Fit prob

Alexander V. Alekseyenko

original: 6/1/2020; revised: 8/7/2020; revised: 9/22/2020

```r
covid = read.csv("../data/covid_clean_2020_06_08.csv")
covid$Age = as.numeric(as.character(covid$Age))
covid$Age.cat = cut(covid$Age, breaks = c(0, 40, 70, 200))
covid$positive = (covid$RESULT_TEXT == "Positive")
train = covid$Day_order<521
covid$train = covid$Day_order<521
covid$hospital = covid$FACILITY== "MUSC HOSPITAL"
table(train, covid$positive)
```

```
##
## train   FALSE  TRUE
##   FALSE  6812   325
##   TRUE  24502  1212
```

```r
colnames(covid)
```

```
##  [1] "DEID_MRN"          "Follow_up"         "Previous_negative"
##  [4] "Previous_positive" "Age"               "FACILITY"
##  [7] "Month"             "Day"               "Day_order"
## [10] "RESULT_TEXT"       "HOSPITALIZED"      "Training"
## [13] "Age.cat"           "positive"          "train"
## [16] "hospital"
```

Table 1

```r
table(covid$train)
```

```
##
## FALSE  TRUE
##  7137 25714
```

```r
prop = function(x) sum(x)/length(x)
table(covid$positive)
```

```
##
## FALSE  TRUE
## 31314  1537
```

```r
with(covid, tapply(positive, train, prop))
```

```
##      FALSE       TRUE
## 0.04553734 0.04713386
```

```r
prop(covid$positive)
```

```
## [1] 0.04678701
```

```r
chisq.test(with(covid, table(positive, train)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  with(covid, table(positive, train))
## X-squared = 0.28448, df = 1, p-value = 0.5938
```

```r
prop(covid$Follow_up)
```

```
## [1] 0.06066786
```

```r
with(covid, tapply(Follow_up, train, prop))
```

```
##      FALSE       TRUE
## 0.09583859 0.05090612
```

```r
chisq.test(with(covid, table(Follow_up, train)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  with(covid, table(Follow_up, train))
## X-squared = 197.13, df = 1, p-value < 2.2e-16
```

```r
prop(covid$Previous_positive)
```

```
## [1] 0.008797297
```

```r
with(covid, tapply(Previous_positive, train, prop))
```

```
##       FALSE        TRUE
## 0.009948157 0.008477872
```

```r
chisq.test(with(covid, table(Previous_positive, train)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  with(covid, table(Previous_positive, train))
## X-squared = 1.2215, df = 1, p-value = 0.2691
```

```r
table(covid$Age.cat)/length(covid$Age.cat)
```

```
##
##    (0,40]   (40,70]  (70,200]
## 0.2818788 0.5268637 0.1912575
```

```r
with(covid, table(Age.cat, train))
```

```
##           train
## Age.cat    FALSE  TRUE
##   (0,40]    1767  7493
##   (40,70]   3878 13430
##   (70,200]  1492  4791
```

```r
table(covid$train)
```

```
##
## FALSE  TRUE
##  7137 25714
```

```r
chisq.test(with(covid, table(Age.cat, train)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(covid, table(Age.cat, train))
## X-squared = 57.854, df = 2, p-value = 2.737e-13
```

```r
with(subset(covid, Previous_positive==1),
     table(Age.cat))
```

```
## Age.cat
##   (0,40]  (40,70] (70,200]
##       78      123       88
```

```r
with(subset(covid, Previous_positive==1),
     table(Age.cat, train))
```

```
##           train
## Age.cat    FALSE TRUE
##   (0,40]      27   51
##   (40,70]     18  105
##   (70,200]    26   62
```

```r
prop(covid$HOSPITALIZED=="Y")
```

```
## [1] 0.1256887
```

```r
with(covid, tapply(HOSPITALIZED=="Y", train, prop))
```

```
##     FALSE      TRUE
## 0.1192378 0.1274792
```

```r
chisq.test(with(covid, table(HOSPITALIZED, train)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  with(covid, table(HOSPITALIZED, train))
## X-squared = 3.3783, df = 1, p-value = 0.06606
```

```r
prop(covid$hospital)
```

```
## [1] 0.1426441
```

```r
with(covid, tapply(hospital, train, prop))
```

```
##     FALSE      TRUE
## 0.1261034 0.1472350
```

```r
chisq.test(with(covid, table(hospital, train)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
```

```
##
## data:  with(covid, table(hospital, train))
## X-squared = 20.225, df = 1, p-value = 6.883e-06
```

```
(full.m = glm(positive~Follow_up+Previous_positive*Age.cat+HOSPITALIZED + hospital, data=covid, subset =
```

```
##
## Call:  glm(formula = positive ~ Follow_up + Previous_positive * Age.cat +
##     HOSPITALIZED + hospital, family = "binomial", data = covid,
##     subset = train)
##
## Coefficients:
##                      (Intercept)                            Follow_up
##                         -2.93297                             -0.50675
##                Previous_positive                       Age.cat(40,70]
##                          4.00880                             -0.35756
##                   Age.cat(70,200]                        HOSPITALIZEDY
##                         -0.31034                              0.24921
##                     hospitalTRUE   Previous_positive:Age.cat(40,70]
##                          0.01947                              0.83600
## Previous_positive:Age.cat(70,200]
##                          0.91274
##
## Degrees of Freedom: 25713 Total (i.e. Null);  25705 Residual
## Null Deviance:       9771
## Residual Deviance: 8961  AIC: 8979
```

```
summary(full.m)
```

```
##
## Call:
## glm(formula = positive ~ Follow_up + Previous_positive * Age.cat +
##     HOSPITALIZED + hospital, family = "binomial", data = covid,
##     subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8181  -0.3221  -0.2730  -0.2704   2.7639
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -2.93297    0.05725 -51.233  < 2e-16 ***
## Follow_up                          -0.50675    0.19166  -2.644 0.008191 **
## Previous_positive                   4.00880    0.35202  11.388  < 2e-16 ***
## Age.cat(40,70]                     -0.35756    0.07102  -5.035 4.79e-07 ***
## Age.cat(70,200]                    -0.31034    0.09409  -3.298 0.000972 ***
## HOSPITALIZEDY                       0.24921    0.09032   2.759 0.005795 **
## hospitalTRUE                        0.01947    0.08605   0.226 0.821003
## Previous_positive:Age.cat(40,70]    0.83600    0.38004   2.200 0.027822 *
## Previous_positive:Age.cat(70,200]   0.91274    0.43951   2.077 0.037827 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 9770.7  on 25713  degrees of freedom
## Residual deviance: 8960.8  on 25705  degrees of freedom
## AIC: 8978.8
##
## Number of Fisher Scoring iterations: 6
```

```r
be.m = step(full.m)
```

```
## Start:  AIC=8978.76
## positive ~ Follow_up + Previous_positive * Age.cat + HOSPITALIZED +
##     hospital
##
##                             Df Deviance    AIC
## - hospital                   1   8960.8 8976.8
## <none>                           8960.8 8978.8
## - Previous_positive:Age.cat  2   8966.5 8980.5
## - HOSPITALIZED               1   8968.0 8984.0
## - Follow_up                  1   8968.8 8984.8
##
## Step:  AIC=8976.81
## positive ~ Follow_up + Previous_positive + Age.cat + HOSPITALIZED +
##     Previous_positive:Age.cat
##
##                             Df Deviance    AIC
## <none>                           8960.8 8976.8
## - Previous_positive:Age.cat  2   8966.5 8978.5
## - HOSPITALIZED               1   8968.2 8982.2
## - Follow_up                  1   8968.8 8982.8
```

```r
summary(be.m)
```

```
##
## Call:
## glm(formula = positive ~ Follow_up + Previous_positive + Age.cat +
##     HOSPITALIZED + Previous_positive:Age.cat, family = "binomial",
##     data = covid, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8110  -0.3228  -0.2707  -0.2707   2.7627
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          -2.92832    0.05338 -54.858  < 2e-16 ***
## Follow_up                            -0.50565    0.19160  -2.639 0.008312 **
## Previous_positive                     4.01165    0.35181  11.403  < 2e-16 ***
## Age.cat(40,70]                       -0.36001    0.07017  -5.130 2.89e-07 ***
## Age.cat(70,200]                      -0.31417    0.09253  -3.395 0.000685 ***
## HOSPITALIZEDY                         0.25063    0.09011   2.781 0.005411 **
## Previous_positive:Age.cat(40,70]      0.83262    0.37975   2.193 0.028339 *
## Previous_positive:Age.cat(70,200]     0.90996    0.43934   2.071 0.038340 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 9770.7  on 25713  degrees of freedom
## Residual deviance: 8960.8  on 25706  degrees of freedom
## AIC: 8976.8
##
## Number of Fisher Scoring iterations: 6
```

Save predictions

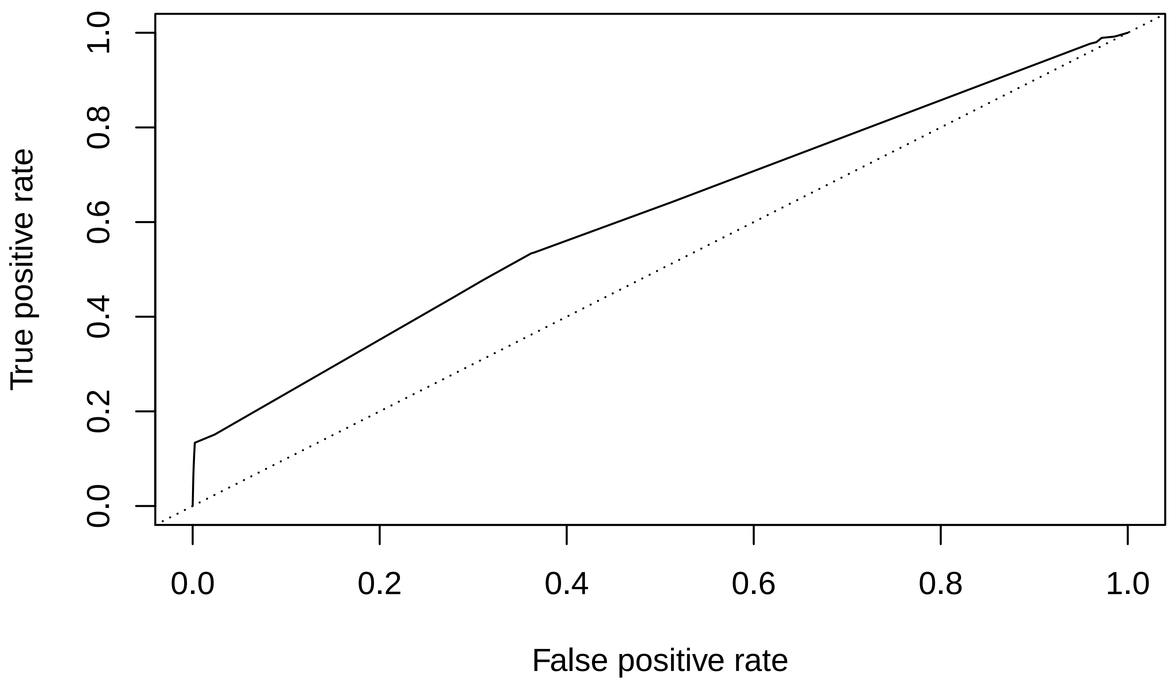
```
covid$be.prob = predict(be.m, newdata = covid, type="response")
```

Compute model AUCs

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.6.2
```

```
# training
pred.training = prediction(predictions = covid$be.prob[train], labels = covid$positive[train])
perf.training = performance(pred.training, "tpr","fpr")
plot(perf.training)
abline(0,1, lty="dotted")
```




```
performance(pred.training, "auc")@y.values
```

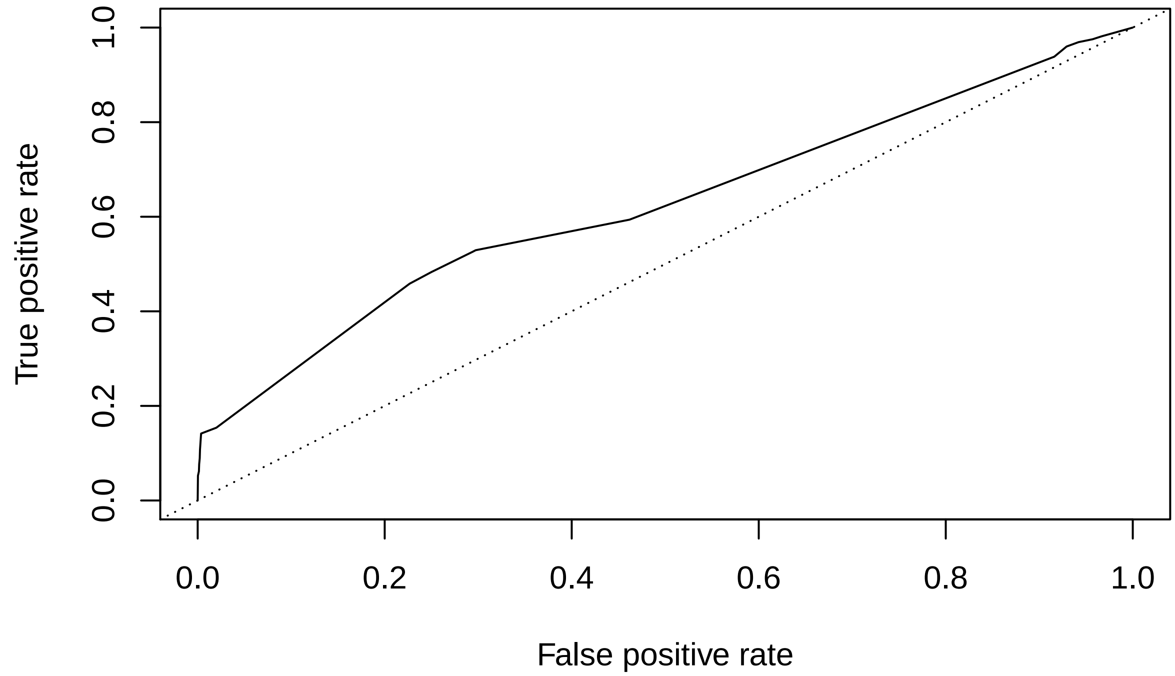
```
## [[1]]
## [1] 0.609755
```

```
1-wilcox.test(covid$be.prob[train]~covid$positive[train])$statistic/prod(table(covid$positive[train]))
```

```
##        W
## 0.609755
```

```
# testing
pred.testing = prediction(predictions = covid$be.prob[!train], labels = covid$positive[!train])
perf.testing = performance(pred.testing, "tpr","fpr")
```

```r
plot(perf.testing)
abline(0,1, lty="dotted")
```




```r
performance(pred.testing, "auc")@y.values
```

```
## [[1]]
## [1] 0.6232154
```

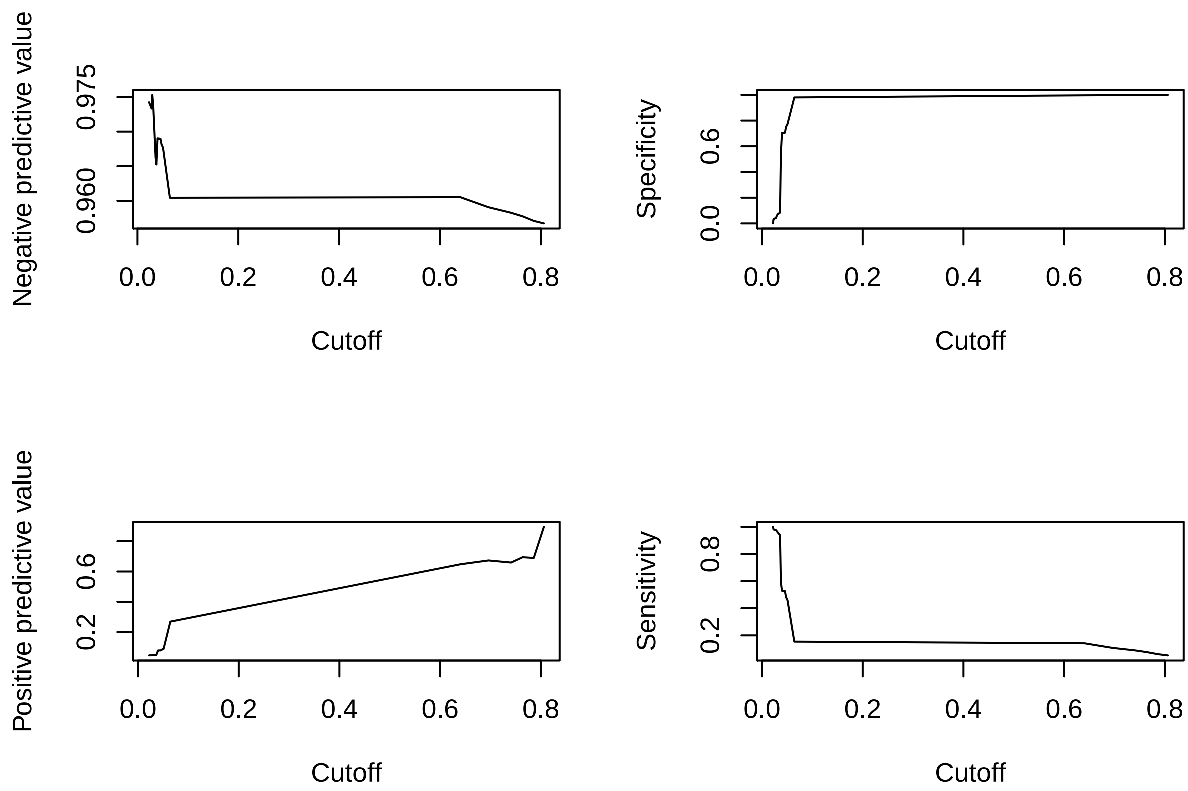
```r
1-wilcox.test(covid$be.prob[!train]~covid$positive[!train])$statistic/prod(table(covid$positive[!train])
```

```
##         W
## 0.6232154
```

Compute other model performance characteristics.

```r
par(mfrow=c(2,2))
plot(performance(pred.testing, "npv"))
plot(performance(pred.testing, "spec"))

plot(performance(pred.testing, "ppv"))
plot(performance(pred.testing, "sens"))
```



Prediction with just age

```
(age.m = glm(positive~Age.cat,
             data=covid,
             subset = train,
             family = "binomial"))
```

```
##
## Call:  glm(formula = positive ~ Age.cat, family = "binomial", data = covid,
##     subset = train)
##
## Coefficients:
##     (Intercept)   Age.cat(40,70]  Age.cat(70,200]
##         -2.8390          -0.2864          -0.1346
##
## Degrees of Freedom: 25713 Total (i.e. Null);  25711 Residual
## Null Deviance:       9771
## Residual Deviance: 9752  AIC: 9758
```

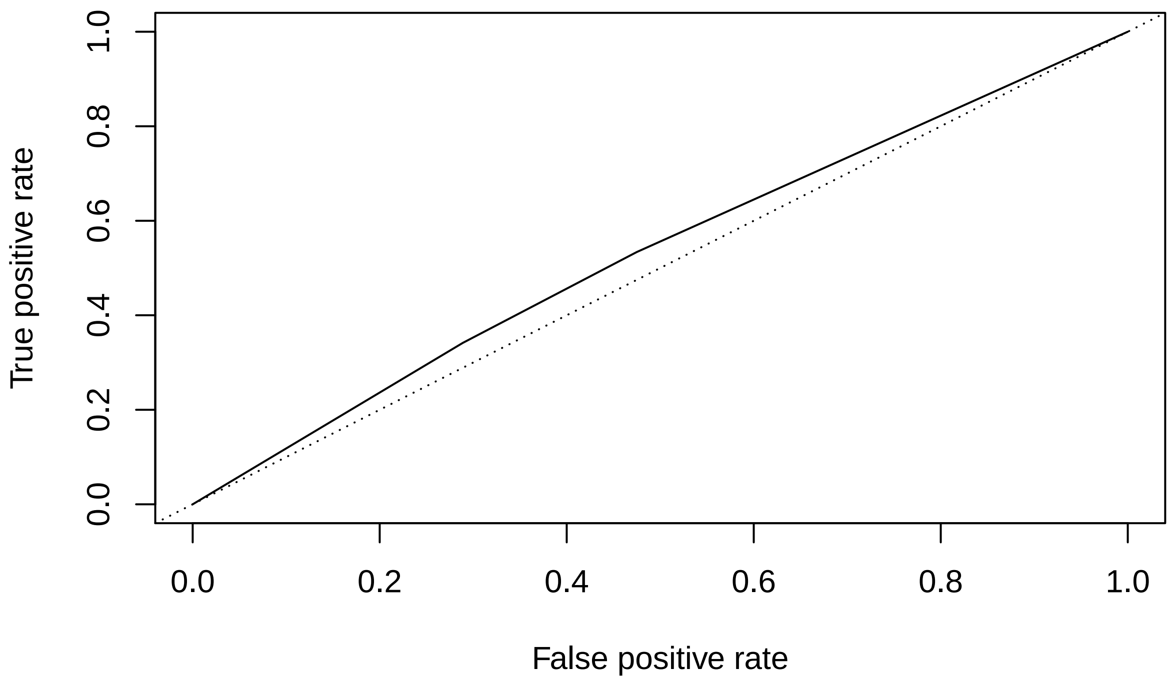
```
summary(age.m)
```

```
##
## Call:
## glm(formula = positive ~ Age.cat, family = "binomial", data = covid,
##     subset = train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -0.3372  -0.3372  -0.2932  -0.2932   2.5173
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.83902    0.05056 -56.148  < 2e-16 ***
## Age.cat(40,70]  -0.28642    0.06636  -4.316 1.59e-05 ***
## Age.cat(70,200] -0.13458    0.08407  -1.601    0.109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9770.7  on 25713  degrees of freedom
## Residual deviance: 9752.1  on 25711  degrees of freedom
## AIC: 9758.1
##
## Number of Fisher Scoring iterations: 5
```

```r
covid$age.prob = predict(age.m, newdata = covid, type="response")
```

```r
# training
age.pred.training = prediction(predictions = covid$age.prob[train], labels = covid$positive[train])
age.perf.training = performance(age.pred.training, "tpr","fpr")
plot(age.perf.training)
abline(0,1, lty="dotted")
```



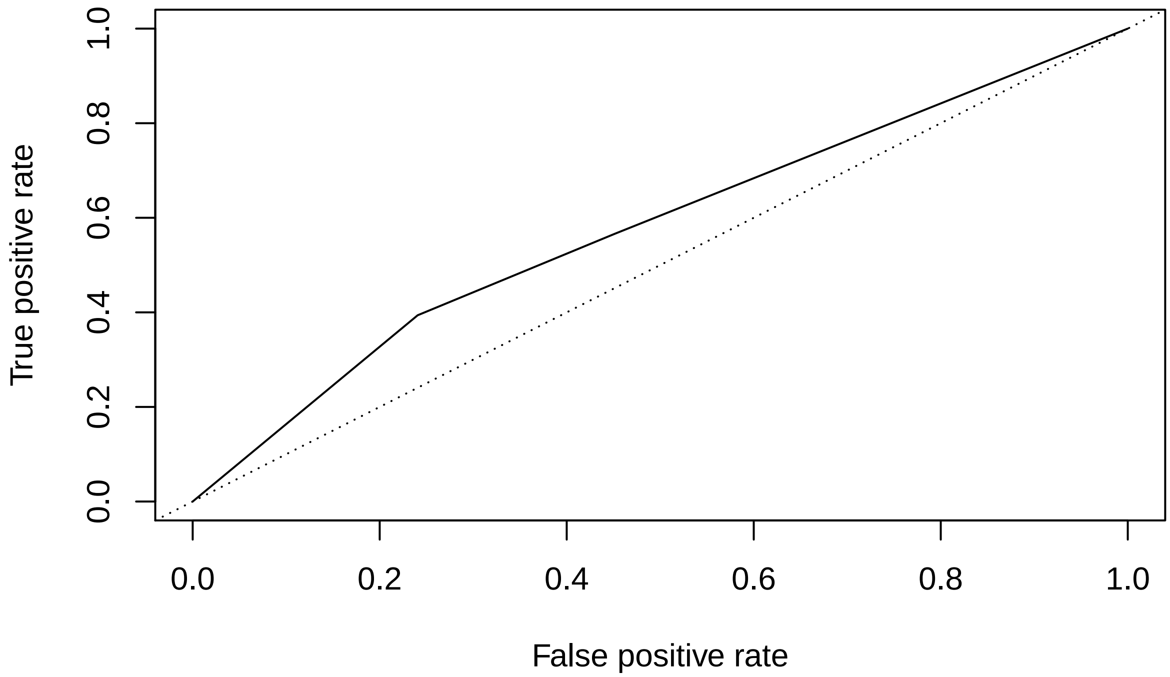

```r
performance(age.pred.training, "auc")@y.values
```

```
## [[1]]
## [1] 0.5334443
```

```
1-wilcox.test(covid$age.prob[train]~covid$positive[train])$statistic/prod(table(covid$positive[train]))
```

```
##         W
## 0.5334443
```

```
# testing
age.pred.testing = prediction(predictions = covid$age.prob[!train], labels = covid$positive[!train])
age.perf.testing = performance(age.pred.testing, "tpr","fpr")
plot(age.perf.testing)
abline(0,1, lty="dotted")
```




```
performance(age.pred.testing, "auc")@y.values
```

```
## [[1]]
## [1] 0.5781555
```

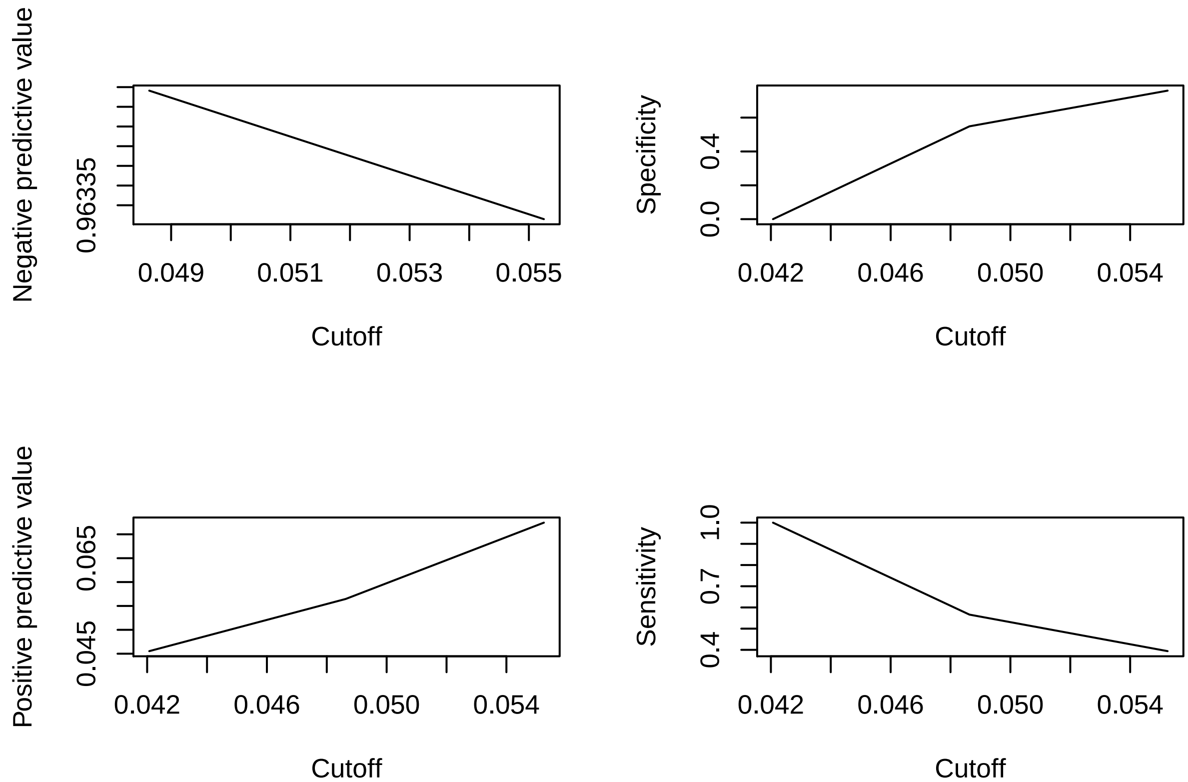
```
1-wilcox.test(covid$age.prob[!train]~covid$positive[!train])$statistic/prod(table(covid$positive[!train
```

```
##         W
## 0.5781555
```

Calculate other predictive model characteristics.

```
par(mfrow=c(2,2))
plot(performance(age.pred.testing, "npv"))
plot(performance(age.pred.testing, "spec"))

plot(performance(age.pred.testing, "ppv"))
plot(performance(age.pred.testing, "sens"))
```




```r
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.6.2
```

```r
predictionEval = as.data.frame(with(covid[train,], table(be.prob, positive)))
predictionEval = dcast(predictionEval, be.prob~positive)
```

```
## Using Freq as value column: use value.var to override.
```

```r
predictionEval$n = (predictionEval$"TRUE" + predictionEval$"FALSE")
predictionEval$EmpiricalRate = predictionEval$"TRUE"/predictionEval$n
predictionEval$FittedRate = as.numeric(as.character(predictionEval$be.prob))
predictionEval
```

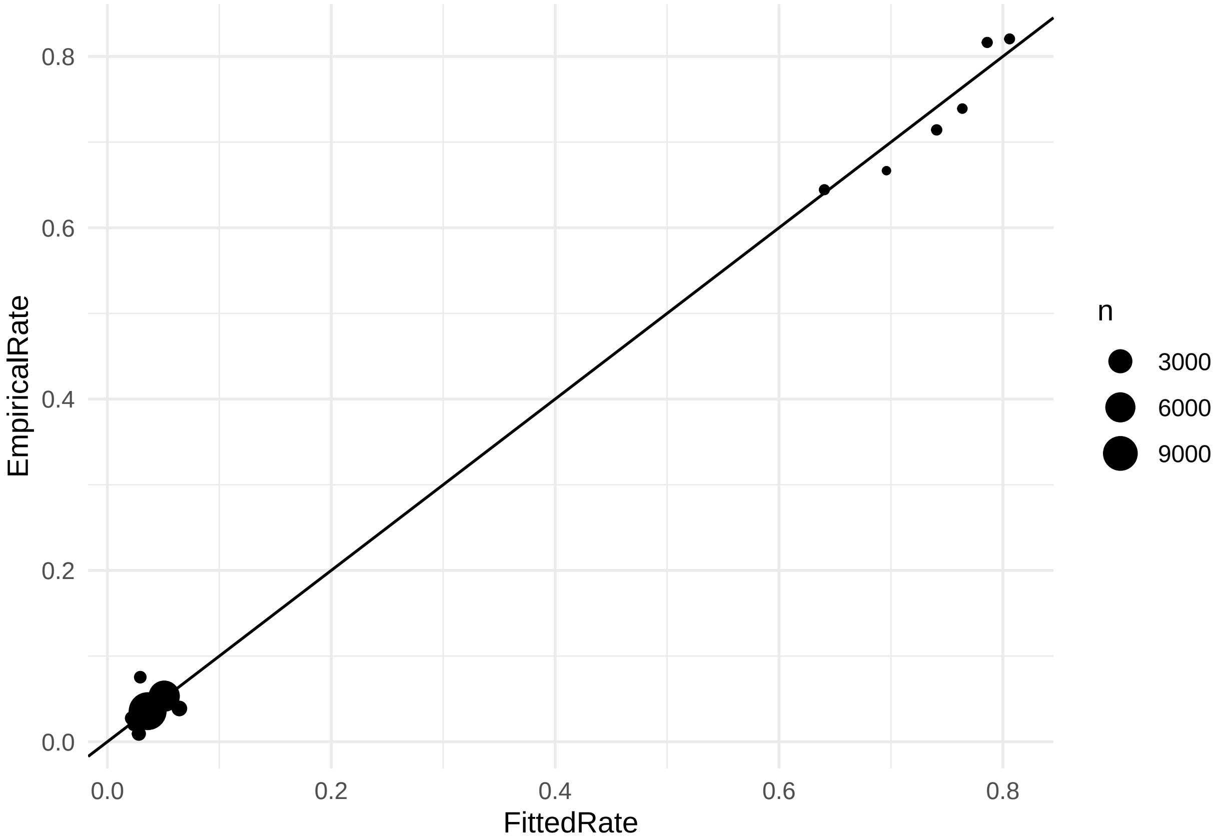
```
##               be.prob FALSE TRUE     n EmpiricalRate FittedRate
## 1  0.0220104600785734   319    9   328   0.027439024 0.02201046
## 2  0.0230191454872697    51    1    52   0.019230769 0.02301915
## 3  0.0281034795944723   316    3   319   0.009404389 0.02810348
## 4  0.0293830039335229   135   11   146   0.075342466 0.02938300
## 5  0.0312505462890353   167    4   171   0.023391813 0.03125055
## 6  0.0359737645708749 10969  406 11375   0.035692308 0.03597376
## 7  0.0375977535624169  3602  129  3731   0.034575181 0.03759775
## 8  0.0397973540036863    73    2    75   0.026666667 0.03979735
## 9  0.0457513331625897  1236   67  1303   0.051419800 0.04575133
## 10 0.0477948341490868   757   43   800   0.053750000 0.04779483
## 11 0.0507713936823829  6301  354  6655   0.053193088 0.05077139
## 12 0.0643026433981086   520   21   541   0.038817006 0.06430264
## 13  0.640533760886308    16   29    45   0.644444444 0.64053376
## 14   0.69599679335406     2    4     6   0.666666667 0.69599679
## 15  0.740830861679789    16   40    56   0.714285714 0.74083086
## 16  0.763772483864247     6   17    23   0.739130435 0.76377248
```

```
## 17   0.78598921930498     9   40    49    0.816326531 0.78598922
## 18  0.805980330028943     7   32    39    0.820512821 0.80598033
```

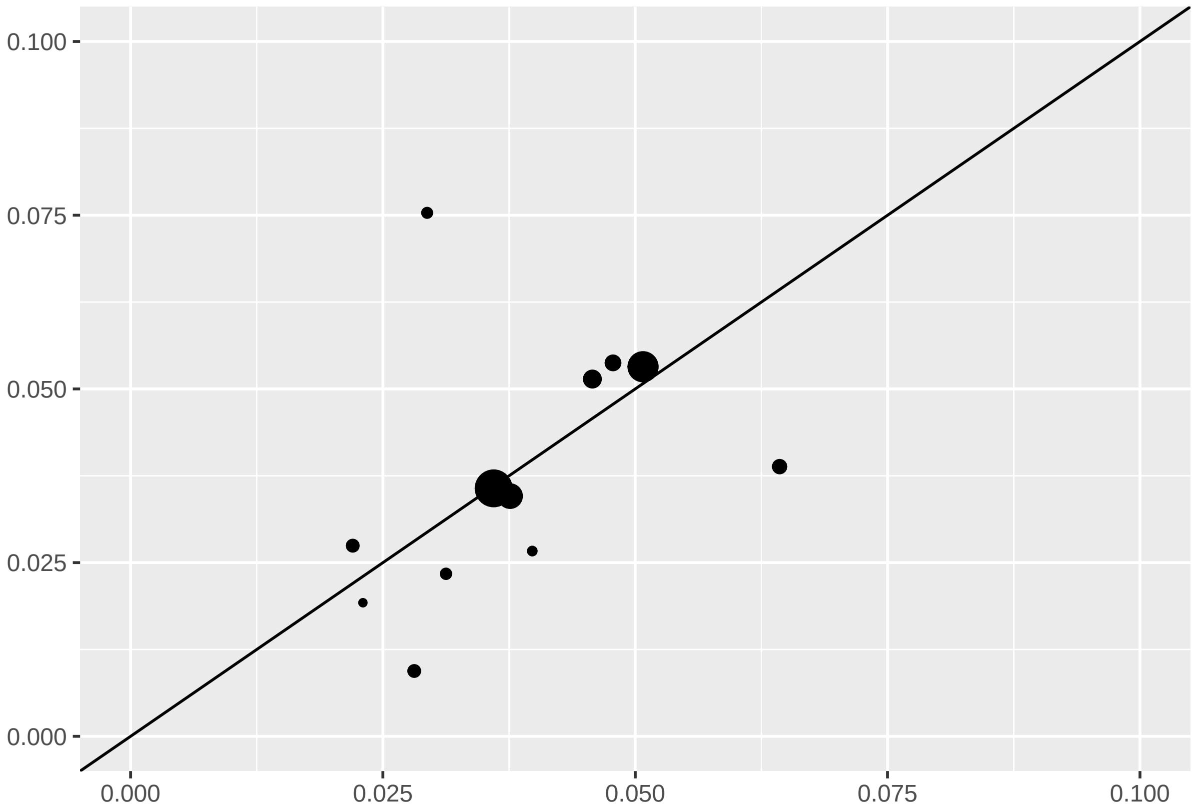
```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

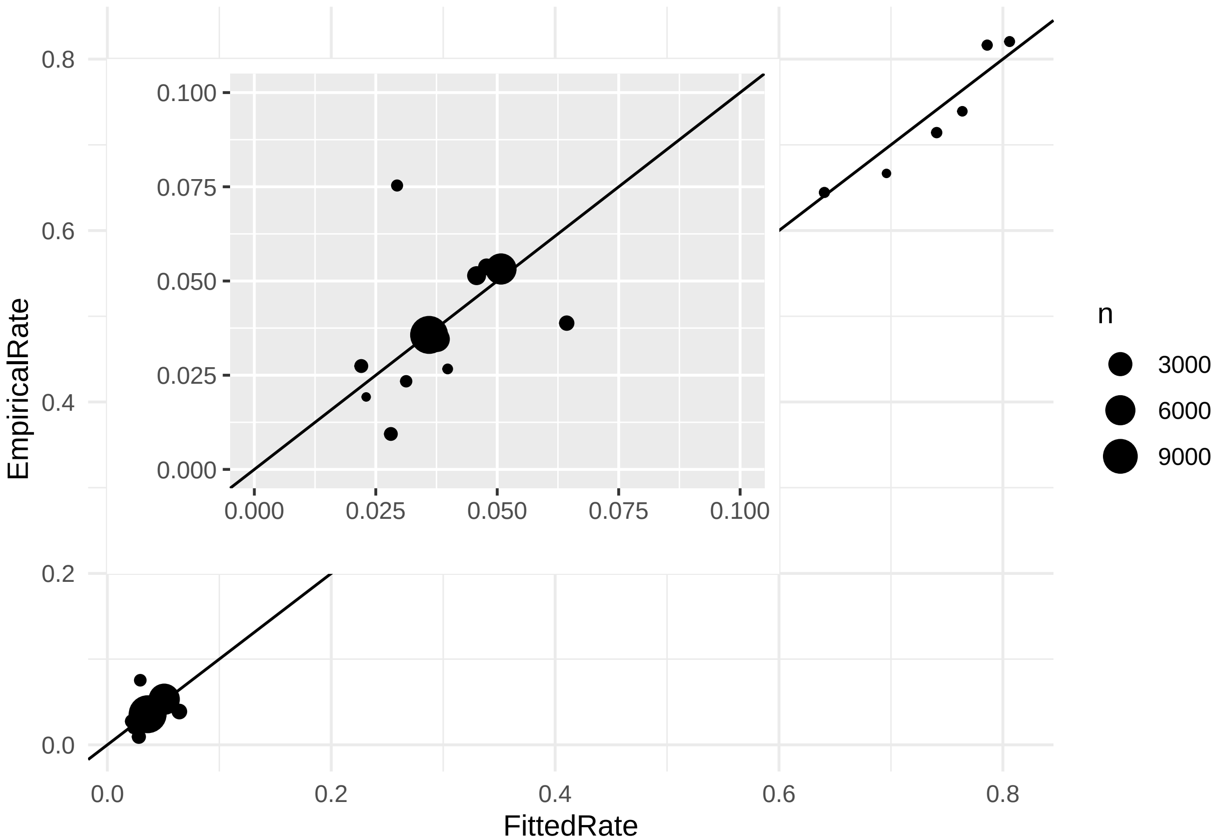
```r
mp = ggplot(data=predictionEval, aes(x=FittedRate, y=EmpiricalRate, size=n)) +
  geom_point() +
  geom_abline(slope=1, intercept = 0) +
  theme_minimal()
mp
```




```r
islow = ggplot(data=subset(predictionEval, FittedRate<0.5),
                                  aes(x=FittedRate,
                                      y=EmpiricalRate,
                                      size=n)) +
  geom_abline(slope=1, intercept = 0) + xlim(0,0.1) + ylim(0,0.1)+
  geom_point() + ylab("") + xlab("") + theme(legend.position = "none")
islow
```




```
mp + annotation_custom(grob=ggplotGrob(islow),
                       ymin = 0.2, ymax=0.8, xmin=0, xmax=.6)
```




```r
predictionEval.test = as.data.frame(with(covid[!train,],
                                         table(be.prob, positive)))
predictionEval.test = dcast(predictionEval.test, be.prob~positive)
```

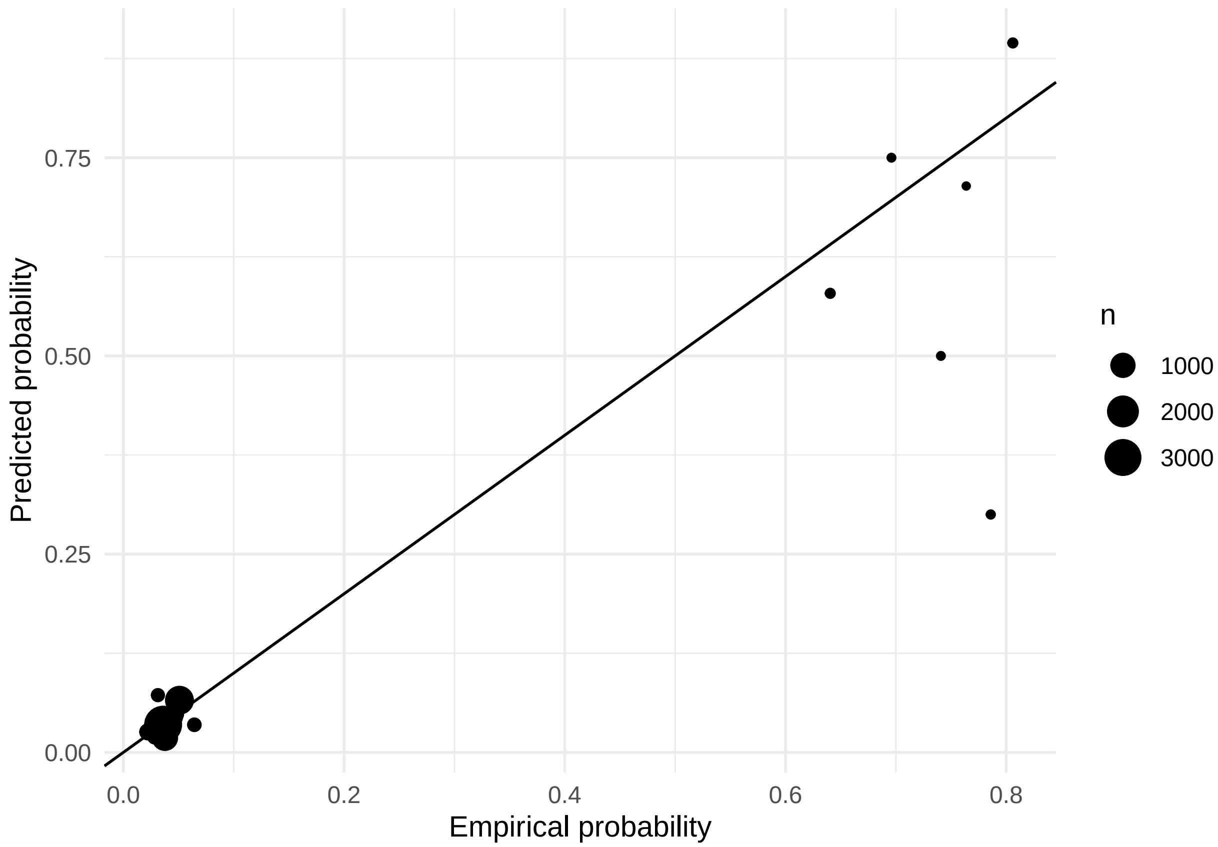
```
## Using Freq as value column: use value.var to override.
```

```r
predictionEval.test$n = (predictionEval.test$"TRUE" + predictionEval.test$"FALSE")
predictionEval.test$EmpiricalRate = predictionEval.test$"TRUE"/predictionEval.test$n
predictionEval.test$FittedRate = as.numeric(as.character(predictionEval.test$be.prob))
predictionEval.test
```

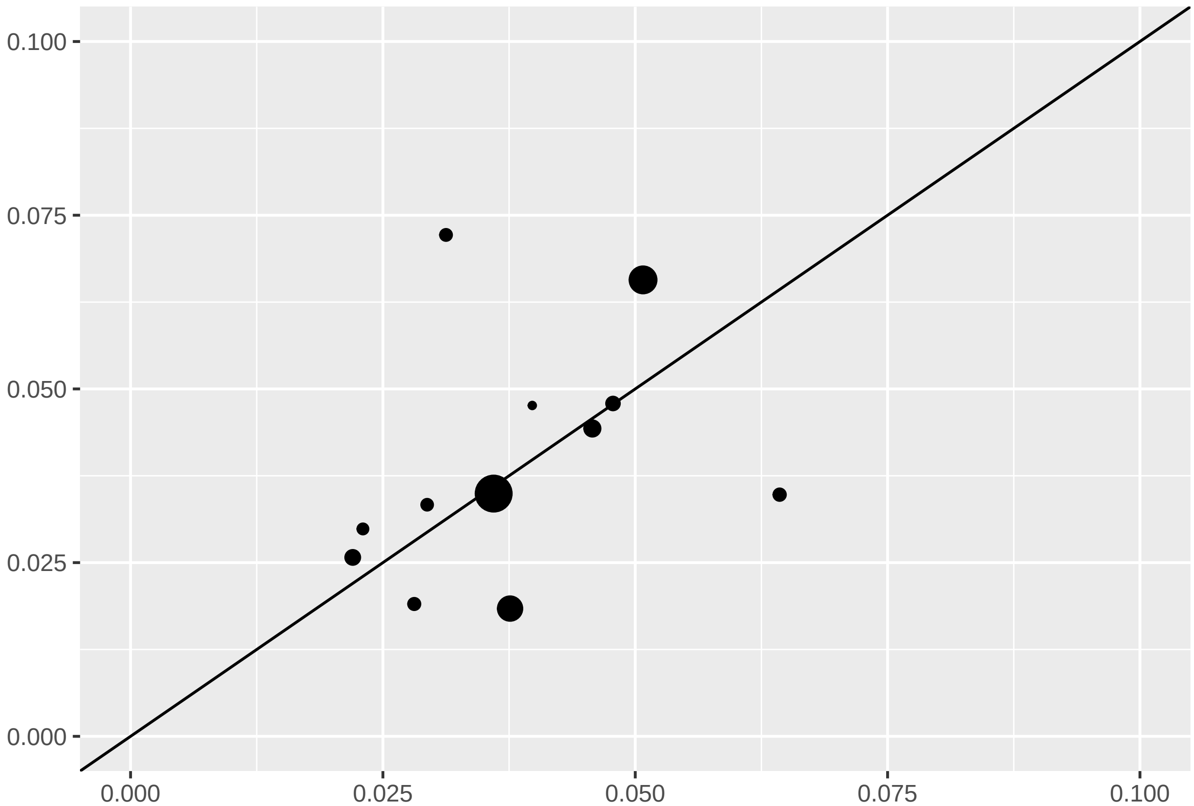
```
##                be.prob FALSE TRUE    n EmpiricalRate FittedRate
## 1  0.0220104600785734   227    6  233    0.02575107 0.02201046
## 2  0.0230191454872697    65    2   67    0.02985075 0.02301915
## 3  0.0281034795944723   103    2  105    0.01904762 0.02810348
## 4  0.0293830039335229    87    3   90    0.03333333 0.02938300
## 5  0.0312505462890353    90    7   97    0.07216495 0.03125055
## 6  0.0359737645708749  3094  112 3206    0.03493450 0.03597376
## 7  0.0375977535624169  1121   21 1142    0.01838879 0.03759775
## 8  0.0397973540036863    20    1   21    0.04761905 0.03979735
## 9  0.0457513331625897   302   14  316    0.04430380 0.04575133
## 10 0.0477948341490868   159    8  167    0.04790419 0.04779483
## 11 0.0507713936823829  1408   99 1507    0.06569343 0.05077139
## 12 0.0643026433981086   111    4  115    0.03478261 0.06430264
## 13  0.640533760886308     8   11   19    0.57894737 0.64053376
## 14   0.69599679335406     2    6    8    0.75000000 0.69599679
## 15  0.740830861679789     4    4    8    0.50000000 0.74083086
## 16  0.763772483864247     2    5    7    0.71428571 0.76377248
```

```
## 17   0.78598921930498     7     3    10     0.30000000 0.78598922
## 18  0.805980330028943     2    17    19     0.89473684 0.80598033
```

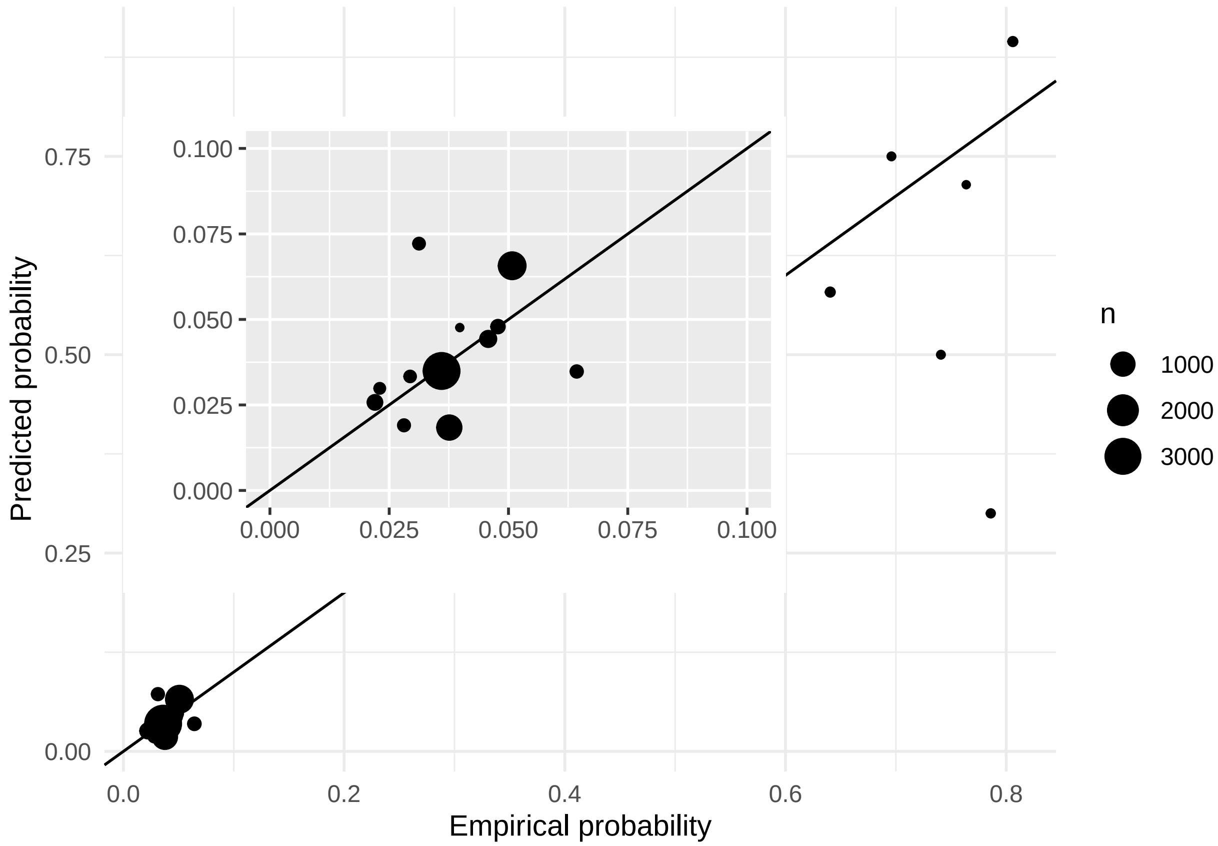
```
mp = ggplot(data=predictionEval.test, aes(x=FittedRate, y=EmpiricalRate, size=n)) +
  geom_point() + geom_abline(slope=1, intercept = 0)+ theme_minimal() +
  ylab("Predicted probability") + xlab("Empirical probability")
mp
```




```
islow = ggplot(data=subset(predictionEval.test, FittedRate<0.5),
                                  aes(x=FittedRate,
                                      y=EmpiricalRate,
                                      size=n))+
  geom_abline(slope=1, intercept = 0) + xlim(0,0.1) + ylim(0,0.1)+
  geom_point() + ylab("") + xlab("") + theme(legend.position = "none")
islow
```




```
mp + annotation_custom(grob=ggplotGrob(islow),
                       ymin = 0.2, ymax=0.8, xmin=0, xmax=.6)
```




```
topool = covid[!train & covid$Previous_positive!=1,]
dim(topool)
```

```
## [1] 7066   18
```

```
with(topool, tapply(positive, Day, function(x) sum(x)/length(x)))
```

```
##          1          2          3          4          5          6         21
## 0.06329114 0.04568528 0.04500703 0.05958549 0.06035889 0.00000000 0.02264151
##         22         23         24         25         26         27         28
## 0.03760072 0.01619433 0.13888889 0.00000000 0.02189781 0.03402187 0.03674541
##         29         30         31
## 0.05072464 0.02252252 0.00000000
```

```
with(topool, tapply(be.prob, Day, mean))
```

```
##          1          2          3          4          5          6         21
## 0.04166851 0.04105118 0.03987289 0.03989880 0.03902074 0.04126222 0.03986034
##         22         23         24         25         26         27         28
## 0.03908537 0.04049043 0.04398086 0.03974054 0.04016264 0.03955304 0.04054868
##         29         30         31
## 0.03891991 0.03992889 0.04205304
```

```
with(covid[!train,], tapply(positive, Day, function(x) sum(x)/length(x)))
```

```
##          1          2          3          4          5          6         21
## 0.06172840 0.05276382 0.05013928 0.06122449 0.06785137 0.00000000 0.02808989
##         22         23         24         25         26         27         28
## 0.04355556 0.02800000 0.18421053 0.00000000 0.03068592 0.03389831 0.05141388
```

```
##         29         30         31
## 0.05622010 0.02690583 0.05882353
```

```r
with(covid[!train,], tapply(be.prob, Day, mean))
```

```
##          1          2          3          4          5          6         21
## 0.05714004 0.04802254 0.04667174 0.05014870 0.04573127 0.04126222 0.04527111
##         22         23         24         25         26         27         28
## 0.04370810 0.04884569 0.08303393 0.06168903 0.04774181 0.04211218 0.05529942
##         29         30         31
## 0.04566510 0.04287090 0.08698994
```

```r
mean(with(covid[train,], tapply(positive, Day, function(x) sum(x)/length(x))))
```

```
## [1] 0.05143471
```

```r
mean(with(covid[train,], tapply(be.prob, Day, mean)))
```

```
## [1] 0.04803045
```

```r
sd(with(covid[train,], tapply(be.prob, Day, mean)))
```

```
## [1] 0.004680663
```

Bottom line, compare three alternative strategies: 1. Test everyone individually. 2. Test everyone in pools of (a) 5 or (b) 6 specimens. 3. Use AI Informed Adaptive Testing (AIIAT) with backward elimination model. *4. Comparison model which is more inferior to (3) only includes age

Performance characteristics: 1. Total number of tests needed. 2. Fraction of pools testing positive.

```r
#returns the number of tests needed to test all outcomes in pools of pool_size
simulate_pooling = function(pool_size, outcomes){
  if(length(outcomes)%%pool_size>0){
    outcomes = c(outcomes, rep(F, pool_size-length(outcomes)%%pool_size))
  }
  sum(apply(matrix(outcomes, ncol=pool_size, byrow=T),1,any))*pool_size + # individual re-tests
    length(outcomes)/pool_size
}

simulate_poolingn = function(n, pool_size, outcomes){
  mean(replicate(n, simulate_pooling(pool_size, sample(outcomes))))
}

simulate_positive_pools = function(pool_size, outcomes){
  if(length(outcomes)%%pool_size>0){
    outcomes = c(outcomes, rep(F, pool_size-length(outcomes)%%pool_size))
  }
  sum(apply(matrix(outcomes, ncol=pool_size, byrow=T),1,any))/(length(outcomes)/pool_size)
}

simulate_positive_poolsn = function(n, pool_size, outcomes){
  mean(replicate(n, simulate_positive_pools(pool_size, sample(outcomes))))
}

#simulate_pooling(5, covid$positive[covid$Day_order==520])
simulate_poolingn(100, 5, covid$positive[covid$Day_order==520])
```

```
## [1] 246.75
```

```r
simulate_positive_poolsn(100, 5, covid$positive[covid$Day_order==520])
```

```
## [1] 0.1666418
```

```r
res = c()
for(d in unique(covid[!train,]$Day_order)){
  day = (covid$Day_order==d)
  res = rbind(res,
              c(d,
                length(covid[day,]$DEID_MRN), # tests with strategy 1
                simulate_poolingn(1000, 5, covid[day,]$positive), # strategy 2(a)
                simulate_positive_poolsn(1000, 5, covid[day,]$positive), # strategy 2(a)
                simulate_poolingn(1000, 6, covid[day,]$positive), # strategy 2(b)
                simulate_positive_poolsn(1000, 6, covid[day,]$positive) # strategy 2(a)
          ))
}
res= data.frame(res)
colnames(res) = c("Day", "N", "UP5_ntests", "UP5_fracPos", "UP6_ntests", "UP6_fracPos")
(res.strategies = res)
```

```
##    Day    N UP5_ntests UP5_fracPos UP6_ntests UP6_fracPos
## 1  521  534    178.005   0.1329813    173.408   0.1580000
## 2  522 1125    449.460   0.1999911    451.748   0.2342766
## 3  523  250     83.340   0.1333000     81.756   0.1570952
## 4  524   38     33.370   0.6428750     35.554   0.6782857
## 5  525   68     14.000   0.0000000     12.000   0.0000000
## 6  526  554    191.295   0.1445405    187.482   0.1706559
## 7  527  826    297.225   0.1578735    292.860   0.1871159
## 8  528  389    168.850   0.2320256    171.308   0.2736615
## 9  529  836    378.575   0.2506726    386.132   0.2931929
## 10 530  223     73.705   0.1275556     72.200   0.1486053
## 11 531   17      9.000   0.2500000      9.000   0.3333333
## 12 601   81     39.470   0.2664118     40.604   0.3140714
## 13 602  398    175.055   0.2373500    178.180   0.2763582
## 14 603  718    307.195   0.2269861    311.268   0.2656583
## 15 604  392    185.285   0.2695696    189.960   0.3145303
## 16 605  619    307.785   0.2957500    317.552   0.3429615
## 17 606   69     14.000   0.0000000     12.000   0.0000000
```

AIIAT

```r
pool_positive = function(probs){
  1 - prod(1-probs)
}
cnp = function(n, pp){
  n/(1 + n*pp)
}

# Probability of pool being positive by adding a new specimen with individual probability pi
# to a pool with *pool* positive probability pp
add1pool_positive = function(pi, pp){
  pp + (1-pp)*pi
}
```

```r
# Capacity gain by additing a new specimen with individual positvie probability pi
# to a pool of n-1 specimens with *pool* positive probability pp
cnnp = function(pi, pp, n){
  n / (1 + n * add1pool_positive(pi, pp))
}

next_pool = function(probs){
  if(length(probs)<2)
    return(probs)
  pp = probs[1]
  for(i in 2:length(probs)){
    a = cnp(i-1, pp)
    b = cnnp(probs[i], pp, i)
    if((a > b) | (b<1)){
      return(probs[1:(i-1)])
    }
    pp = add1pool_positive(probs[i], pp)
  }
  return(probs)
}

form_pools = function(probs){
  res = list()
  while(length(probs)>0){
    pool = next_pool(probs)
    res = append(res, list(pool))
    probs = probs[-(1:length(pool))]
  }
  res
}

# probs = rep(.01, 14)
# pp = probs[1]
# i = 2
# cnp(i-1, pp)
# cnnp(probs[i], pp, i)
#
# form_pools(probs)
```

Get information on number of tests with AIIAT, fraction of pools testing positive, and average pool size.

```r
pools2indices = function(pools){
  z=unlist(lapply(pools, length))
  unlist(sapply(1:length(z), function(i) rep(i, z[i])))
}
# total number of tests
evaluate_pools = function(pools, positive){
  sum(tapply(positive, pools, function(x) 1+length(x)*any(x)))
}

# fraction of positive pools
evaluate_pools_positive = function(pools, positive){
  sum(tapply(positive, pools, function(x) any(x)))/length(unique(pools))
}
```

```r
# mean pool size
evaluate_pools_size = function(pools, positive){
  mean(table(pools))
}

simulate_pools = function(probs, positive, oo=1:length(probs)){
  pools = form_pools(probs[oo])
  pool_idx = pools2indices(pools)
  c(evaluate_pools(pool_idx, positive[oo]),
    evaluate_pools_positive(pool_idx, positive[oo]),
    evaluate_pools_size(pool_idx, positive[oo]))
}

evaluate_poolsn = function(n, probs, positive){
  rowMeans(replicate(n, simulate_pools(probs, positive, oo=sample(length(probs)))))
}



do = 520

simulate_pools(covid[covid$Day_order==do,]$be.prob,
                 covid[covid$Day_order==do,]$positive)
```

```
## [1] 228.000   0.184   5.344
```

```r
evaluate_poolsn(100,
                 covid[covid$Day_order==do,]$be.prob,
                 covid[covid$Day_order==do,]$positive)
```

```
## [1] 224.3200000   0.1858115   5.3616885
```

```r
aiiat.pools = c()
for(do in res.strategies$Day){
  pools = form_pools(covid[covid$Day_order==do,]$be.prob)
  ntests = evaluate_pools(pools2indices(pools), covid[covid$Day_order==do,]$positive)
  aiiat.pools = rbind(aiiat.pools,
                  evaluate_poolsn(100,
                  covid[covid$Day_order==do,]$be.prob,
                  covid[covid$Day_order==do,]$positive))
}

res.strategies$aiiat_ntests = aiiat.pools[,1]
res.strategies$aiiat_fracPos = aiiat.pools[,2]
res.strategies$aiiat_poolSize = aiiat.pools[,3]

res.strategies$positive = table(covid[!train,]$Day_order, covid[!train,]$positive)[,2]
res.strategies$UP5.c = res.strategies$N/res.strategies$UP5_ntests
res.strategies$UP6.c = res.strategies$N/res.strategies$UP6_ntests
res.strategies$aiiat.c = res.strategies$N/res.strategies$aiiat_ntests
```

age only model

```r
age.aiiat.pools = c()
for(do in res.strategies$Day){
```

```r
  pools = form_pools(covid[covid$Day_order==do,]$age.prob)
  ntests = evaluate_pools(pools2indices(pools), covid[covid$Day_order==do,]$positive)
  age.aiiat.pools = rbind(age.aiiat.pools,
                  evaluate_poolsn(100,
                  covid[covid$Day_order==do,]$age.prob,
                  covid[covid$Day_order==do,]$positive))
}

res.strategies$age.aiiat_ntests = age.aiiat.pools[,1]
res.strategies$age.aiiat_fracPos = age.aiiat.pools[,2]
res.strategies$age.aiiat_poolSize = age.aiiat.pools[,3]

res.strategies$age.aiiat.c = res.strategies$N/res.strategies$age.aiiat_ntests
```

```
## Warning: package 'knitr' was built under R version 3.6.2
```

Day	N	UP5_ntests	UP5_fracPos	UP6_ntests	UP6_fracPos	aiiat_ntests	aiiat_fracPos	aiiat_poolSize		UP5.c	UP6.c	aiiat.c	age.aiiat_ntests	age.aiiat_fracPos	age.aiiat_poolSize	age.aiiat.c
521	534	178.00	0.132981	173.40	0.158001	166.76	0.145665	8.413098	15	2.99993	3.07943	3.202207	176.75	0.143272	2.377007	3.021216
522	1125	449.46	0.199994	451.74	0.234274	432.42	0.221225	5.301213	49	2.50300	2.490327	2.601645	452.59	0.212728	1.370896	2.485694
523	250	83.340	0.133308	81.756	0.157095	73.49	0.142764	8.143392	7	2.99976	3.05783	3.401882	82.86	0.142528	7.341601	3.017137
524	38	33.370	0.642873	35.554	0.678283	37.94	0.581787	7.774283	7	1.13874	1.06879	1.002281	34.57	0.695535	7.102857	1.099219
525	68	14.000	0.000000	12.000	0.000000	14.96	0.000000	4.054568	0	4.857143	5.666667	4.545454	13.19	0.000000	0.159780	5.155421
526	554	191.29	0.144541	187.48	0.170655	175.37	0.155849	9.245358	17	2.89602	2.95493	3.159033	188.95	0.154488	0.389556	2.931993
527	826	297.22	0.157872	292.86	0.187112	297.99	0.176045	5.590536	28	2.77902	2.82046	2.771929	295.01	0.170077	1.386648	2.799905
528	389	168.85	0.232025	171.30	0.273661	156.93	0.239604	7.918339	20	2.30382	2.27077	2.478816	169.11	0.247117	7.388390	2.300278
529	836	378.57	0.250673	386.13	0.293193	374.63	0.273635	0.405490	47	2.20828	2.16506	2.231537	379.97	0.266119	5.373358	2.200174
530	223	73.705	0.127557	72.200	0.148606	53.02	0.141795	8.442638	6	3.02557	3.08864	4.230942	72.42	0.136210	4.366814	3.079260
531	17	9.000	0.250000	9.000	0.333333	5.84	0.208666	6.747333	1	1.888889	1.888889	2.910959	8.86	0.260000	4.420000	1.918736
601	81	39.470	0.266414	40.604	0.314074	41.26	0.256745	5.663404	5	2.05219	1.99487	1.963160	39.41	0.293023	8.221607	2.055316
602	398	175.05	0.237351	178.18	0.276358	168.91	0.251315	9.823282	21	2.27357	2.23369	2.356287	176.03	0.253391	5.370341	2.260978
603	718	307.19	0.226983	311.26	0.265653	304.59	0.246005	8.843114	36	2.33727	2.30669	2.357236	308.43	0.240166	9.377008	2.327919
604	392	185.28	0.269569	189.96	0.314531	190.11	0.275475	0.117833	24	2.11568	2.06359	2.061967	187.91	0.290671	0.400062	2.086105
605	619	307.78	0.295753	317.55	0.342963	306.49	0.325515	0.385027	42	2.01111	1.949287	2.019641	311.32	0.314421	0.397517	1.988308
606	69	14.000	0.000000	12.000	0.000000	12.75	0.000000	0.419533	0	4.928571	5.750000	5.411765	13.05	0.000000	0.292528	5.287356

Test if AIIAT with backward elimination LR model is better than uniform pools

```r
wilcox.test(res.strategies$aiiat_ntests, res.strategies$UP5_ntests, paired = T, alternative = "less")
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$aiiat_ntests and res.strategies$UP5_ntests
## V = 19, p-value = 0.002319
## alternative hypothesis: true location shift is less than 0
```

```r
wilcox.test(res.strategies$aiiat_ntests, res.strategies$UP6_ntests, paired = T, alternative = "less")
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$aiiat_ntests and res.strategies$UP6_ntests
## V = 18, p-value = 0.001923
## alternative hypothesis: true location shift is less than 0
```

Test if AIIAT with age only LR model is better than uniform pools

```r
wilcox.test(res.strategies$age.aiiat_ntests, res.strategies$UP5_ntests, paired = T, alternative = "less
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$age.aiiat_ntests and res.strategies$UP5_ntests
## V = 87, p-value = 0.6944
## alternative hypothesis: true location shift is less than 0
```

```r
wilcox.test(res.strategies$age.aiiat_ntests, res.strategies$UP6_ntests, paired = T, alternative = "less
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$age.aiiat_ntests and res.strategies$UP6_ntests
## V = 58, p-value = 0.2019
## alternative hypothesis: true location shift is less than 0
```

Not really! Can we say anything about the opposite?

```r
wilcox.test(res.strategies$UP5_ntests, res.strategies$age.aiiat_ntests, paired = T, alternative = "less
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$UP5_ntests and res.strategies$age.aiiat_ntests
## V = 66, p-value = 0.3221
## alternative hypothesis: true location shift is less than 0
```

```r
wilcox.test(res.strategies$UP6_ntests, res.strategies$age.aiiat_ntests, paired = T, alternative = "less
```

```
##
##  Wilcoxon signed rank test
##
## data:  res.strategies$UP6_ntests and res.strategies$age.aiiat_ntests
## V = 95, p-value = 0.8111
## alternative hypothesis: true location shift is less than 0
```

No! It looks like the Age-only model performs on par with the fixed pools.