Categorical Risk Factors

# Motivation

For the disease risk prediction program, disease models can include categorical risk factors. This document describes how such risk factors are modelled, and how that is implemented. This document also describes testing done to ensure the correctness of the categorical risk factor modelling.

# Introduction

Categorical personal attributes have been discovered to be risk factors for some multifactorial diseases. Examples

* Sex - females are twice as likely to suffer Major Depression (MD) as males.
* Smoker - smoking is a risk factor for heart disease and various cancers. If coded as Smoker/Non-smoker the variable is categorical. It could also be coded as continuous, e.g. Smoker Years. Continuous risk factors are not considered in this document.

Single Nucleotide Polymorphism (SNP) genotypes are another important class of categorical risk factor.

This document details how categorical risk factors can be incorporated into the disease model implemented in the MultifactorialDiseaseRiskCalculator program, and the online equivalent - <http://grass.cgs.hku.hk:3838/mdrc/current> .

# Disease Model incorporating categorical risk factors

Multifactorial disease is modelled via a liability threshold model. Categorical risk factors are incorporated into this in the following way. The disease is modelled using a liability threshold model in which the population liability distribution is a mixture distribution of risk factor strata liability distributions. The proportions of this mixture are the frequencies of the risk factor strata/categories. The liability distribution within each risk factor stratum is assumed to be Gaussian. The means may differ across the risk factor strata liability distributions but the variances are constrained to be equal. These strata means and variances are constrained by the population liability distribution, which is constrained to have a mean of 0 and variance of 1.

## Example

Consider a disease model, which includes a three category risk factor, as follows.

Disease lifetime risk = 0.05

|  |  |  |
| --- | --- | --- |
| **Strata** | **Frequency** | **Risk Relative**  **to red strata** |
| Red | 0.5 | 1 |
| Green | 0.2 | 100 |
| Blue | 0.3 | 200 |

The solution for this scenario is given in the following table and also depicted in Figure 1. The strata are identified by 3 colours. The upper plot shows the probability density function for liability for each stratum weighted by its population frequency. The lower plot shows the population liability distribution. It is the sum of the upper plot functions, it has a mean of 0 and variance of 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Strata** | **Frequency** | **Risk Relative**  **to red strata** | **Liability Mean** | **Liability Variance** | **Risk** |
| Red | 0.5 | 1 | -0.69 | 0.515 | 0.00062 |
| Green | 0.2 | 100 | 0.525 | 0.515 | 0.06211 |
| Blue | 0.3 | 200 | 0.8 | 0.515 | 0.12422 |

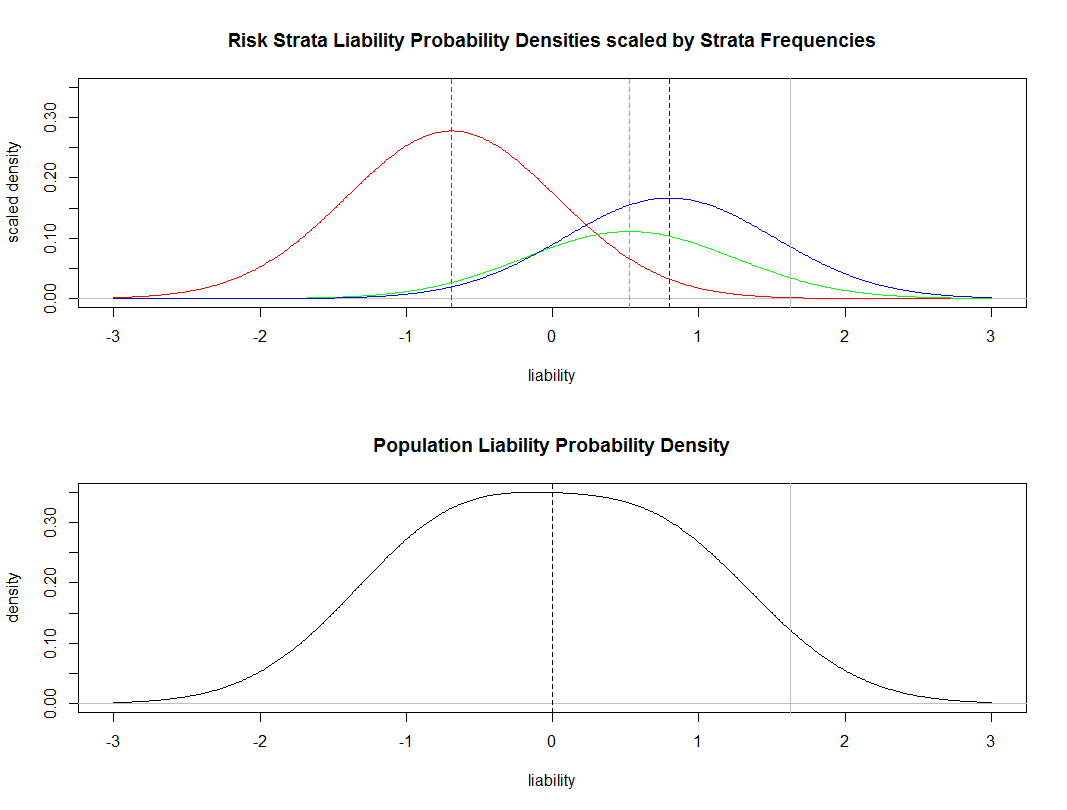
Figure

Upper – the liability distribution within each risk factor stratum.

Lower – the population liability distribution.

Strata liability means are shown by vertical dashed lines.

The vertical grey line is the critical threshold for the disease.



### Technical

The solution to the scenario was found using

./CategoricalRiskFactorTestHarness.R 0.05 "0.5 0.2 0.3" "1 100 200"

Then the plots above were generated from the solution via

./plotCatRiskFactorSolution.R

(for details see section 4)

# Method for modelling Categorical Risk Factors

## Inputs

The following is the minimum set of inputs the user must supply in order to allow a risk factor with *n+1* categories to be incorporated into a disease model

* K = Disease lifetime risk
* **m** = an *n* vector containing relative risks. These are relative to an arbitrarily chosen reference stratum
* **f** = an *n* vector containing the frequencies of the non-reference strata

## Directly Calculable Parameters

Given these inputs, one can directly calculate the following

* The frequency of all strata
* The risk for all strata

The reference stratum frequency and risk , can be directly calculated via

The lifetime risks for the non-reference strata can be calculated via

## Simultaneous Equations

We can then specify *n+1* simultaneous equations

where

* = Gaussian pdf
* = intra risk stratum liability variance

These equations contain *n+1* unknowns

* *T* = critical threshold
* = liability mean for *i* th non-reference stratum (of which there are *n*)

## Solving the simultaneous equations

Non-linear equations are difficult to solve analytically. Instead we used optimisation algorithms (intelligent guessing), e.g. gradient descent, to solve this set of equations. The guessing is done in the parameter space of the *n+1* unknowns. The optimisation algorithms iteratively improve their guess using the discrepancy between the true strata risks (pre-calculated in section 3.2), and the strata risks that would follow given the current guess of the *n+1* unknowns. The performance of the optimisation algorithms depends largely on the quality of the discrepancy function used.

## Discrepancy Function

The discrepancy function initially used to drive the optimisation was the population average of the square of the Euclidean distance between the true per strata lifetime risks, and per strata lifetime risks according to current guess, i.e.

**where**

* = the true per strata lifetime risks
* = per strata lifetime risks according to the current guess

That discrepancy function turned out to perform poorly when the risk factor contained very rare categories. In such cases the solution found, although accurate for common strata, tended to be inaccurate for very rare categories.

In the light of this, the discrepancy function was modified to the following

The idea behind this new discrepancy function is that if either of the two terms is solved then the other term dominates the discrepancy. The two terms are

* The population average (i.e. as before)
* The discrepancy averaged across categories when all categories are given equal weighting

In effect this upweights rare categories and guards against the possibility that a good fit for the population may come at the price of a disastrous fit for very rare strata.

### Parameter Space Constraint

Not all values of the parameter space give valid solutions. It is possible to specify a parameter vector that implies the liability variance explained by risk factor strata liability means exceeds 1. As the population liability variance is constrained to be 1, this implies a negative (i.e. impossible) .

If such a parameter vector occurs, we set the discrepancy function to

The rationale behind this is as follows. Any valid risk discrepancy must lie between 0 and 1, whereas this value always exceeds that ( being negative). Consequently this discrepancy is always worse than that for any valid guess. Also the more negative is the larger the discrepancy is. Thus the gradient of the function points in the direction of the valid parameter space.

## Optimisation Methods

Optimisation was based around use of the optim() R function. Two optimisation methods were used

* Nelder-Mead
* BFGS (a quasi-Newton method)

For dichotomous risk factors, Nelder-Mead is not recommended in such cases only BFGS is tried.

For risk factors with more than two categories, Nelder-Mead is used by default. If this fails to converge on a solution then BFGS is tried.

## Performance

The solution we developed works well over a wide range of categorical risk factor scenarios. In fact the range far exceeds that likely to be encountered in actual disease models, in terms of the number of categories, the relative risks, and the rarity of categories. Section 6.1 gives an idea of the range of scenarios over which our solution works.

# Validation

## Test Harness

The algorithm and its implementation were validated and characterised using a program known as a test harness. A set of test cases are created. These may be valid or invalid categorical risk factor specifications. The test harness is applies to each of these test cases in turn.

The test harness CategoricalRiskFactorTestHarness.R does the following

* reads command line inputs specifying a categorical risk factor scenario
* validates these inputs
* uses optimisation to find a solution for the scenario.
* calculates metrics regarding the quality of the solution.
* writes the solution to file – CategoricalRiskFactorTestHarness.out.tsv
  + The solution is written as
    - The liability distribution (mean and variance) of each stratum
    - The liability threshold
* Creates figures plotting how the optimisation approached the solution over increasing numbers of iterations. These it writes to file – trace-plot-<TIMESTAMP>.pdf

## Further validation

When the test harness runs successfully it writes the solution to file. The solution can be converted to graphical form by entering the following at the command prompt

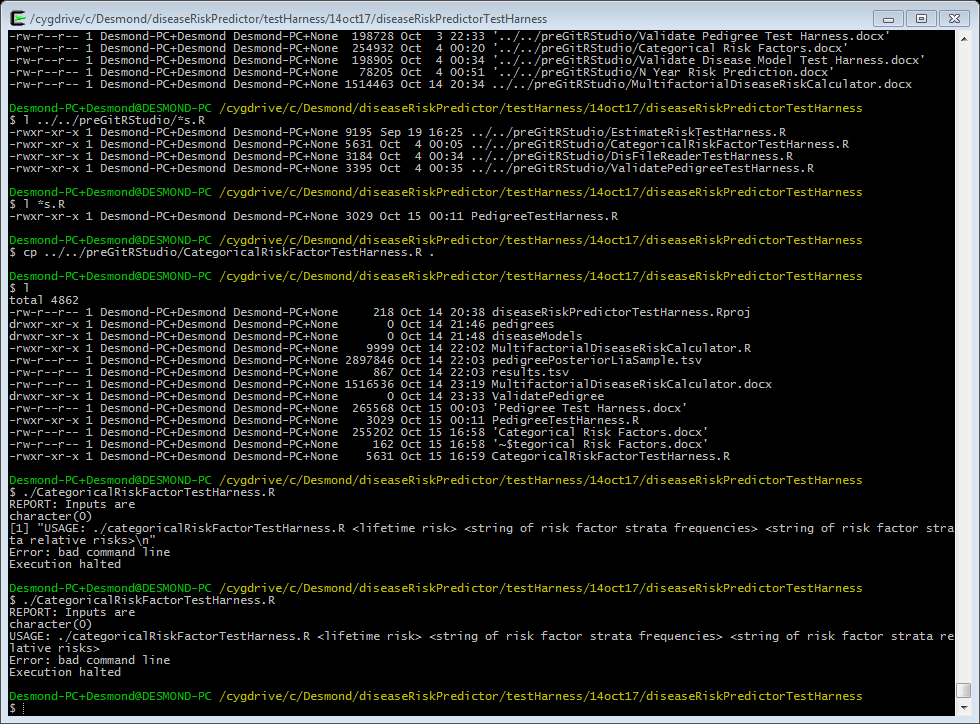
./plotCatRiskFactorSolution.R

This program reads the test harness’s solution file. It generates plots of the risk strata and population liability distributions, an example of those plots is shown in Figure 1. It also performs a validation. The area under the curve is calculated for sectors of the strata and population liability distributions. From this the risk and relative risk is calculated for each stratum. These are then compared to the scenario for which the solution was found.

## Check setup

Input the following at the command prompt

./CategoricalRiskFactorTestHarness.R

The output should include a usage report telling you what constitutes a valid command line. This is what that looked like when run on my machine from a Cygwin prompt 

If you don’t see the usage, then the program probably didn’t run. Possible reasons

* The test harness must have execute permissions
* R must be on your PATH

## Test Harness Example

Enter the following at the command prompt

./CategoricalRiskFactorTestHarness.R 0.05 "0.5 0.2 0.3" "1 100 200"

This command finds the solution to the section 2.1 scenario.

The program reports progress to screen. Near the end of its output it reports the solution found

REPORT: The solution is given below

SOLUTION method nofAttempts freq relRisk bRefStratum risk propPopAff liaMean liaVarInter liaVarIntra liaThr\_imp risk\_imp propPopAff\_imp

1 SOLUTION Nelder-Mead 1 0.5 1 TRUE 0.000621118 0.000310559 -0.6896837 0.4847435 0.5152565 1.628137 0.000621118 0.000310559

2 SOLUTION Nelder-Mead 1 0.2 100 FALSE 0.062111801 0.012422360 0.5246388 0.4847435 0.5152565 1.628123 0.062109354 0.012421871

3 SOLUTION Nelder-Mead 1 0.3 200 FALSE 0.124223602 0.037267081 0.7997136 0.4847435 0.5152565 1.628163 0.124231052 0.037269316

The columns of this table are

* SOLUTION – just identifies what this table contains
* Method – the optimisation method
* nofAttempts – number of optimisation attempts made (1 or 2)
* Freq, relRisk, bRefStratum – defined directly by the command line
* propPopAff – calculable from the inputs, should sum to the lifetimeRisk
* liaMean, liaVarInter, liaVarIntra – the solution (risk strata distributions)
* liaThr\_imp, risk\_imp, propPopAff\_imp – parameter values implied by the solution (risk\_imp, propPopAff\_imp should equal risk, propPopAff)

This table is written to file.

Diagnostics on the quality of the solution found are then reported

REPORT: Diagnostics on how good a solution it is are given below

DIAGNOSTICS liaThr\_imp\_var risk\_imp\_mse popLiaMean\_dev lifetimeRisk\_dev

1 DIAGNOSTICS 4.204302e-10 2.049579e-11 0 1.745358e-06

The columns of this table are

* DIAGNOSTICS – just identifies what this table contains
* liaThr\_imp\_var – the across strata variance of the liability threshold implied by the solution
* risk\_imp\_mse – Mean standard error across strata of the risk implied by the solution
* popLiaMean\_dev – deviance of population liability mean implied by the solution from zero
* lifetimeRisk\_dev – the across strata deviance of the risk threshold implied by the solution from the true strata risks

These numbers should all be low, indicating the simultaneous equations were successfully solved.

If the program completes without error, the last line reported is

REPORT: Completed Successfully

# Test Cases

## Pass Cases

This section details tests where the program is expected to find a good solution to the scenario. The following file contains a list of test harness commands that can be copy and pasted to the command prompt

* CategoricalRiskFactorTestHarness.R.tests.pass.txt

The first few lines of it are

#### Categorical risk factor solver tests

#### good solutions should be found for all these scenarios

#### indicated by low diagnostics and no errors

./CategoricalRiskFactorTestHarness.R 0.1 "0.9 0.05 0.05" "1 2 3"

#### swapping stratum order

# these should give the same results per equiv statum

./CategoricalRiskFactorTestHarness.R 0.1 "0.9 0.09 0.01" "1 2 9"

./CategoricalRiskFactorTestHarness.R 0.1 "0.09 0.9 0.01" "2 1 9"

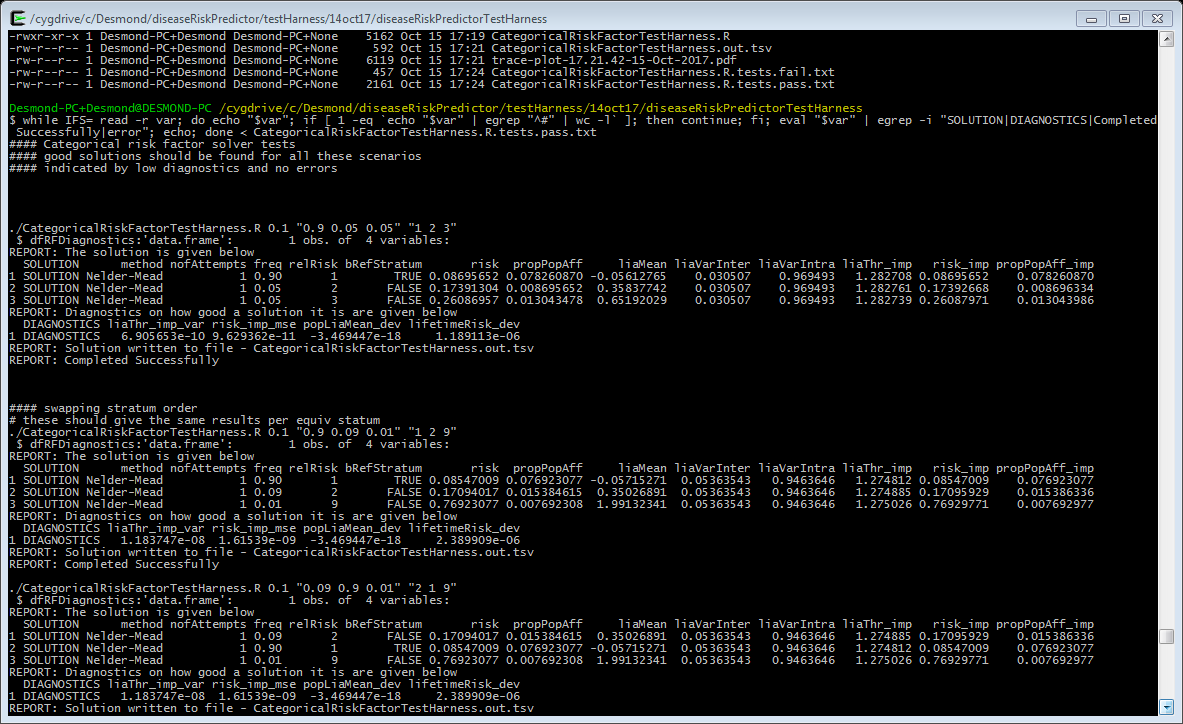
...

Rather than run each command individually, you might want to just run all the commands therein, checking just

* the solution
* diagnostics
* and whether any errors were reported.

The following command allows you to do that. It reads each command from the file, reports it to screen, runs it, and filters out the salient program output.

while IFS= read -r var; do echo "$var"; if [ 1 -eq `echo "$var" | egrep "^#" | wc -l` ]; then continue; fi; eval "$var" | egrep -i "SOLUTION|DIAGNOSTICS|Completed Successfully|error"; echo; done < CategoricalRiskFactorTestHarness.R.tests.pass.txt

Enter it at the command prompt. You should get output similar to that below 

In particular

* Diagnostics should be low
* No errors should be reported

## Fail Cases

The following file contains invalid test cases. The program should detect the problem and report an appropriate error

* CategoricalRiskFactorTestHarness.R.tests.fail.txt

As before the work can be automated. The following command entered at the command prompt will rehearse all the test cases.

while IFS= read -r var; do echo "$var"; if [ 1 -eq `echo "$var" | egrep "^#" | wc -l` ]; then continue; fi; eval "$var" | egrep -i "SOLUTION|DIAGNOSTICS|Completed Successfully|error"; echo; done < CategoricalRiskFactorTestHarness.R.tests.fail.txt

When this is done, each test case should fail and an appropriate error should be reported.