$predictions.base <- NA
  testObservations <- sample(1:nrow(mydata), size = nrow(mydata)/2, replace = FALSE) # chooses half the

  learningSet <-  mydata[-testObservations, ]
  testSet <- mydata[testObservations,]

  # Run the models on the learning set
```

Table 1:

	Dependent variable:	
	admit	
	(1)	(2)
GRE	0.004***	0.003**
	(0.001)	(0.001)
GPA		0.755**
		(0.320)
Constant	−2.901***	−4.949***
	(0.606)	(1.075)
Observations	400	400
Log Likelihood	−243.028	−240.172
Akaike Inf. Crit.	490.056	486.344
Note:	*p<0.1; **p<0.05; ***p<0.01	

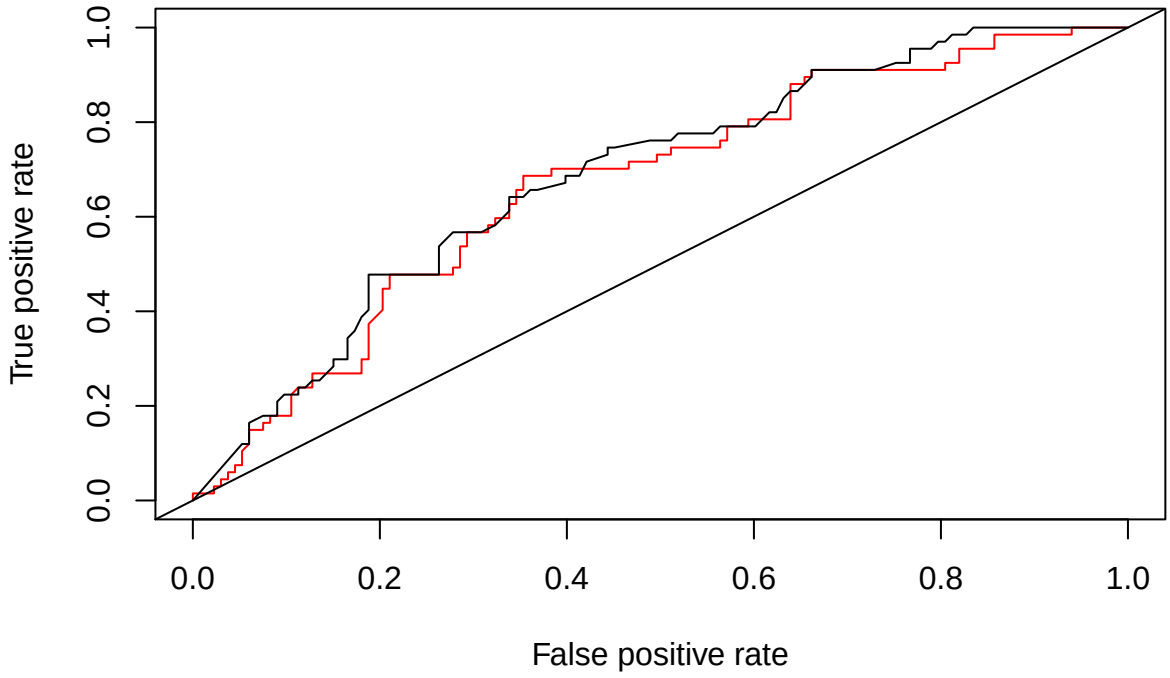
```
learningModel <- glm(admit ~ gre + gpa, data = learningSet, family='binomial')
learningModel.baseline <- glm(admit ~  gpa, data = learningSet, family='binomial', na.action=na.exclu

# Predict on the test set
mydata$predictions[testObservations] <- as.numeric(predict(learningModel, newdata = testSet))
mydata$predictions.base[testObservations] <- as.numeric(predict(learningModel.baseline, newdata = tes

# Calculate metrics
library(ROCR)
pred <- prediction(mydata$predictions, mydata$admit)
pred.base <- prediction(mydata$predictions.base, mydata$admit)

# ROC curve and plot it
roc <- performance(pred, measure = "tpr", x.measure = "fpr")
roc.base <- performance(pred.base, measure = "tpr", x.measure = "fpr")
if(reportPlots == TRUE){
  plot(roc, col=2)
  abline(a=0, b=1)

  plot(roc.base, col=1, add=T)
}


# Precision-recall curve
pr <- performance(pred, measure = "prec", x.measure = "rec")
pr.base <- performance(pred.base, measure = "prec", x.measure = "rec")
if(reportPlots == TRUE){
  plot(pr)
  plot(pr.base, col=2)
}
```

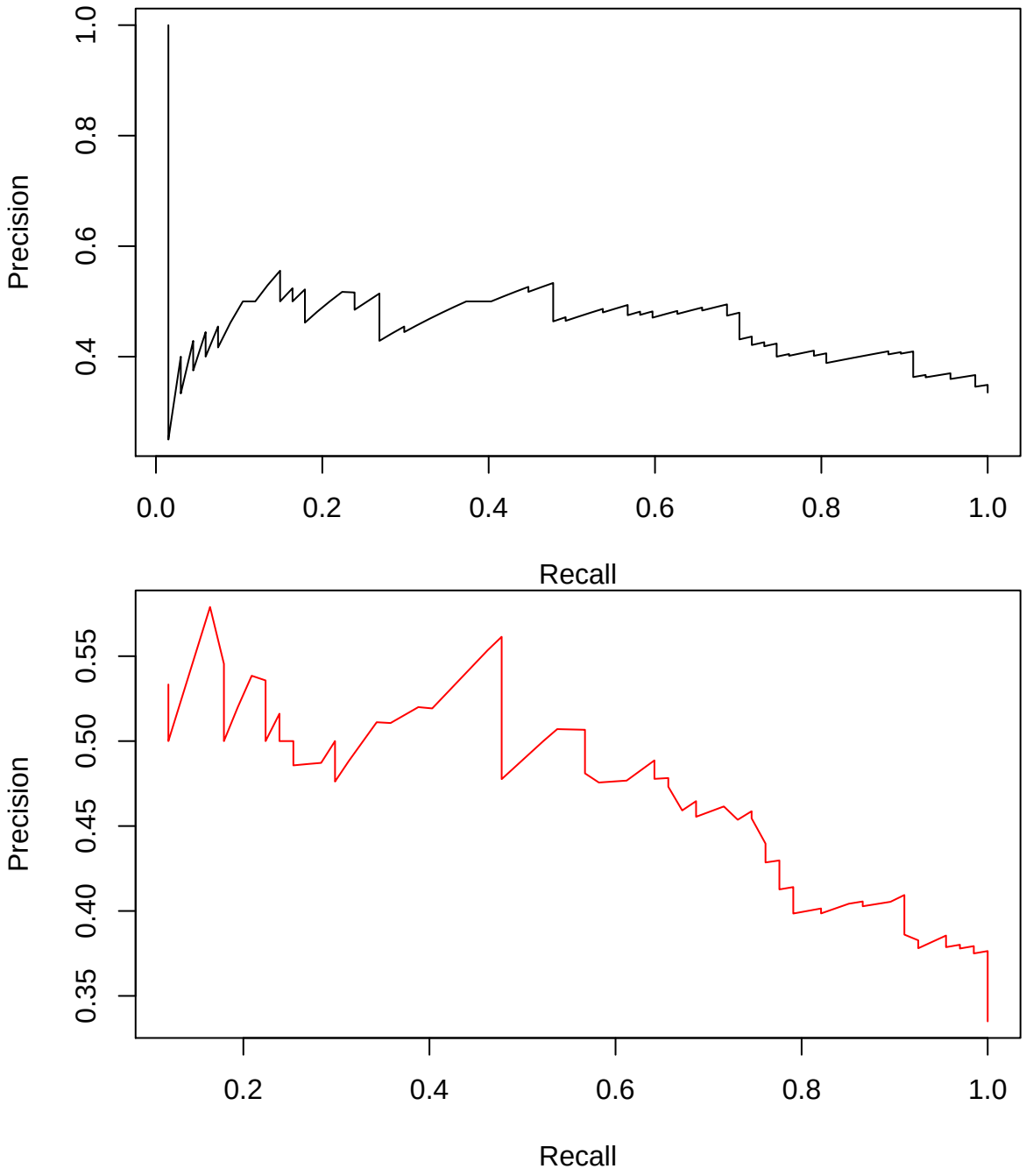
```
  # Calculate area under the ROC
  auc <- performance(pred, measure = "auc")
  this.auc <- auc@y.values[[1]]

  # same thing, but for base model
  auc.base <- performance(pred.base, measure = "auc")
  this.auc.base <- auc.base@y.values[[1]]
  return(c(this.auc.base, this.auc))
}

aucs <- runSomePredictions(mydata = data, reportPlots = TRUE) # call the function we just defined.
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```






```
#(note that the "reportPlots" parameter is just for convenience, namely to turn off the plotting later
```

```
# Now let's see what we got as AUC for the base model:
aucs[1]
```

```
## [1] 0.6869038
```

```
# and for our model:
aucs[2]
```

```
## [1] 0.6689485
```

Great, our area under the ROC is larger, but not that much. Now let's ask whether that difference is

significant. To do that, we'll run exactly the same as above, but resampling every time so we calculate metrics on new data.

Step 3: calculate confidence

```
AUCs.basemodel <- NULL
AUCs.mymodel <- NULL
for(i in 1:1000){
  if(i%%50 == 0) print(i) # just print an update of where we are in the look, every 10 iterations.
  sampleObs <- sample(1:nrow(data), size=nrow(data), replace=T) # sample observations numbers from 1 to
  alternate.world.data <- data[sampleObs, ]
  this.auc <- runSomePredictions(mydata = data, reportPlots = FALSE)
  AUCs.basemodel <- c(AUCs.basemodel, this.auc[1])
  AUCs.mymodel <- c(AUCs.mymodel, this.auc[2])
}
```

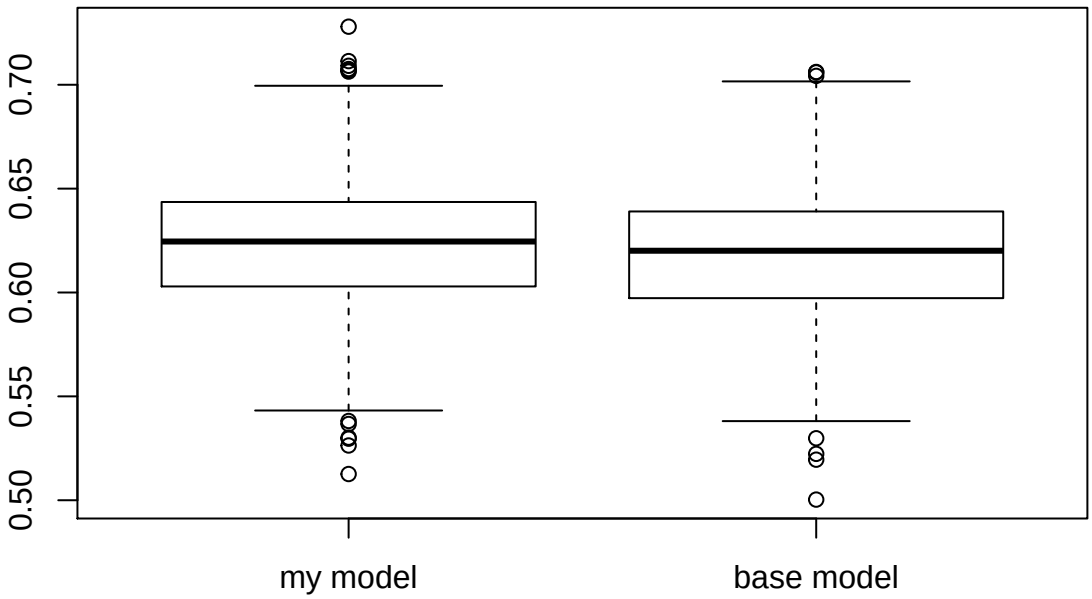
```
## [1] 50
## [1] 100
## [1] 150
## [1] 200
## [1] 250
## [1] 300
## [1] 350
## [1] 400
## [1] 450
## [1] 500
## [1] 550
## [1] 600
## [1] 650
## [1] 700
## [1] 750
## [1] 800
## [1] 850
## [1] 900
## [1] 950
## [1] 1000
```

So now we have 1000 AUCs for the base model, and 1000 AUCs for my model. Let's take a look:

```
head(AUCs.mymodel, 20)
```

```
##  [1] 0.5596852 0.6275030 0.6309168 0.6128008 0.5856245 0.6198291 0.6108686
##  [8] 0.6675712 0.5678424 0.6245014 0.5805631 0.6263187 0.6024161 0.5665615
## [15] 0.5838008 0.6995134 0.6109952 0.6116827 0.6693894 0.6341550
```

```
# let plot the distribution of AUCs for the 2 models:
boxplot(AUCs.mymodel, AUCs.basemodel, names = c('my model', 'base model'))
```




```
# Ok, looks a bit better for my model, but is that difference statistically significant?
t.test(AUCs.mymodel, AUCs.basemodel)
```

```
##
##  Welch Two Sample t-test
##
## data:  AUCs.mymodel and AUCs.basemodel
## t = 3.79, df = 1996.5, p-value = 0.0001551
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.002542904 0.007996697
## sample estimates:
## mean of x mean of y
## 0.6238440 0.6185742
```

```
# yes! (but a very small difference! Report this test in the text, i.e., tell us whether the difference
```