R documentation

of all in
'/Users/barryzeeberg/personal/hearts/score/score.testing.01.23.19/fina
version/man'

August 1, 2021

R topics documented:

`hearts.probabilities.package-package`
hearts.probabilities.package

Description

Hearts Card Game: Given Each Player's Score, Compute Each Player's Probability of Winning

Details

Package:	hearts.probabilities.package
Type:	Package
Version:	1.0
Date:	2019-07-20
License:	GPL(>=2)

Author(s)

Barry Zeeberg

Maintainer: Barry Zeeberg barryz2013@gmail.com

References

only 2 books I am aware of in which hearts technique is presented:

The Complete Win at Hearts by Joseph Andrews (Jan 30, 2004)

How to be a Consistent Winner in the Most Popular Card Games by John R. Crawford (1953)

online heart card game `hearts.vex.net`

probably hundreds of books in which bridge technique is presented:

The Play of the Cards by Terence Reese and Albert Dormer (1967)

Master Play by Terence Reese (1966)

Killing Defense at Bridge by H. W. Kelsey (1966)

Watson's the Play of the Hand at Bridge by Louis H. Watson (1959)

Card Reading - The art of guessing right at the bridge table by Eric Jannersten (1972)

`adjustTargetInterval()`
adjustTargetInterval()

Description

compute the endpoints of the target intervals

Usage

```
adjustTargetInterval(lint,targ,level,dev)
```

Arguments

`lint`	numeric vector returned by interpolateSquashFunction()
`targ`	vector of endpoints for the target intervals for the 4 players
`level`	character string "novice", "intermediate", "advanced", or "expert"
`dev`	parameter passed to targetRangeDriver()

Value

returns no values, but has side effect of computing the endpoints of the target intervals

Author(s)

Barry Zeeberg

See Also

winner.awk

`BINCV()` *BINCV()*

Description

compute mean and standard deviation of coefficients of variation that lie within a certain range

Usage

```
BINCV(CVs,PROBs,BCVs,bmin,bmax)
```

Arguments

`CVs`	numeric array of coefficients of variation among probabilities that theoretically should be identical
`PROBs`	numeric array of prob values
`BCVs`	binned values of CVs [computed in BINCV()]
`bmin`	min value for current bin
`bmax`	max value for current bin

Value

returns no values, but has side effect of populating the array BCVs

Author(s)

Barry Zeeberg

See Also

computeNoise.awk

`computeNoise.awk` *computeNoise.awk*

Description

estimate the noise within the results of a winner.awk run compare p vals that should theoretically be identical

Usage

```
gawk -f computeNoise.awk xls odir
```

Arguments

`xls`	output file for a winner.awk run
`odir`	character string containing the path name for the output directory

Value

returns no value, but has side effect of estimating the noise within the results of a winner.awk run

Author(s)

Barry Zeeberg

`count_26()` *count_26()*

Description

how many players receive exactly 26 points?

Usage

```
count_26(score)
```

Arguments

`score`	integer array of the scores of the 4 players

Value

returns integer number of players that received exactly 26 points

Author(s)

Barry Zeeberg

See Also

scenario.awk

`count_npoints()` *count_npoints()*

Description

how many players receive >= npoints?

Usage

```
count_npoints(score,comp_op,npoints)
```

Arguments

`score`	integer array of the scores of the 4 players
`comp_op`	character string containing the name of a comparison operator such as "eq"
`npoints`	integer containing a number of points

Value

returns an integer containing the number of players that received >= npoints

Author(s)

Barry Zeeberg

See Also

scenario.awk

`FCV()` *FCV()*

Description

compute mean and standard deviation of coefficients of variation

Usage

```
FCV(CVs,stats)
```

Arguments

`CVs`	numeric array of coefficients of variation among probabilities that theoretically should be identical
`stats`	numeric array containing mean and std for CVs [computed in FCV()]

Value

returns no values, but has sude effect of computing mean and standard deviation of coefficients of variation

Author(s)

Barry Zeeberg

See Also

computeNoise.awk

`generateSkillLevels()` *generateSkillLevels()*

Description

assign bias weight depending on general level of players
ie, novice will not try to hit low whatsoever
ie, expert try to hit low as often as possible AND as effectively as possible
these values are based on intuition,
and they can be adjusted manually after examining tabulation generated in adjustTargetInterval()

Usage

```
generateSkillLevels(skill)
```

Arguments

`skill`	vector of bias weight that depend on general level of players these go into determining the endpoints of the target intervals skill["novice"]=0.00 skill["intermediate"]=0.25 skill["advanced"]=0.50 skill["expert"]=1.00

Details

more details are given in the manual entry for winner.awk

Value

returns the matrix skill

Author(s)

Barry Zeeberg

See Also

winner.awk

`generateSquashFunction()`
generateSquashFunction()

Description

the numerical values in squash were subjectively determined by exhaustive trial and error in order to result in the behavior that I think is reasonable

Usage

```
generateSquashFunction(squash,maxv)
```

Arguments

`squash`	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points
`maxv`	integer equal to skill[level] (see man page for winner.awk)
`squashn`	squash5, squash10, squash20, squash 100 are manually-selected values that give the squash table the correct shape

Value

returns no values, but has side effect of generating the squash matrix

Author(s)

Barry Zeeberg

See Also

winner.ask

`interpolateSquashFunction()`
interpolateSquashFunction()

Description

search for the tabulated point that is closest above the data point

Usage

```
interpolateSquashFunction(squash,n,v)
```

Arguments

squash	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points
n	integer = length(squash)
v	numeric = a dev value to be mapped to squash function

Value

returns the tabulated point that is closest above the data point

Author(s)

Barry Zeeberg

See Also

winner.awk

---

intitializeBias() *intitializeBias()*

Description

call functions to generate skill levels and generate squash function

Usage

```
intitializeBias(level,squash)
```

Arguments

level	character string "novice", "intermediate", "advanced", or "expert"
squash	2 D matrix with 9 rows and 2 columns column 1 contains a deviation from the mean (dev) value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points

Value

returns no values, but has side effect of calling functions to generate skill levels and generate squash function

Author(s)

Barry Zeeberg

See Also

winner.awk

`linearInterpolate()` *linearInterpolate()*

Description

linear interpolation between end points of tabulated interval

Usage

```
linearInterpolate(squash,n,i,v)
```

Arguments

`squash`	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points
`n`	integer = length(squash)
`i`	integer = position in squash table to perform interpolation
`v`	numeric = a dev value to be mapped to squash function

Value

returns the value of the linear interpolation between end points of tabulated interval

Author(s)

Barry Zeeberg

See Also

winner.awk

matchProbs.awk *matchProbs.awk*

Description

compare a single score for 4 players across multiple files that had been generated by winner.awk
this allows spot check reproducibility of repeated runs
or comparison the effect of number of iterations
or changing parameters such as "novice" versus "expert" level
or calibration of proper squash function parameters

Usage

```
gawk -f  matchProbs.awk xls1 xls2 . . . score00 score1 score2 score3
gawk -f  matchProbs.awk `ls ../results/*xls` score00 score1 score2 score3
```

Arguments

xlsn	character string containing the path name for an xls output file generated by winner.awk or alternatively ‘ls ../results/*xls‘ to process all xls in ../results
scoren	score for player n

Details

note that the 4 score parameters must be a valid set that is present in the data file
eg, must be a score that could actually occur in hearts game
ie, sum of 4 scores must be multiple of 26
and scores must be given in non-descending order
each individual score must be less than 100

Value

returns no values, but has side effect of generating a table like Table 6 in the manucript.
one row for each inut file, 4 columns of probabilities for the 4 players

Author(s)

Barry Zeeberg

See Also

winner.ask

match_individual() *match_individual()*

Description

how many players receive exactly 26 points?

Usage

```
match_individual(score,p0,perm,meta,vals)
```

Arguments

score	integer array of the scores of the 4 players
p0	numeric array of probabilities that match the 4 scores in array score, computed by match_initial_scores()
perm	integer permutation matrix to allow looping through all combinations of orders of 4 scores
meta	does not seem to be used in current implementation
vals	integer matrix of each row of which contains the 4 players' scores

Details

Details about the permutation matrix are given in the manuscript

Value

returns integer = length(vals)

Author(s)

Barry Zeeberg

See Also

scenario.awk

---

match_initial_scores() *match_initial_scores()*

Description

does the current line match the initial set of scores?

Usage

```
match_initial_scores(score,p0)
```

Arguments

score	integer array of scores for the 4 players
p0	numeric array of probabilities that match the 4 scores in array score

Value

returns integer 1 if the current line matches the initial set of scores, 0 otherwise

Author(s)

Barry Zeeberg

See Also

scenario.awk

---

match_total() *match_total()*

Description

total score with added points must equal original total + 26

Usage

```
match_total(orig,tot)
```

Arguments

orig	integer = total of 4 players' scores
tot	integer = 79 if moonflag == 1 = 26 if moonflag == 0

Value

returns integer value = orig + tot

Author(s)

Barry Zeeberg

See Also

scenario.awk

`meanDev()` *meanDev()*

Description

compute mean and deviations from mean for the current set of 4 scores for the 4 players

Usage

```
meanDev(sc)
```

Arguments

`sc`	integer vector of the 4 players' scores

Value

returns no values, but has side effect of computing mean and deviations from mean for the current set of 4 scores for the 4 players

Author(s)

Barry Zeeberg

See Also

winner.awk

`printTimeinterval()` *printTimeinterval()*

Description

formatted printing of how long the simulation took, for archiving

Usage

```
printTimeinterval(sec)
```

Arguments

`sec`	systime()-systime_start

Value

returns formatted printing of how long the simulation took

Author(s)

Barry Zeeberg

See Also

winner.awk

`prob.score.scatter.awk` *prob.score.scatter.awk*

Description

for a given original position, retrieve the subsequent points taken and probabilities from the scenario output file
this allows analysis of the relationship between points taken and probabilities, to assess when to stop a moon.
prob.score.scatter.awk was used to retrieve the data from the scenario.awk output file to construct figure 5 in the manuscript.

Usage

```
gawk -f prob.score.scatter.awk scenario.dir rp1 rp2 rp3 rp4
```

Arguments

`scenario.dir`	character string containing the path name for the scenario directory
`rpn`	optional reference probability to include in scatter plot file

Value

returns no values, but has side effect of generating an output file whose lines are of the form:
subsequent_probability TAB subsequent_score - original_score

Author(s)

Barry Zeeberg

`rand2target()` *rand2target()*

Description

map the rand number to 1 of 4 target ranges ranges indexed as 0,1,2,3

Usage

```
rand2target(sc,squash,dev,level,targ)
```

Arguments

sc	
squash	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried rectangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points
dev	parameter passed to targetRangeDriver()
level	character string "novice", "intermediate", "advanced", or "expert"
targ	parameter passed to targetRangeDriver()

Value

returns integer index of the target range that rand mapped to

Author(s)

Barry Zeeberg

See Also

winner.awk

---

s3start *s3start*

Description

s3start is selected so that the sum s0 + s1 + s2 + s3 is a valid score, ie a multiple of 26

Usage

```
s3start(s,initial,final)
```

Arguments

s	integer vector of length 4 s[0] = loop index for score of player 0, usually runs from 20 to 99 s[1] = loop index for score of player 1, usually runs from 20 to 99 s[2] = loop index for score of player 2, usually runs from 20 to 99 s[3] = loop index for score of player 3, selected so that the sum s0 + s1 + s2 + s3 is a valid score, ie a multiple of 26
initial	initial value for loop indices s[0], s[1], s[2], usually runs from 20 to 99
final	obsoleted, no longer used

Details

max value for sum + initial = s[0] + s[1] + s[2] + 100 = 99 + 99 + 99 + 100 = 297 + 100 = 397

Value

returns an integer value to be used as the initial value for the loop index for player 3
selected so that the sum s0 + s1 + s2 + s3 is a valid score, ie a multiple of 26

Author(s)

Barry Zeeberg

See Also

winner.awk

`scenario.awk` *scenario.awk*

Description

given a current score and 26 points to be added to the score, how are those points best distributed to optimize the chance of winning?

this program computes the more general solution provides the p vals for all scenarios for all players
generates a histogram of p vals for each player

Usage

```
gawk -f  matchProbs.awk xls score00 score1 score2 score3 moonflag
```

Arguments

`xls`	character string containing the path name for an xls output file generated by winner.awk
`scoren`	score for player n
`moonflag`	integer if 1 analyze scenario in which one player moons if 0 analyze scenario in which there is no moon

Details

note that the 4 score parameters must be a valid set that is present in the data file
eg, must be a score that could actually occur in hearts game
ie, sum of 4 scores must be multiple of 26
and scores must be given in non-descending order
each individual score must be less than 100

Value

returns no values, but has side effect of generating a histogram of p vals for each player

Author(s)

Barry Zeeberg

`targetRangeDriver()` *targetRangeDriver()*

Description

co-ordinate procedures to map random number to target range
compute mean and deviations from mean
linear interpolation within Squash Function
compute the biased end points of the biased target ranges

Usage

```
targetRangeDriver(sc,squash,dev,level,targ)
```

Arguments

`sc`	integer vector of the 4 players' scores
`squash`	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points
`dev`	numeric vector of the deviation from the mean score for 4 players [computed in meanDev()]
`level`	character string "novice", "intermediate", "advanced", or "expert"
`targ`	vector of endpoints for the target intervals for the 4 players [computed in adjustTargetInterval()]

Value

returns no values, but has side effects:
compute mean and deviations from mean
linear interpolation within Squash Function
compute the biased end points of the biased target ranges

Author(s)

Barry Zeeberg

See Also

winner.awk

winner.awk *winner.awk*

Description

Given each player's score, compute each player's probability of winning

Usage

```
gawk -f winner.awk ODIR="odir" DEBUG=0
squash5=0.10 squash10=0.40 squash20=0.50 squash100=0.60
Pmoon=0.05 N=1000 BIAS=1 level= initial=20 final=99 del=1
```

Arguments

ODIR	character string containing the path name for the output directory
DEBUG	Boolean if 1 then debug info is printed, typically set to 0
squashn	2 D matrix with 9 rows and 2 columns column 1 contains a dev value column 2 contains the corresponding squashed value of dev I tried to find a function to generate these points, but nothing had the right shape I tried retangular hyperbola and atan etc So I was forced to tabulate and then interpolate using the tabulated points squash5, squash10, squash20, squash 100 are manually-selected values that give the squash table the correct shape
Pmoon	a priori probability that a player will shoot the moon, typically 0.05
N	number of simulations, typically 1000
BIAS	Boolean if 1, functions apply bias depending on general skill level bias is implemented by altering the target interval length for mapping randoms. with no bias, all 4 intervals are of length 1
level	general level of experience of the 4 players: "novice", "intermediate", "advanced", or "expert"
initial	initial value for the score of the first player, typically 20
final	initial value for the score of the first player, typically 99
del	increment for the score of the first player, typically 1

Value

returns a character string containing the pathname of the output file,
and a character string containing the name of the program source directory

Author(s)

Barry Zeeberg

Index