screenr Tutorial

Steve Gutreuter
sgutreuter@gmail.com

Introduction

The screenr package eases the development and validation of pre-testing screening tools of the sort reviewed by Clemens et al[1], but is applicable to any test-screening problem. Universal testing for a condition can be impractical if the test procedure is relatively expensive and the condition is rare. Alternatively, testing only those subjects having a sufficiently large probability of testing positive may be viable if a screening procedure can be developed which has an acceptable sensitivity. That may be possible if easily measured/recorded risk factors for test positivity can be identified.

This vignette demonstrates the development and validation of such a screening tool using lasso-like $L1$ regularization[2] of logistic models, as implemented in the glmpath package[3]. The screenr package also supports maximum-likelihood estimation of logistic model parameters, but the regularization approach is simpler to use, selects predictor covariates automatically to prevent over-fitting, is robust to correlations among the predictors, and handles any separation of outcomes that might exist for some predictor(s). Model performance is evaluated using receiver-operating characteristics [4, 5, 6].

An (Artificial) Example

Unicorns suffer an epidemic of infection by the Unicorn Immunodeficiency Virus (UIV). There is currently no cure for UIV infection, but highly effective antiretrovial therapy has the potential to block forward transmission and to avert death from opportunistic infections associated with UIV infection. Therefore it is critical that UIV infection is detected, and that infected unicorns are enrolled in treatment. However, UIV infection is sufficiently rare that universal testing of unicorns at health-care entry points is impractical.

A sample of 6000 properly consented adult unicorns were enrolled in a study aimed at evidence-based targeting of testing for UIV infection. The subjects were administered a questionnaire identifying the presence or absence of putative risk factors for UIV infection. The prevalence of UIV is low, and therefore universal testing was deemed impractical. The challenge then is to identify unicorns who should be prioritized for testing. Because UIV is transmissible and fatal if left untreated, it is important that the screening tool have an acceptably high sensitivity. The screenr package enables development and validation of such screening tools.

The screening questionnaire included seven questions which were selected based on published information on infection risk. The data consists of the responses to the screening questions Q1, ..., Q7 (coded as 1 for an affirmative response and 0 for a negative response), and the testing outcome (testresult), again coded 0 and 1 for negative and positive results, respectively.

NOTE: It is critically important that all of the putative risk factors have a *positive* association with the outcome. That is, the questionnaire must be designed so that affirmative (yes) responses are hypothesized to indicate an *increased* risk of UIV infection.

The data from the unicorn study look like:

```
R> ## The first six lines of the unicorn screening data:
R> head(unicorns)
  ID Q1 Q2 Q3 Q4 Q5 Q6 Q7 testresult
1  1  0  0  0  0  0  0  1          0
2  2  0  0  0  0  0  0  0          0
3  3  0  0  0  0  0  0  0          0
4  4  1  0  0  0  0  0  1          0
5  5  1  0  0  1  1  0  1          0
6  6  0  0  0  0  0  0  0          0
```

The prevalence of UIV in the sample of unicorns is 0.016. Under universal testing at health-care entry points, discovery of a single infection would require, on average, testing 62 unicorns.

Screening Tool Development

Screening-tool development and thorough validation consists of four steps:

1. Model fitting
2. Selection of a screening threshold
3. External validation on entirely new data
4. Implementation

In practice, unfortunately, the third step is sometimes omitted due to resource limitations.

Model fitting

The first step is estimation of the logistic-model parameters from the training data. A method known as $L1$ regularization has desirable properties for that task[2, 3]. The function lasso_screenr() provides easy, integrated access to the $L1$ regularization algorithm of Park and Hastie[3], as implemented in the glmpath R package, using a convenient formula interface:

```
R> uniobj1 <- lasso_screenr(testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7,
+                           data = unicorns, Nfolds = 10, seed =  123)


R> class(uniobj1)
[1] "lasso_screenr"
```

The formula `testresult ~ Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7` is an expression of the statement "predict `testresult` from the seven covariates `Q1, ..., Q7`". The argument `Nfolds = 10` specifies the desired number of partitions (folds) to be used for cross validation (discussed under **Selection of a screening threshold**, below).

NOTE: The optional `seed` argument specifies the starting value for the random-number generator used for cross-validation data-splitting. Here it is set to a fixed value to insure reproducibility. By default, the seed is set by the system clock, and therefore results will vary from run to run. Set the seed only if reproducible results are needed.

The fitting algorithm computes logit-scale coefficients along the entire regularization path, and has the effect of selecting covariates based on penalization by $L1$-norm of their

coefficients, similar to the ordinary lasso[2]. Unlike the lasso, the algorithm can also impose a very small penalty on the $L2$-norm, which eliminates adverse effects of any strong correlations among the predictors and provides useful estimates even if the response variable is separable by the predictors (albeit unlikely in test-screening applications). Two solutions along the regularization path which yield the smallest AIC and BIC values, denoted `"minAIC"` and `"minBIC"`, respectively, are given special status. Those are likely the most useful solutions in nearly all settings. However all solutions are accessible.

The resulting lasso_screenr-class object (uniobj1) includes the results from regularization and performance measures from the fits to all of the data, and from k-fold cross validation. uniobj1, contains all of the information need to develop and (internally) validate a screening tool for diagnostic tests.

For example, the logit-scale coefficients for the special solutions are obtained using:

```
R> coef(uniobj1)
```

```
          AIC-best model BIC-best model
Intercept      -7.568603   -6.833257953
Q1              2.767832    2.468497841
Q2              1.560257    1.100266591
Q3              2.891689    2.575660525
Q4              0.000000    0.000000000
Q5              1.822216    1.578669298
Q6              0.363779    0.000656197
Q7              0.843276    0.581431699
```

Note that the coefficients for Q4 have been shrunk to zero in both the AIC- and BIC-best models, which happen to coincide in the unicorn data. In effect, Q4 has been eliminated from both of those models. Q4 was not an important predictor of the test result.

One can also obtain the adjusted odds ratios and/or omit the intercept. For example, the adjusted odds ratios for the seven predictors can be obtained using:

```
R> coef(uniobj1, or = TRUE, intercept = FALSE)
```

```
   AIC-best model BIC-best model
Q1       15.92408       11.80470
Q2        4.76004        3.00497
Q3       18.02373       13.13999
Q4        1.00000        1.00000
Q5        6.18555        4.84850
Q6        1.43876        1.00066
Q7        2.32397        1.78860
```

The adjusted odds ratios for Q4 are 1.0, again indicating that Q4 was not predictive of the test result.

One can also examine the coefficients everywhere the active set changed along the regularization path for either of the special fits by extracting the regularization path object.

```
R> pathobj <- get_what(from = uniobj1, what = "glmpathObj", model = "minAIC")
```

Figure 1 is produced using the plot method. The horizontal axis is the $L1$-norm of the coefficient vector and the vertical axis shows values of the individual coefficients (see [3] for details).

The screenr package provides the following methods for lasso_screenr-class objects:

```
R> methods(class = "lasso_screenr")
```

```
R> plot(pathobj)
```

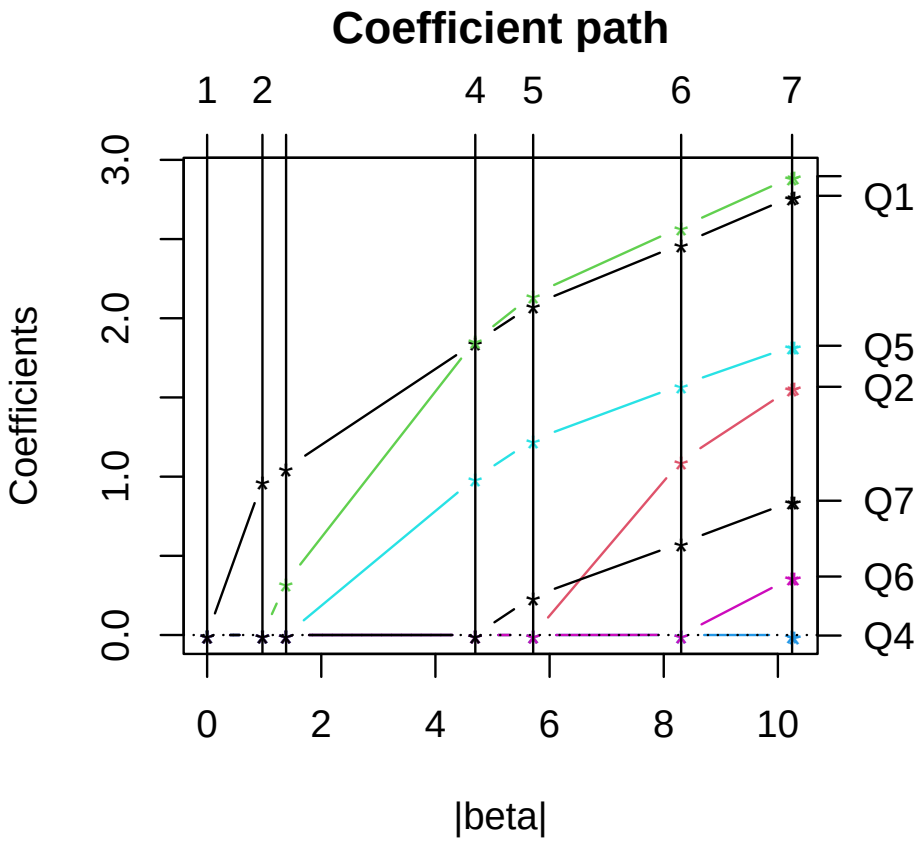


Fig 1. Plot of model coefficient values along the regularization path.

```
[1] coef     get_what ntpp     plot     predict  print    summary
see '?methods' for accessing help and source code
```

The methods should make it mostly unnecessary to access components of lasso_screenr-class objects directly using R code similar to `object\$component`.

IMPORTANT: The object returned by lasso_screenr() (or any of the other "_screenr" functions) contains information critical to validation and implementation of test screening. Therefore **the object should be saved for future use**, for example:

```
R> saveRDS(uniobj1, file = "uniobj1.rds" )
```

Selection of a screening threshold

The next task is identifying whether sufficiently effective pre-test screening is possible and, if so, selecting the most appropriate screening threshold. That task requires careful consideration and expert judgment, neither of which are provided by the screenr package. However the screenr package does provide the results that are most relevant to that task.

The receiver-operating characteristic (ROC) provides measures of the accuracy of screening at multiple thresholds. The screenr package incorporates k-fold cross-validation to estimate the out-of-sample performance using only the training data (see also External validation on entirely new data, below). The ROC curve is a plot of sensitivity on the false-positive fraction (1 - specificity) or, equivalently, plots sensitivity against specificity with specificity plotted in *decreasing* order. The ROC curves for the unicorns are displayed using the R code to produce Figure 2. Both the overly-optimistic in-sample ROC and the cross-validated

```
R> plot(uniobj1, model = "minAIC")
================================================================================
```

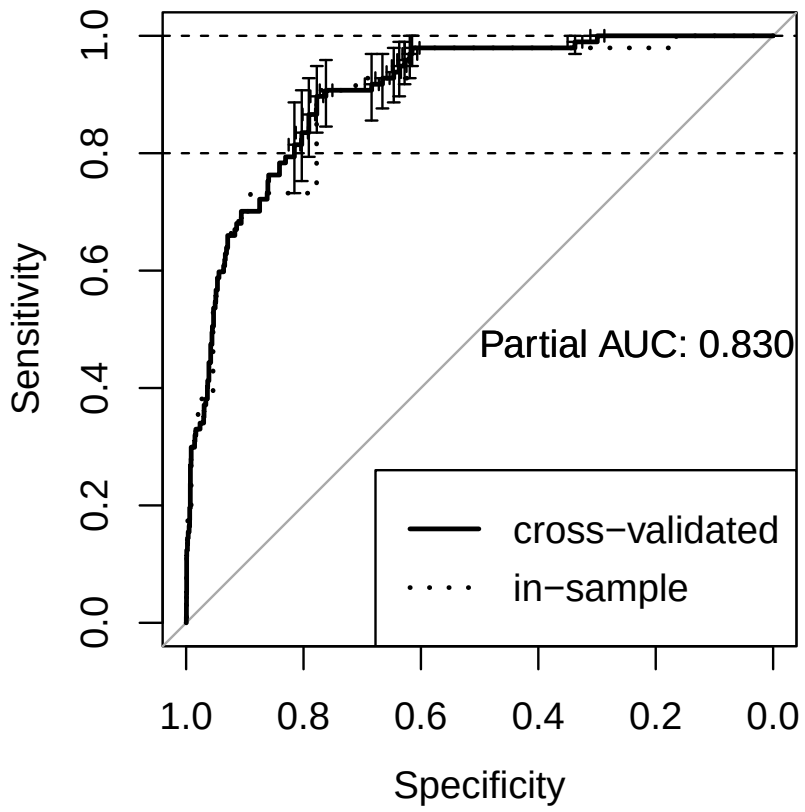


Fig 2. ROC curve from the AIC-best model along the regularization path. The partial AUC was estimated for that portion of the curve for which sensitivity ranges from 0.8 and 1.0.

out-of-sample ROC from the AIC-best model are shown. 95% confidence intervals for sensitivity and specificity are plotted at the local maxima (largest distances perpendicular to the 1:1 line). Those maxima are the complete set options for screening. A partial AUC of 0.5 would be produced from random assignment, and a value of 1.0 indicates perfect classification for that portion of the ROC curve for which sensitivity is at least 0.8.

It is difficult to read sensitivities and specificities from the ROC curve. Instead, the numerical values can be obtained using:

```
R> roc_maximas <- get_what(from = uniobj1, what = "ROCci", se_min = 0.9)
```

```
R> print(roc_maximas)
    Threshold Sensitivity   se.lcl   se.ucl Specificity   sp.lcl   sp.ucl
1  0.00127581    1.000000 1.000000 1.000000    0.299170 0.287820 0.311028
2  0.00131461    0.989691 0.969072 1.000000    0.337456 0.325258 0.349653
3  0.00617520    0.979381 0.948454 1.000000    0.615111 0.602744 0.627308
4  0.00679759    0.969072 0.927835 1.000000    0.618838 0.606641 0.631035
5  0.00685893    0.958763 0.917526 0.989691    0.627732 0.615276 0.639679
6  0.00689048    0.948454 0.896907 0.989691    0.636964 0.624428 0.648992
7  0.00706060    0.938144 0.886598 0.979381    0.646282 0.634084 0.658309
8  0.00729264    0.927835 0.876289 0.979381    0.665255 0.653227 0.677118
9  0.00839102    0.917526 0.855670 0.969072    0.683805 0.672031 0.695579
10 0.01061016    0.907216 0.845361 0.958763    0.761985 0.750970 0.772319
```

The argument `what = "ROCci"` specifies extraction of the threshold probabilities of testing positive and cross-validated sensitivities and specificities along with their 0.95% confidence limits at the local maxima along the cross-validated ROC curve. The argument `se.min = 0.9` limits the local maximas to those which produced a cross-validated sensitivity estimate of at least 0.90. Here there are ten options for screening. For example, testing those unicorns for which the predicted probability of testing positive is 0.00686 (row 5) or larger would have an out-of-sample sensitivity of 0.959 and specificity of 0.628.

For example, suppose we have screening results on two additional unicorns who have not been tested previously. We can compute their probabilities of testing positive using the predict method for lasso_screenr-class objects:

```
R> new_corns <- data.frame(ID = c("Alice D.", "Bernie P."),
+                          testresult = c(NA, NA),
+                          Q1 = c(0, 0), Q2 = c(0, 0), Q3 = c(0, 1),
+                          Q4 = c(0, 0), Q5 = c(0, 1), Q6 = c(0, 0),
+                          Q7 = c(0, 0))
R> new <- predict(uniobj1,  newdata = new_corns )
R> print(new)
         ID testresult Q1 Q2 Q3 Q4 Q5 Q6 Q7 phat_minAIC phat_minBIC
1  Alice D.         NA  0  0  0  0  0  0  0 0.000516147  0.00107618
2 Bernie P.         NA  0  0  1  0  1  0  0 0.054438999  0.06422827
```

If the chosen screening threshold was 0.00686, then testing Bernie P. but screening out Alice D. would have point estimates of sensitivity and specificity of 0.959 and 0.628 if the data from those two unicorns came from the same statistical population as those in the study data.

External validation on entirely new data

The estimates of screening performance based on cross-validation using only the training data will hold for screening on entirely new subjects if and only if those new subjects are from the same statistical population as the subjects in the training sample. However, subjects from other geographic areas, demographic groups and other health facilities may differ from those in the training sample. In that case, cross-validated estimates of sensitivity and specificity might still be overly optimistic. **Therefore it is highly desirable to validate the screening tool prior to use in new populations of subjects**.

External validation requires a repetition of the study for the new population(s). For example, a follow-up study was performed on unicorns who sought care at facilities that were not represented in the screening sample. A total of 3000 new unicorns were interviewed and tested for UIV as part of that follow-up study. External validation consists of assessment of the performance of the screening tool on those new subjects.

The predict method provides the means to predict test-positivity among the new subjects:

```
R> new_preds <- predict(uniobj1, newdata = val_data)
R> head(new_preds)
  ID Q1 Q2 Q3 Q4 Q5 Q6 Q7 testresult phat_minAIC phat_minBIC
1  1  0  0  0  1  0  0  1          0 0.001198689  0.00192323
2  2  0  0  0  1  1  1  1          0 0.010567675  0.00926227
3  3  1  0  0  1  1  1  1          0 0.145355982  0.09939160
4  4  0  0  0  0  0  0  1          0 0.001198689  0.00192323
5  5  0  0  0  0  0  1  0          0 0.000742442  0.00107689
6  6  0  0  0  0  0  0  1          0 0.001198689  0.00192323
```

Note that we are predicting using the model fit from the initial study, as contained in uniobj1. The data frame val_data contains the new data from the follow-up study.

The data frame new_preds contains the results of tests from the new subjects and predicted probabilities of a positive outcome based on the AIC- and BIC-best models fitted to the training data. The questionnaire responses from the new subjects are also included. Those are needed to assess performance of simplified screening based on the sums of question weights.

The performance of the screening tool when applied to the new subjects is again measured using the ROC. The ROC for testing in the new subjects is obtained using the roc function from the pROC package:

```
R> new_roc <- pROC::roc(testresult ~ phat_minAIC, data = new_preds,
+ auc = TRUE)
R> class(new_roc)
[1] "roc"
```

The pROC package provides a plot method for roc-class objects. Figure 3 shows the result. Note that the partial AUC is slightly smaller than the partial AUC from cross-validation of the training data. However the roc_ci function provided by the screenr package provides more informative information, including confidence intervals for sensitivity and specificity:

```
R> new_perf <- roc_ci(new_roc, se_min = 0.8)
R> print(new_perf)
    Threshold Sensitivity   se.lcl   se.ucl Specificity   sp.lcl   sp.ucl
1 0.000000000    1.000000 1.000000 1.000000    0.000000 0.000000 0.000000
2 0.000970565    0.984375 0.953125 1.000000    0.164510 0.151567 0.178474
3 0.006524482    0.921875 0.858984 0.984375    0.613420 0.595368 0.631131
4 0.008154231    0.859375 0.765625 0.937500    0.706744 0.690054 0.723093
5 0.008689097    0.843750 0.750000 0.921875    0.746935 0.731267 0.762602
6 0.009894768    0.812500 0.718750 0.906250    0.756812 0.741144 0.772139
```

Implementation

There are two basic approaches to implementation of test screening based on the results from the unicorn study. The first requires the use of smartphones or tablets, and the second requires only the ability to count.

```
R> plot(new_roc, print_auc = TRUE, partial_auc =  c(0.8, 1.0), type = "S",
+      partial_auc_focus = "sensitivity", partial_auc_correct = TRUE)
```

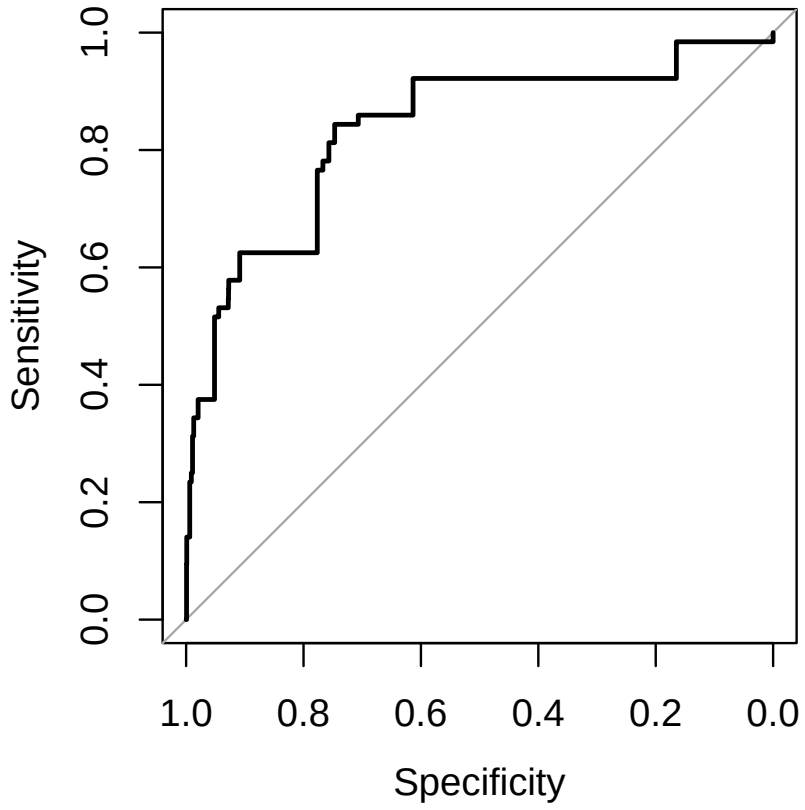


Fig 3. ROC curve from a sample of entirely new subjects, produced from the new_roc object

Implementation based on probabilities of testing positive

First, a smartphone or tablet application could be developed which contains the questionnaire and code that mimics the functionality of the 'predict' method. The predicted probability of infection is given by

$$P = \frac{1}{1 + exp(-\mathbf{Xb})}$$

where $\mathbf{X}$ is a (design) matrix containing values of 1 in the first column and the 0/1 question responses in the remaining columns, and $\mathbf{b}$ is a column vector of the estimated logit-scale coefficients, including the intercept.

The interviewer would enter the responses to the questions and the application would indicate whether the subject should be referred for testing based on the internally computed value of P. This is the ideal approach, and should be used wherever possible.

NOTE: Screening questions for which estimated coefficients equal 0 (and those zero-valued coefficients) can and should be eliminated from the screening application.

However, an approach which does not require electronic technology may be needed in

many health-care settings.

Simplified implementation based on question weights

Alternatively, the non-zero coefficient estimates could be rounded to whole numbers between 1 and m, where m is some number from, say, 3 to 10. Those rounded values are used as weights for the question responses. The score for an individual is the sum of the question weights. The choice of m involves a trade-off. Small values, say $m = 3$, make it easy to add-up or tally the scores. However, they will also result in degraded performance in terms of sensitivity and specificity. Large values for m may approach the performance of screening based on the predicted probabilities of infection, but require more difficult additions and therefore are more prone to human error.

We can explore this strategy for UIV screening in unicorns using the easy_tool function:

```
R> et_3 <- easy_tool(uniobj1, max = 3, model = "minAIC", crossval = TRUE)
R> class(et_3)
[1] "easy_tool"
```

Recall that uniobj1 contains the results of lasso estimation. The argument max represents m, in this case 3. The remaining arguments specify creation of question weights from the AIC-best model, with performance base on the cross-validated results in uniobj1. The resulting object, et_3, is of class easy_tool.

Methods for that class are found by executing:

```
R> methods(class = "easy_tool")
[1] get_what ntpp     plot     predict  print    summary
see '?methods' for accessing help and source code
```

The question weights can be extracted using the get_what method:

```
R> qwts <- get_what(from = et_3, what = "QuestionWeights")
R> print(qwts)
   [,1]
Q1    3
Q2    2
Q3    3
Q4    0
Q5    2
Q6    1
Q7    1
```

The decision to use those question weights for screening should be based on their receiver-operating characteristic. Again, a plot method is available to display the ROC curve: Note that the AUC (Fig. 4) is almost as large as the AUC from the actual model fit. However there are fewer local maximas, and therefore fewer choices for screening thresholds. Again, the screening thresholds for the scores based on the sums of the question weights can be printed:

```
R> qw_maximas <- get_what(from = et_3, what = "ROCci")
R> print(qw_maximas)
  Threshold Sensitivity   se.lcl   se.ucl Specificity   sp.lcl   sp.ucl
1         1    1.000000 1.000000 1.000000    0.140352 0.131967 0.149246
2         3    0.979381 0.948454 1.000000    0.553956 0.540738 0.566322
3         4    0.948454 0.896907 0.989691    0.706251 0.694731 0.717771
```

```
R> plot(et_3)
================================================================================
```

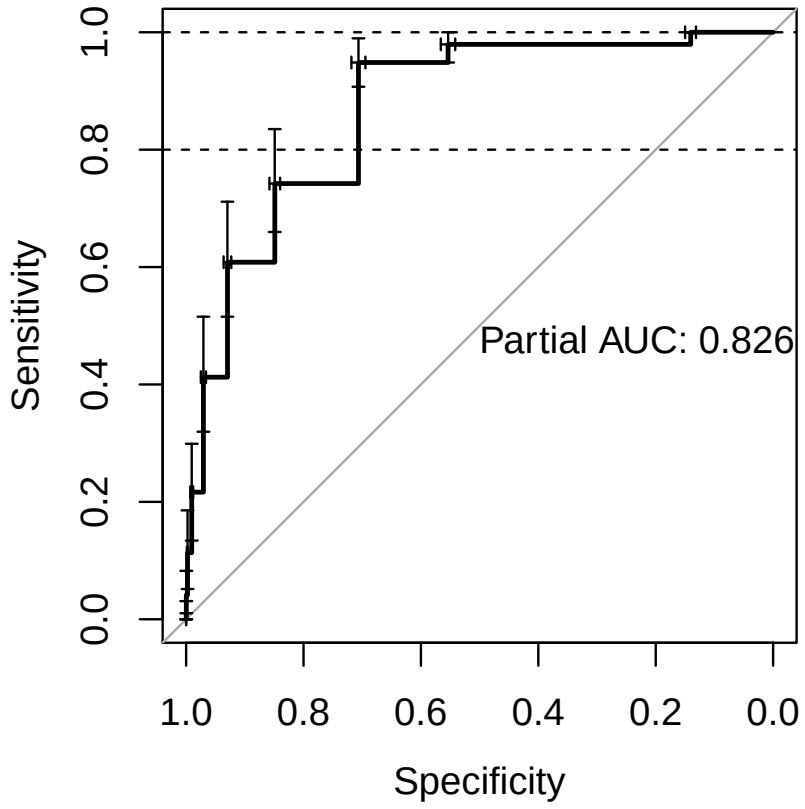


Fig 4. ROC curve produced from application of easy_tool function.

Referral for testing all unicorns having a screening score (threshold) of 3 or more will have point estimates for sensitivity and specificity of 0.98 and 0.55, respectively.

An example of a simple clinical screening form based on question weights is shown in Fig. 5.

NOTE: Screening questions for which the question weights equal 0 can and should be eliminated from the screening form.

Effects of screening on testing results

How will implementation of that screening tool affect testing? To help answer that question, we can compute the ratio of total number of tests performed to the anticipated number of positive test results and the anticipated prevalence among those who are screened *out* of testing. Effective screening will greatly reduce the prevalence among those who are screened out of testing and will have high sensitivity. Efficient screening will reduce the ratio of numbers of tests to positive test results. Both measures are obtained by using the ntpp (number of tests per positive) function:

```
R> ntpp(et_3)
   sensitivity specificity      prev     ntpp prev_untested
1    1.0000000    0.000000 0.0161667 61.85567           NaN
2    1.0000000    0.140437 0.0161667 53.30928   0.000000000
3    0.9793814    0.372692 0.0161667 39.97895   0.000908265
4    0.9793814    0.553786 0.0161667 28.72632   0.000611434
5    0.9484536    0.706420 0.0161667 19.83696   0.001197605
6    0.7422680    0.848890 0.0161667 13.38889   0.004964257
7    0.6082474    0.929527 0.0161667  8.05085   0.006877828
8    0.4123711    0.970523 0.0161667  5.35000   0.009851365
9    0.2164948    0.990513 0.0161667  3.66667   0.012831335
10   0.1134021    0.997628 0.0161667  2.27273   0.014393305
11   0.0412371    0.999661 0.0161667  1.50000   0.015515516
12   0.0103093    0.999831 0.0161667  2.00000   0.016005335
13   0.0000000    1.000000 0.0161667      NaN   0.016166667
```

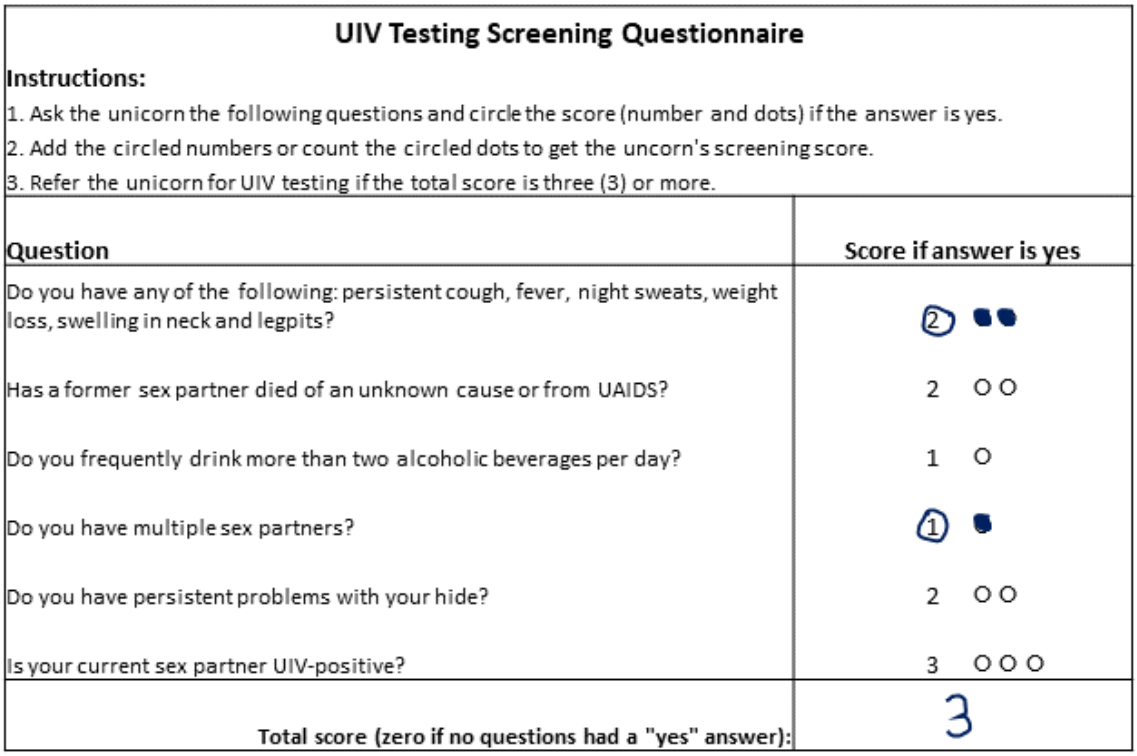
After screening out unicorns having score less than 3 (row 4, as identified by sensitivity and specificity), the prevalence of UIV among the *untested* should be approximately 0.0006 (0.06%). In contrast, the prevalence of UIV among all subjects in the training data is 0.0162.

The ratio of total tests to positive results is 28.7 when testing unicorns with a score of at least 3. Without screening, in contrast, an average of approximately 61.7 would need to be tested in order to detect a single infection.

Epilogue

The screenr package eases the workload for development and evaluation of screening tools. Hopefully screenr will prove useful to both novice and experienced R users. This tutorial documents only the basic steps. However much more is possible. Execute `help(screenr)` in an R session to obtain a broader list of capabilities. End-users are encouraged to further explore the methods that are available for objects. Make use of the R commands `class(some_object)` and `methods(class = "some_class")` where `some_object` is the result of some function call, and `"some_class"` is the class of that object. Then use `help(methodname.some_class)` to obtain help for use of method `methodname`.

Some may want results that are not directly available from screenr functions and methods. In those cases, the get_what methods may provide an easy way to extract the results contained in the objects produced by, for example, the glmpath::glmpath and pROC::roc functions. However, experienced R users may also use `str(some_object)` to identify other components for extraction using the idiom `some_object$some_component`.



UIV Testing Screening Questionnaire

Instructions:

1. Ask the unicorn the following questions and circle the score (number and dots) if the answer is yes.
2. Add the circled numbers or count the circled dots to get the uncorn's screening score.
3. Refer the unicorn for UIV testing if the total score is three (3) or more.

Question	Score if answer is yes
Do you have any of the following: persistent cough, fever, night sweats, weight loss, swelling in neck and legpits?	(2) ●●
Has a former sex partner died of an unknown cause or from UAIDS?	2 O O
Do you frequently drink more than two alcoholic beverages per day?	1 O
Do you have multiple sex partners?	(1) ●
Do you have persistent problems with your hide?	2 O O
Is your current sex partner UIV-positive?	3 O O O
Total score (zero if no questions had a "yes" answer):	3

Fig 5. Example screening questionnaire based on simplification of the AIC-best logistic classification model.

References

[1] Clemens SL, Macneal KD, Alons CL, Cohn JE. Screening algorithms to reduce burden of pediatric HIV testing: A systematic review and meta-analysis. The Pediatric Infectious Disease Journal. 2020;39(10):e303–e309.

[2] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. New York: Springer; 2009.

[3] Park MY, Hastie T. L1-regularization path algorithm for generalized linear models. Journal of the Royal Statistical Society, Series B. 2007;69(4):659–677.

[4] Fawcett T. An introduction to ROC analysis. Pattern Recognition Letters. 2006;27(8):861–874.

[5] Streiner DL, Cairney J. What's under the ROC? An introduction to receiver operating characteristics curves. The Canadian Journal of Psychiatry. 2007;52(2):121–128.

[6] Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, et al. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics. 2011;12(77):1–8.