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, report the regression results, and export the output 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')

# Create a nice-looking table, export to Word for
# easy insertion in your dissertation
library(stargazer) # package needed for the table. Install if not already done
```

```
##
## 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 - 23:48:41

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

```
# We'll wrap the whole prediction business into a function
# this will prove useful later when we want to run 1000 of these!
runSomePredictions <- function(mydata, reportPlots){

  # Initialise a variable to store my predictions
  mydata$predictions <- NA
  mydata$predictions.base <- NA
```

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	

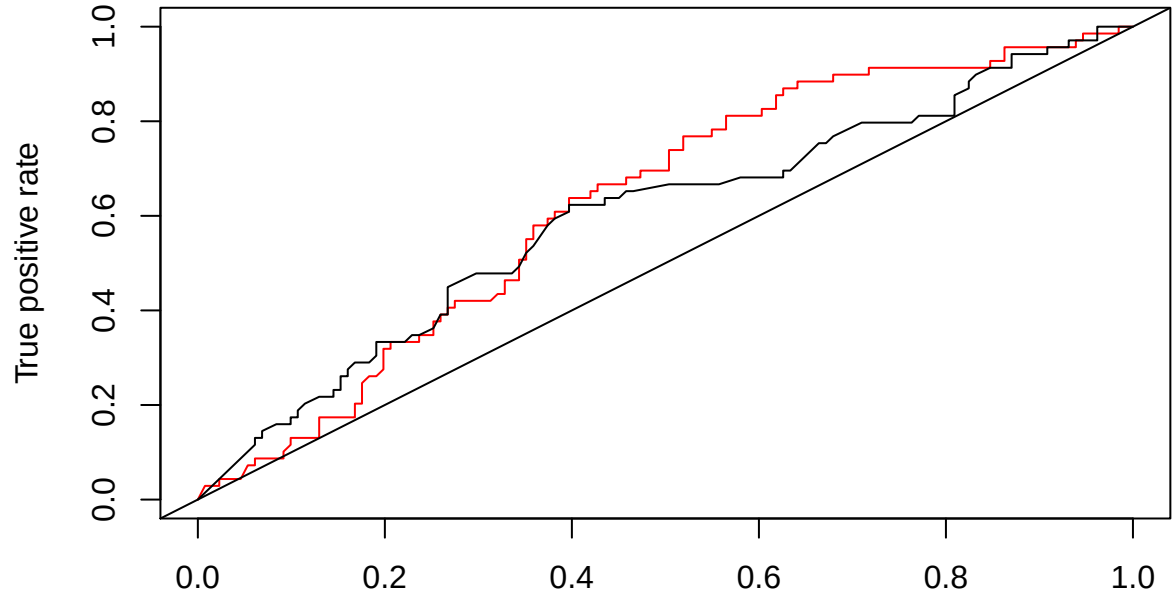
```
# Take a random sample of half the observations
testObservations <- sample(1:nrow(mydata),
                           size = nrow(mydata)/2,
                           replace = FALSE)

# Create a learning set and test set
learningSet <-  mydata[-testObservations, ]
testSet <- mydata[testObservations,]

# Run the models on the learning set
learningModel <- glm(admit ~ gre + gpa,
                     data = learningSet,
                     family='binomial',
                     na.action=na.exclude)
# same thing, but without the variable I care about, i.e., gre:
learningModel.baseline <- glm(admit ~  gpa,
                              data = learningSet,
                              family='binomial',
                              na.action=na.exclude)

# 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")
```

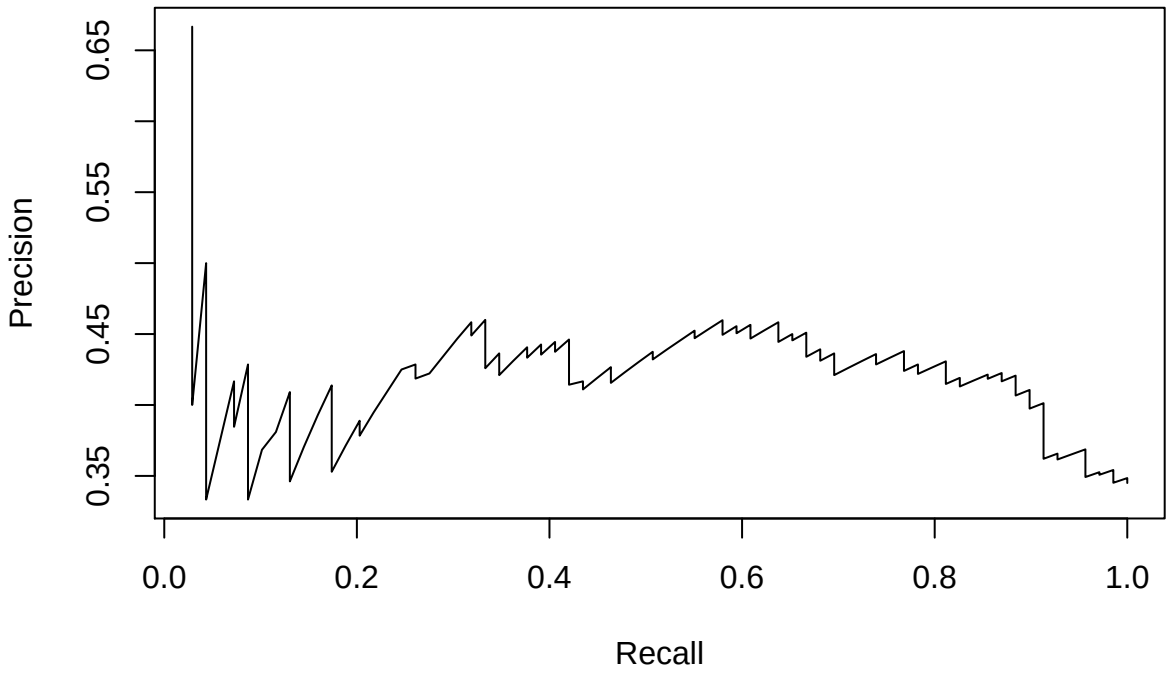
```
  roc.base <- performance(pred.base, measure = "tpr", x.measure = "fpr")

  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)) # return these results so they get stored
}

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

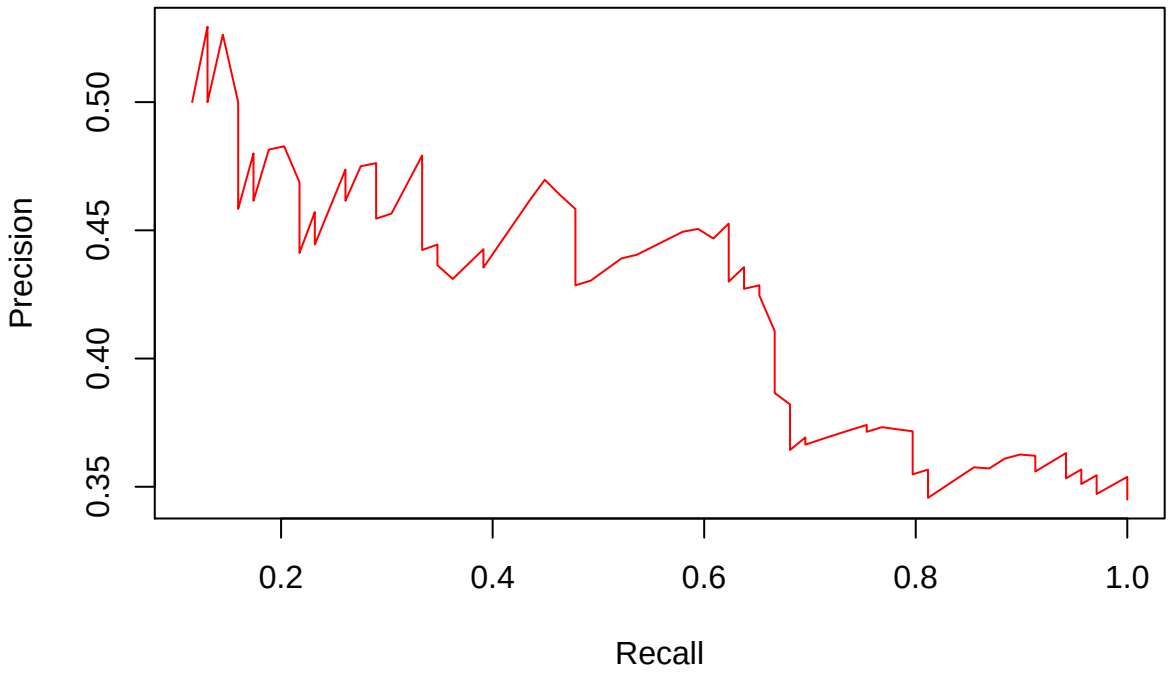
```
## Loading required package: gplots
```

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

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


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


Precision
0.35
0.45
0.55
0.65
0.0
0.2
0.4
0.6
0.8
1.0
Recall




```
# (note that the "reportPlots" parameter is just for convenience,
# namely to turn off the plotting later when I run a loop--
# I don't want to display 1000s of plots!)

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

```
## [1] 0.5980197
```

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

```
## [1] 0.6271711
```

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

```
# Create an empty variable in which we'll store our
# AUCs for each model:
AUCs.basemodel <- NULL
AUCs.mymodel <- NULL

for(i in 1:1000){ # Loop: create a `new' data 1000 times and calculate the AUC for each

  # sample observations numbers from 1 to N, and
  # pick N of them (with replacement)
  sampleObs <- sample(1:nrow(data), size=nrow(data), replace=T)

  # create a new dataset with data from the sample observations
```

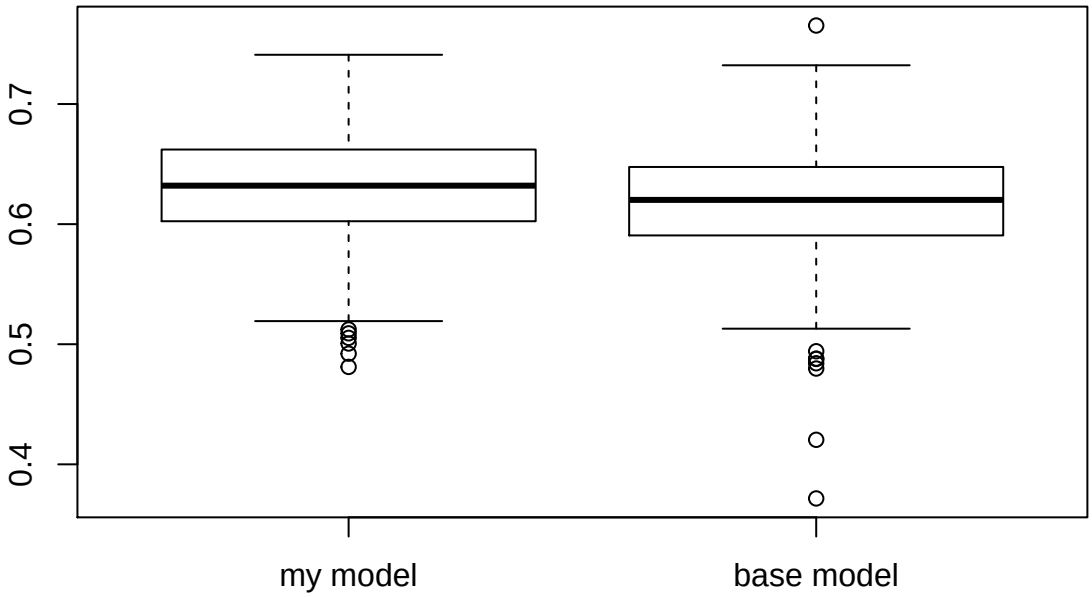
```
  alternate.world.data <- data[sampleObs, ]

  # run the predictions function defined above on that new data
  this.auc <- runSomePredictions(mydata = alternate.world.data, reportPlots = FALSE)
  AUCs.basemodel <- c(AUCs.basemodel, this.auc[1])
  AUCs.mymodel <- c(AUCs.mymodel, this.auc[2])
}
```

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] 0.6225071 0.6729174 0.6310119 0.5917314 0.5958365 0.6245859 0.6119098
##  [8] 0.5973675 0.5929144 0.5566802 0.6867806 0.5833801 0.6419642 0.6999646
## [15] 0.6261489 0.5915097 0.6378571 0.6356589 0.6655271 0.6611912
```

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



Ok, so AUC looks a bit better for my model, but is that difference statistically significant? We need a statistical test, and the obvious one is a t-test:

```
t.test(AUCs.mymodel, AUCs.basemodel)
```

```
##
##  Welch Two Sample t-test
##
## data:  AUCs.mymodel and AUCs.basemodel
## t = 6.0722, df = 1995.5, p-value = 1.507e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.008091108 0.015810802
## sample estimates:
## mean of x mean of y
## 0.6304742 0.6185232
```

The difference is significant, but small. Either way, report this test in the text, i.e., tell us whether the

difference was actually significant or not (and at what significance level))