Tuwel quiz score optimization

Emile Johnston

Contents

1. Putting the quiz logic into functions **2**
1.1. Four statements, five boxes to check . . . 2
1.2. Calculating the score of a single statement . . . 2
1.3. Calculating the score of a batch of statements . . . 3

2. Expectancy of the score for completely random answers to all statements **3**
2.1. Generating random answers . . . 3
2.2. Generating random statements . . . 4
2.3. Calculating the score for all possible combinations of statements . . . 5
2.4. The score for a batch can't be negative . . . 7
2.5. Scores by random answers . . . 7
2.6. Taking into account probabilities . . . 7
2.7. Overall expectancy . . . 8

3. Calculating all possible scenarios **8**
3.1. Function for guessing the answers . . . 8
3.2. Cases one by one . . . 9
3.2.1. 0 true, 0 false statements . . . 9
3.2.2. 0 true, 1 false statement . . . 10
3.2.3. 0 true, 2 false statements . . . 10
3.2.4. 0 true, 3 false statements . . . 11
3.2.5. 1 true, 0 false statements . . . 11
3.2.6. 1 true, 1 false statement . . . 11
3.2.7. 1 true, 2 false statements . . . 12
3.2.8. 2 true, 0 false statements . . . 12
3.2.9. 2 true, 1 false statement . . . 12
3.2.10. 3 true, 0 false statements . . . 13
3.3. Final results . . . 13

TUWEL enables the creation of quizzes with a special score system: for a given batch of statements which must be marked as true or false, the score is calculated as follows: each true statement which is marked as true by the student gives 1 point divided by the number of true statements in the batch, and each false statement which is marked as true takes off 1 point, divided by the number of false statements in the batch. Statements which are answered false don't result in positive or negative points.

Here we are interested in the particular case where there are four statements per batch, and there are five boxes to check: one for each statement, and one "none of the above" box to check in case none of the statements are true. This is not strictly necessary but is made so that it's impossible to get any points without checking any boxes, and is a convenient way to give points when there are no true statements. It also introduces a bias in the scoring system: it artificially increases the number of false "statements", therefore decreasing the potential penalty of guessing true compared to the potentially missed opportunity of guessing false. We will see that this results in a statistical benefit in guessing true in the majority of cases.

1. Putting the quiz logic into functions

1.1. Four statements, five boxes to check

First, we need to clearly distinguish the four statements, which are answered, and the five boxes, which are checked. Guessing is done on the statements, but the score is determined from the checks. This function takes the answers to the four questions, and returns the boxes that would be checked according to these answers:

```
n_q <- 4 # number of questions per batch

answers_to_checks <- function(answers) {
  # if all booleans in x are FALSE:
  if (!any(answers)) {
    # if none of the statements are true, the last box "none of the above" is
    # checked
    return(c(answers, TRUE))
  } # otherwise
  return(c(answers, FALSE))
  # if at least one of the statements is true, the last box is not checked
}

answers_to_checks(c(TRUE, FALSE, TRUE, FALSE))
```

```
## [1]  TRUE FALSE  TRUE FALSE FALSE
```

```
answers_to_checks(c(FALSE, FALSE, FALSE, FALSE))
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

1.2. Calculating the score of a single statement

Next, a function that calculates the score contribution of a single question:

```
question_score <- function(answer, correct_answer, n_true, n_false) {
  if (answer & correct_answer) { # statement is true, answer is TRUE
    return(1/n_true)
  }
  if (!answer) { # statement is true or false, answer is FALSE
    return(0)
  }
  return(-1/n_false) # statement is false, answer is TRUE
}

# generate all combinations for a boolean of length 2:
possibilities <- expand.grid(rep(list(c(TRUE, FALSE)), 2))

cat("For a case where there are 2 true statements and 2 false statements: \n")
```

```
## For a case where there are 2 true statements and 2 false statements:
```

```
for (i in 1:nrow(possibilities)) {
  cat("student's answer: ", possibilities[i,1],
      " correct answer: ", possibilities[i,2],
      "score: ", question_score(possibilities[i,1], possibilities[i,2], 2, 3), "\n")
}
```

```
## student's answer:  TRUE  correct answer:  TRUE score:  0.5
## student's answer:  FALSE  correct answer:  TRUE score:  0
```

```
## student's answer:  TRUE  correct answer:  FALSE score:  -0.3333333
## student's answer:  FALSE  correct answer:  FALSE score:  0
```

1.3. Calculating the score of a batch of statements

Next, a function that calculates the score based on the boxes checked, and the correct answers:

```
score <- function(checks, correct_checks) {
  # count number of TRUEs in correct_checks
  num_true_statements <- sum(correct_checks)
  num_false_statements <- length(correct_checks) - num_true_statements
  scores <- numeric(length(checks))
  for (i in 1:length(checks)) {
    scores[i] <- question_score(checks[i], correct_checks[i],
                                num_true_statements, num_false_statements)
  }
  sum(scores)
}

# randomly generate a boolean vector of length 4:
set.seed(1)
cat("answers to statements: ",
    answers <- sample(c(TRUE, FALSE), n_q, replace = TRUE), "\n")
```

```
## answers to statements:  TRUE FALSE TRUE TRUE
```

```
cat("boxes checked: ", checks <- answers_to_checks(answers), "\n")
```

```
## boxes checked:  TRUE FALSE TRUE TRUE FALSE
```

```
cat("boxes that should be checked",
    correct_checks <- c(TRUE, FALSE, TRUE, FALSE, FALSE), "\n")
```

```
## boxes that should be checked TRUE FALSE TRUE FALSE FALSE
```

```
cat("score: ", score(checks, correct_checks), "\n")
```

```
## score:  0.6666667
```

2. Expectancy of the score for completely random answers to all statements

What score do you get by answering everything randomly?

We can consider only the number of true or false statements in a batch without looking at the order of the statements, because the truth value of the statements doesn't depend on their order.

2.1. Generating random answers

First, generate all possible combinations of four booleans, where order does not matter:

```
# create an empty boolean vector of lenght n_q:
booleans <- rep(FALSE, n_q) # n_q is the number of questions per batch
combinations <- rep(list(booleans), n_q+1)
for (i in 1:n_q) {
  combinations[[i+1]][1:i] <- TRUE
```

```
}
print(combinations)
```

```
## [[1]]
## [1] FALSE FALSE FALSE FALSE
##
## [[2]]
## [1]  TRUE FALSE FALSE FALSE
##
## [[3]]
## [1]  TRUE  TRUE FALSE FALSE
##
## [[4]]
## [1]  TRUE  TRUE  TRUE FALSE
##
## [[5]]
## [1] TRUE TRUE TRUE TRUE
```

```
cat("number of combinations: ", length(combinations))
```

```
## number of combinations:  5
```

```
typeof(combinations[[1]])
```

```
## [1] "logical"
```

2.2. Generating random statements

Next, generate all possible random answers: combinations where order does matter:

```
permutations <- expand.grid(rep(list(c(TRUE, FALSE)), n_q))
permutations
```

```
##     Var1  Var2  Var3  Var4
## 1   TRUE  TRUE  TRUE  TRUE
## 2  FALSE  TRUE  TRUE  TRUE
## 3   TRUE FALSE  TRUE  TRUE
## 4  FALSE FALSE  TRUE  TRUE
## 5   TRUE  TRUE FALSE  TRUE
## 6  FALSE  TRUE FALSE  TRUE
## 7   TRUE FALSE FALSE  TRUE
## 8  FALSE FALSE FALSE  TRUE
## 9   TRUE  TRUE  TRUE FALSE
## 10 FALSE  TRUE  TRUE FALSE
## 11  TRUE FALSE  TRUE FALSE
## 12 FALSE FALSE  TRUE FALSE
## 13  TRUE  TRUE FALSE FALSE
## 14 FALSE  TRUE FALSE FALSE
## 15  TRUE FALSE FALSE FALSE
## 16 FALSE FALSE FALSE FALSE
```

```
cat("typeof(permutations[[1]]) : " , typeof(permutations[[1]]), "\n")
```

```
## typeof(permutations[[1]]) :  logical
```

```
cat("length(permutations) : ", length(permutations), "\n")
```

```
## length(permutations) :  4
```

```r
cat("nrows(permutations) : ", nrow(permutations), "\n")
```

```
## nrows(permutations) :  16
```

```r
cat(as.logical(permutations[1,]))
```

```
## TRUE TRUE TRUE TRUE
```

2.3. Calculating the score for all possible combinations of statements

for n questions, there are 2^n possible answers. Let's calculate the score for all of them:

```r
# This code chunk prints 330 lines. To make the pdf more readable the number of
# printed lines can be limited with a variable.
max_lines <- 40 # change this value to print more lines
end <- FALSE # to signal that the printed output ends
col_names <- c("0T 4F", "1T 3F", "2T 2F", "3T 1F", "4T 0F")
scores <- matrix(NA,
                 nrow=nrow(permutations),
                 ncol=length(combinations),
                 dimnames=list(NULL, col_names))
for (i in 1:length(combinations)) { # for each combination of true statements
  if (!end) {
    cat("\n if the correct answers are: ", combinations[[i]], "\n\n")
  }
  for (j in 1:nrow(permutations)) { # for each random answer
    s_answers <- as.logical(permutations[j,])
    s_checks <- answers_to_checks(s_answers)
    t_checks <- answers_to_checks(combinations[[i]])
    s <- score(s_checks, t_checks)
    scores[j,i] <- s
    if (max_lines > 0) {
      cat("student's answers: ", s_answers, "\n")
      cat("student checks these boxes: ", s_checks, "\n")
      # cat("correct answers: ", combinations[[i]], "\n")
      cat("boxes that should be checked: ", t_checks, "\n") # true checks
      cat("score: ", s, "\n")
      max_lines <- max_lines - 4
    }
    else {
      if (!end) {
        cat("\n Output is cut off here for readability. To print the full output,
            change the max_lines variable to 350.")
        end <- TRUE
      }
    }
  }
}
```

```
##
##  if the correct answers are:  FALSE FALSE FALSE FALSE
##
## student's answers:  TRUE TRUE TRUE TRUE
## student checks these boxes:  TRUE TRUE TRUE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
```

```
## score:  -1
## student's answers:  FALSE TRUE TRUE TRUE
## student checks these boxes:  FALSE TRUE TRUE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.75
## student's answers:  TRUE FALSE TRUE TRUE
## student checks these boxes:  TRUE FALSE TRUE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.75
## student's answers:  FALSE FALSE TRUE TRUE
## student checks these boxes:  FALSE FALSE TRUE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.5
## student's answers:  TRUE TRUE FALSE TRUE
## student checks these boxes:  TRUE TRUE FALSE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.75
## student's answers:  FALSE TRUE FALSE TRUE
## student checks these boxes:  FALSE TRUE FALSE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.5
## student's answers:  TRUE FALSE FALSE TRUE
## student checks these boxes:  TRUE FALSE FALSE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.5
## student's answers:  FALSE FALSE FALSE TRUE
## student checks these boxes:  FALSE FALSE FALSE TRUE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.25
## student's answers:  TRUE TRUE TRUE FALSE
## student checks these boxes:  TRUE TRUE TRUE FALSE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.75
## student's answers:  FALSE TRUE TRUE FALSE
## student checks these boxes:  FALSE TRUE TRUE FALSE FALSE
## boxes that should be checked:  FALSE FALSE FALSE FALSE TRUE
## score:  -0.5
##
##  Output is cut off here for readability. To print the full output,
##              change the max_lines variable to 350.
```

Overall, the scores look like this:

```
scores
```

```
##       0T 4F 1T 3F      2T 2F      3T 1F 4T 0F
##  [1,] -1.00  0.25  0.3333333  0.5000000  1.00
##  [2,] -0.75 -0.75 -0.1666667  0.1666667  0.75
##  [3,] -0.75  0.50 -0.1666667  0.1666667  0.75
##  [4,] -0.50 -0.50 -0.6666667 -0.1666667  0.50
##  [5,] -0.75  0.50  0.6666667  0.1666667  0.75
##  [6,] -0.50 -0.50  0.1666667 -0.1666667  0.50
##  [7,] -0.50  0.75  0.1666667 -0.1666667  0.50
##  [8,] -0.25 -0.25 -0.3333333 -0.5000000  0.25
##  [9,] -0.75  0.50  0.6666667  1.0000000  0.75
## [10,] -0.50 -0.50  0.1666667  0.6666667  0.50
```

```
## [11,] -0.50  0.75  0.1666667  0.6666667  0.50
## [12,] -0.25 -0.25 -0.3333333  0.3333333  0.25
## [13,] -0.50  0.75  1.0000000  0.6666667  0.50
## [14,] -0.25 -0.25  0.5000000  0.3333333  0.25
## [15,] -0.25  1.00  0.5000000  0.3333333  0.25
## [16,]  1.00 -0.25 -0.3333333 -0.5000000 -1.00
```

Looking at the average per category:

```
average_scores <- colMeans(scores)
average_scores
```

```
##      0T 4F      1T 3F      2T 2F      3T 1F      4T 0F
## -0.4375000  0.1093750  0.1458333  0.2187500  0.4375000
```

2.4. The score for a batch can't be negative

Now, we need to take into account the fact that a score for a batch of four statements can't be negative. So we take the maximum of 0 and x for each score x:

```
pos_scores <- pmax(scores, 0) # positive scores
pos_scores
```

```
##       0T 4F 1T 3F     2T 2F     3T 1F 4T 0F
##  [1,]     0  0.25 0.3333333 0.5000000  1.00
##  [2,]     0  0.00 0.0000000 0.1666667  0.75
##  [3,]     0  0.50 0.0000000 0.1666667  0.75
##  [4,]     0  0.00 0.0000000 0.0000000  0.50
##  [5,]     0  0.50 0.6666667 0.1666667  0.75
##  [6,]     0  0.00 0.1666667 0.0000000  0.50
##  [7,]     0  0.75 0.1666667 0.0000000  0.50
##  [8,]     0  0.00 0.0000000 0.0000000  0.25
##  [9,]     0  0.50 0.6666667 1.0000000  0.75
## [10,]     0  0.00 0.1666667 0.6666667  0.50
## [11,]     0  0.75 0.1666667 0.6666667  0.50
## [12,]     0  0.00 0.0000000 0.3333333  0.25
## [13,]     0  0.75 1.0000000 0.6666667  0.50
## [14,]     0  0.00 0.5000000 0.3333333  0.25
## [15,]     0  1.00 0.5000000 0.3333333  0.25
## [16,]     1  0.00 0.0000000 0.0000000  0.00
```

2.5. Scores by random answers

Now let's calculate the average score to see the expectancy. Per column:

```
expectancy_per_trues <- colMeans(pos_scores)
expectancy_per_trues
```

```
##     0T 4F     1T 3F     2T 2F     3T 1F     4T 0F
## 0.0625000 0.3125000 0.2708333 0.3125000 0.5000000
```

2.6. Taking into account probabilities

Earlier we ignored the order of the statements, but we do need to take into account the probability of each combination of true and false statements.

```r
probs <- choose(n_q,0:n_q) # probabilities of each combination of true and false statements
cat("probs: ", probs, "\n")
```

```
## probs:  1 4 6 4 1
```

```r
# normalise the probabilities:
probs <- probs/sum(probs)
cat("normalised probs: ", probs, "\n")
```

```
## normalised probs:  0.0625 0.25 0.375 0.25 0.0625
```

2.7. Overall expectancy

Now we can calculate the overall expectancy of answering randomly:

```r
overall_expectancy <- sum(expectancy_per_trues * probs)
overall_expectancy
```

```
## [1] 0.2929688
```

3. Calculating all possible scenarios

3.1. Function for guessing the answers

This function takes answers to some of the four statements as input, and outputs the expectancy of the score for each possible guess for the remaining unanswered statements.
For each case and for each possible guess, all possibilities for the remaining uncertain statements are printed, along with the resulting score based on the guess.
For example if you know that one statement is false, and you guess TRUE FALSE FALSE, and the outcome (actual truth value of the remaining statements) is TRUE TRUE TRUE, the score for the batch is 0.33

```r
guess <- function(n_true, n_false) {
  # Deal with irrelevant input:
  if (n_true < 0 | n_true > n_q | n_false < 0 | n_false > n_q) {
    stop("The number of true and false statements must be between 0 and", n_q)
  }
  if (n_true + n_false > n_q) {
    stop("There are only four statements per batch, so the number of true statements +
    false statements can't be more than", n_q)
  }
  if (n_true + n_false == n_q) {
    cat('You have already answered all four statements in this batch.')
    return(NULL)
  }

  # Explain the context:
  cat('You already know that', n_true, 'statements are true and', n_false,
  'statements are false. \n You have' , n_uncertain <- n_q - n_true - n_false,
      'statements left to answer, and there are', n_uncertain+1,
      "ways to guess them. \n\n")

  # Simulate all possible guesses:
  scores <- list()
  n_guesses <- n_q - n_true - n_false # number of remaining unanswered statements
  booleans <- rep(FALSE, n_guesses) # placeholders for the guesses
```

```
  guesses <- rep(list(booleans), n_guesses+1)
    for (i in 1:n_guesses) {
      guesses[[i+1]][1:i] <- TRUE
  }
  certain_answers <- c(rep(TRUE, n_true), rep(FALSE, n_false))
  permutations <- expand.grid(rep(list(c(TRUE, FALSE)), n_guesses)) # all possible cases
  for (guess in guesses) { # For each possible guess
    cat("if you guess", guess, ", these are the outcomes:", "\n")
    answers <- c(certain_answers, guess)
    checks <- answers_to_checks(answers)
    if (!length(answers) == n_q) { # check if there are 4 answers
        cat('answers: ', answers, '\n')
        stop("The length of the answers vector is not equal to the number of questions per
         batch.")
    }
    guess_scores <- numeric(nrow(permutations))
    for (j in 1:nrow(permutations)) { # for each possible combination of correct answers
      correct_answers <- c(certain_answers, as.logical(permutations[j,]))
      if (!length(correct_answers) == n_q) { # check if there are 4 answers
        cat('correct answers: ', correct_answers, '\n')
        stop("The length of the correct answers vector is not equal to the number of questions per
         batch.")
      }
      correct_checks <- answers_to_checks(correct_answers)
      score <- pmax(score(checks, correct_checks), 0) # score for the batch must be positive
      cat(as.logical(permutations[j,]),":", score, ", ")
      guess_scores[j] <- score
    }
    scores <- append(scores, list(guess_scores))
    cat("\n average: ", mean(guess_scores), "\n\n")
  }
  scores
}
```

3.2. Cases one by one

3.2.1. 0 true, 0 false statements

```
scores0T0F <- guess(0,0)
```

```
## You already know that 0 statements are true and 0 statements are false.
##  You have 4 statements left to answer, and there are 5 ways to guess them.
##
## if you guess FALSE FALSE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE TRUE : 0 , FALSE TRUE TRUE TRUE : 0 , TRUE FALSE TRUE TRUE : 0 , FALSE FALSE TRUE TRU
##  average:  0.0625
##
## if you guess TRUE FALSE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE TRUE : 0.25 , FALSE TRUE TRUE TRUE : 0 , TRUE FALSE TRUE TRUE : 0.3333333 , FALSE FALS
##  average:  0.234375
##
## if you guess TRUE TRUE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE TRUE : 0.5 , FALSE TRUE TRUE TRUE : 0 , TRUE FALSE TRUE TRUE : 0 , FALSE FALSE TRUE TR
##  average:  0.3125
```

```
##
## if you guess TRUE TRUE TRUE FALSE , these are the outcomes:
## TRUE TRUE TRUE TRUE : 0.75 , FALSE TRUE TRUE TRUE : 0.1666667 , TRUE FALSE TRUE TRUE : 0.1666667 , FA
##  average:  0.359375
##
## if you guess TRUE TRUE TRUE TRUE , these are the outcomes:
## TRUE TRUE TRUE TRUE : 1 , FALSE TRUE TRUE TRUE : 0.5 , TRUE FALSE TRUE TRUE : 0.5 , FALSE FALSE TRUE
##  average:  0.375
```

3.2.2. 0 true, 1 false statement

```
scores0T1F <- guess(0,1)
```

```
## You already know that 0 statements are true and 1 statements are false.
##  You have 3 statements left to answer, and there are 4 ways to guess them.
##
## if you guess FALSE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0 , FALSE TRUE TRUE : 0 , TRUE FALSE TRUE : 0 , FALSE FALSE TRUE : 0 , TRUE TRUE FA
##  average:  0.125
##
## if you guess TRUE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0.3333333 , FALSE TRUE TRUE : 0 , TRUE FALSE TRUE : 0.5 , FALSE FALSE TRUE : 0 , TRU
##  average:  0.2916667
##
## if you guess TRUE TRUE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0.6666667 , FALSE TRUE TRUE : 0.1666667 , TRUE FALSE TRUE : 0.1666667 , FALSE FALSE
##  average:  0.4375
##
## if you guess TRUE TRUE TRUE , these are the outcomes:
## TRUE TRUE TRUE : 1 , FALSE TRUE TRUE : 0.6666667 , TRUE FALSE TRUE : 0.6666667 , FALSE FALSE TRUE : 0
##  average:  0.5625
```

3.2.3. 0 true, 2 false statements

```
scores0T2F <- guess(0,2)
```

```
## You already know that 0 statements are true and 2 statements are false.
##  You have 2 statements left to answer, and there are 3 ways to guess them.
##
## if you guess FALSE FALSE , these are the outcomes:
## TRUE TRUE : 0 , FALSE TRUE : 0 , TRUE FALSE : 0 , FALSE FALSE : 1 ,
##  average:  0.25
##
## if you guess TRUE FALSE , these are the outcomes:
## TRUE TRUE : 0.5 , FALSE TRUE : 0 , TRUE FALSE : 1 , FALSE FALSE : 0 ,
##  average:  0.375
##
## if you guess TRUE TRUE , these are the outcomes:
## TRUE TRUE : 1 , FALSE TRUE : 0.75 , TRUE FALSE : 0.75 , FALSE FALSE : 0 ,
##  average:  0.625
```

3.2.4. 0 true, 3 false statements

```
scores0T3F <- guess(0,3)
```

```
## You already know that 0 statements are true and 3 statements are false.
##  You have 1 statements left to answer, and there are 2 ways to guess them.
##
## if you guess FALSE , these are the outcomes:
## TRUE : 0 , FALSE : 1 ,
##  average:  0.5
##
## if you guess TRUE , these are the outcomes:
## TRUE : 1 , FALSE : 0 ,
##  average:  0.5
```

3.2.5. 1 true, 0 false statements

```
scores1T0F <- guess(1,0)
```

```
## You already know that 1 statements are true and 0 statements are false.
##  You have 3 statements left to answer, and there are 4 ways to guess them.
##
## if you guess FALSE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0.25 , FALSE TRUE TRUE : 0.3333333 , TRUE FALSE TRUE : 0.3333333 , FALSE FALSE TRUE
##  average:  0.46875
##
## if you guess TRUE FALSE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0.5 , FALSE TRUE TRUE : 0 , TRUE FALSE TRUE : 0.6666667 , FALSE FALSE TRUE : 0.1666
##  average:  0.4895833
##
## if you guess TRUE TRUE FALSE , these are the outcomes:
## TRUE TRUE TRUE : 0.75 , FALSE TRUE TRUE : 0.1666667 , TRUE FALSE TRUE : 0.1666667 , FALSE FALSE TRUE
##  average:  0.4895833
##
## if you guess TRUE TRUE TRUE , these are the outcomes:
## TRUE TRUE TRUE : 1 , FALSE TRUE TRUE : 0.5 , TRUE FALSE TRUE : 0.5 , FALSE FALSE TRUE : 0.3333333 ,
##  average:  0.46875
```

3.2.6. 1 true, 1 false statement

```
scores1T1F <- guess(1,1)
```

```
## You already know that 1 statements are true and 1 statements are false.
##  You have 2 statements left to answer, and there are 3 ways to guess them.
##
## if you guess FALSE FALSE , these are the outcomes:
## TRUE TRUE : 0.3333333 , FALSE TRUE : 0.5 , TRUE FALSE : 0.5 , FALSE FALSE : 1 ,
##  average:  0.5833333
##
## if you guess TRUE FALSE , these are the outcomes:
## TRUE TRUE : 0.6666667 , FALSE TRUE : 0.1666667 , TRUE FALSE : 1 , FALSE FALSE : 0.75 ,
##  average:  0.6458333
##
## if you guess TRUE TRUE , these are the outcomes:
```

```
## TRUE TRUE : 1 , FALSE TRUE : 0.6666667 , TRUE FALSE : 0.6666667 , FALSE FALSE : 0.5 ,
##  average:  0.7083333
```

3.2.7. 1 true, 2 false statements

```
scores1T2F <- guess(1,2)
```

```
## You already know that 1 statements are true and 2 statements are false.
##  You have 1 statements left to answer, and there are 2 ways to guess them.
##
## if you guess FALSE , these are the outcomes:
## TRUE : 0.5 , FALSE : 1 ,
##  average:  0.75
##
## if you guess TRUE , these are the outcomes:
## TRUE : 1 , FALSE : 0.75 ,
##  average:  0.875
```

3.2.8. 2 true, 0 false statements

```
scores2T0F <- guess(2,0)
```

```
## You already know that 2 statements are true and 0 statements are false.
##  You have 2 statements left to answer, and there are 3 ways to guess them.
##
## if you guess FALSE FALSE , these are the outcomes:
## TRUE TRUE : 0.5 , FALSE TRUE : 0.6666667 , TRUE FALSE : 0.6666667 , FALSE FALSE : 1 ,
##  average:  0.7083333
##
## if you guess TRUE FALSE , these are the outcomes:
## TRUE TRUE : 0.75 , FALSE TRUE : 0.1666667 , TRUE FALSE : 1 , FALSE FALSE : 0.6666667 ,
##  average:  0.6458333
##
## if you guess TRUE TRUE , these are the outcomes:
## TRUE TRUE : 1 , FALSE TRUE : 0.5 , TRUE FALSE : 0.5 , FALSE FALSE : 0.3333333 ,
##  average:  0.5833333
```

3.2.9. 2 true, 1 false statement

```
scores2T1F <- guess(2,1)
```

```
## You already know that 2 statements are true and 1 statements are false.
##  You have 1 statements left to answer, and there are 2 ways to guess them.
##
## if you guess FALSE , these are the outcomes:
## TRUE : 0.6666667 , FALSE : 1 ,
##  average:  0.8333333
##
## if you guess TRUE , these are the outcomes:
## TRUE : 1 , FALSE : 0.6666667 ,
##  average:  0.8333333
```

3.2.10. 3 true, 0 false statements

```
scores3T0F <- guess(3,0)
```

```
## You already know that 3 statements are true and 0 statements are false.
##  You have 1 statements left to answer, and there are 2 ways to guess them.
##
## if you guess FALSE , these are the outcomes:
## TRUE : 0.75 , FALSE : 1 ,
##  average:  0.875
##
## if you guess TRUE , these are the outcomes:
## TRUE : 1 , FALSE : 0.5 ,
##  average:  0.75
```

3.3. Final results

Based on the outputs in the previous section:
- if you know that one statement is true, and you don't know about the 3 others, you will get the highest score expectancy by guessing 1 or 2 remaining statements to be true.
- if you know that at least 2 statements are true, you will get the highest score expectancy by guessing all remaining statements to be false.
- in all other cases, you will get the highest score expectancy by guessing all remaining statements to be true.

Summary table:

```
# make table with knitr kable:
pacman::p_load(knitr)
table <- data.frame(
  "What_you_already_answered" = c("0 T, 0 F", "0 T, 1 F", "0 T, 2 F",
                                  "0 T, 3 F", "1 T, 0 F", "1 T, 1 F",
                                  "1 T, 2 F", "2 T, 0 F", "2 T, 1 F",
                                  "3 T, 0 F"),
    "Best_guess" = c("TTTT", "TTT", "TT", "T or F", "TTF or TFF", "TT", "T", "FF",
                     "T or F", "F")
)
kable(table, format = "markdown", caption = "summary table")
```

Table 1: summary table

What_you_already_answered	Best_guess
0 T, 0 F	TTTT
0 T, 1 F	TTT
0 T, 2 F	TT
0 T, 3 F	T or F
1 T, 0 F	TTF or TFF
1 T, 1 F	TT
1 T, 2 F	T
2 T, 0 F	FF
2 T, 1 F	T or F
3 T, 0 F	F