Multifactorial Disease Risk Calculator

# Introduction

MultifactorialDiseaseRiskCalculator is a program that allows disease risk to be predicted for the family members based on personal attributes (e.g. age and sex) and upon their familial relationship to other family members who have the disorder in question.

The user specifies, via the command line

* a file containing the family history (pedigree) information
* a file containing a disease model for the disease in question

In response, the program predicts the disease risk for each person in the pedigree, and also their *n* year risk.

A web-based version of this program also exists at - <http://grass.cgs.hku.hk:3838/mdrc/current>

Both this program and its web-based counterpart are based on the diseaseRiskPrediction R package further described in (Campbell et al. 2017).

The code within this program provides a template of how an R script can interact with the diseaseRiskPrediction R package.

# Citation

If you want to cite the website or this program, please cite (Campbell et al. 2017).

# Installation

See the appendix.

# Operation

## Checking setup

MultifactorialDiseaseRiskCalculator is a command line program, i.e. to run the program, you enter text at a command prompt. Many command line environments are available and these differ slightly from one to the other and across operating systems. I tend to use Cygwin which provides a command line environment on my Windows machine. This provides unix-like commands and bash shell programming. Other command line interfaces may differ slightly in syntax. Notably, for Windows command lines, the path separator is \ instead of /. The instructions following are based on the assumption of working from a Cygwin or bash prompt.

Open your command line prompt and navigate to the installation directory, i.e. the directory containing file MultifactorialDiseaseRiskCalculator.R.

From the command line enter the following

./MultifactorialDiseaseRiskCalculator.R --help

You should get usage information similar to the following

Usage: ./MultifactorialDiseaseRiskCalculator.R [options]

Options:

-d DISEASEMODEL, --diseaseModel=DISEASEMODEL

path to disease model file

-p PEDIGREE, --pedigree=PEDIGREE

path to pedigree information file

-s, --stdDev

calculate standard deviation of risk estimates [default FALSE]

-i NOFITERATIONS, --nofIterations=NOFITERATIONS

number of iteratations to use for standard deviation estimation [default 10]

-y NOFYEARS, --nofYears=NOFYEARS

estimate risk of becoming affected within the next n years [default 5]

-n NOFDRAWS, --nofDraws=NOFDRAWS

number of draws from pedigree posterior liability distribution [default 20000]

-b NOFBURNIN, --nofBurnIn=NOFBURNIN

number of draws for Gibbs Sampler burn in [default 1000]

-h, --help

Show this help message and exit

The program is a set of instructions written in the R programming language. If you get usage information similar to the above then that means your environment is correctly set up for interaction of the program with R. If you don’t get this then something is wrong with your set up. For instance

* Rscript is not on your PATH
* the file does not have execute permission
* required R packages are not installed, etc.

See the installation section to correct this.

## Command line format

The usage information reported in the previous section describes what constitutes a valid command line for executing the program. The program takes named (not positional) arguments. Arguments can be in long or short form. Some arguments take values, others don’t. The latter are switches, i.e. they provide yes/no information indicated by the argument’s presence or absence. Arguments typically can be supplied in a long or a short form.

Example of equivalent long and the short form of a valued argument

* --diseaseModel=diseaseModels/dm.Con1.txt
* -d diseaseModels/dm.Con1.txt

Example of equivalent long and the short form of a switch argument

* --stdDev
* -s

If the command line input is not in the appropriate format then the program will abort returning an error message.

## Specimen Risk Prediction

Enter the following at the command line. (The command is spread out over several lines but should be entered as one line. Future commands are presented similarly.)

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.depression.txt -p pedigrees/ped.3gen.1aff.tsv

If the first few lines reported by the program are similar to the following then it has started happily enough.

REPORT: patch kinship2::plot.pedigree()

REPORT: Inputs are

VALUE: vsArgs = chr [1:4] "-d" "diseaseModels/dm.depression.txt" "-p" "pedigrees/ped.3gen.1aff.tsv"

REPORT: Program inputs are

VALUE: opt = List of 8

$ diseaseModel : chr "diseaseModels/dm.depression.txt"

$ pedigree : chr "pedigrees/ped.3gen.1aff.tsv"

$ stdDev : logi FALSE

$ nofIterations: int 10

$ nofYears : int 0

$ nofDraws : int 20000

$ nofBurnIn : int 500

$ help : logi FALSE

REPORT: Below is some diagnostic information regarding the setup

If everything is correctly installed etc. the program should run to completion, with the final output being

REPORT: Completed Successfully

What this command line did was predict disease risk for the members of the pedigree specified by ped file (pedigrees/ped.p2c3.affectedMz.tsv). The risks are predicted according to a disease model specified in file diseaseModels/dm.Con1.txt.

## Program Outputs

If the program ran correctly, it will have produced the following outputs

* Progress reports to the command line
* Output files
  + MultifactorialDiseaseRiskCalculator.R.plots.pdf
  + results.tsv
  + pedigreePosteriorLiaSample.tsv

### Progress reports to the command line

The program reports its progress as it runs. It first reports the arguments it has extracted from the command line. It reads the pedigree information from file and reports this to screen. It does the same for the disease model. After the risks have been estimated, the pedigree information is output again but with additional columns appended. These are

* lifetimeRisk – the disease lifetime risk (taken from the disease model)
* thr – the quantile of a standard normal distribution for the lifetime risk
* expressedProportionOfLifetimeRisk – the expressed proportion of lifetime risk is a personal attribute. It expresses the degree of right censoring of an unaffected affection status.
* risk – this is the prime output of interest. It is the predicted disease risk
* nYearRisk – the risk for a currently unaffected person of becoming affected within the next *n* years
* nofYears – the *n* for the nYearRisk

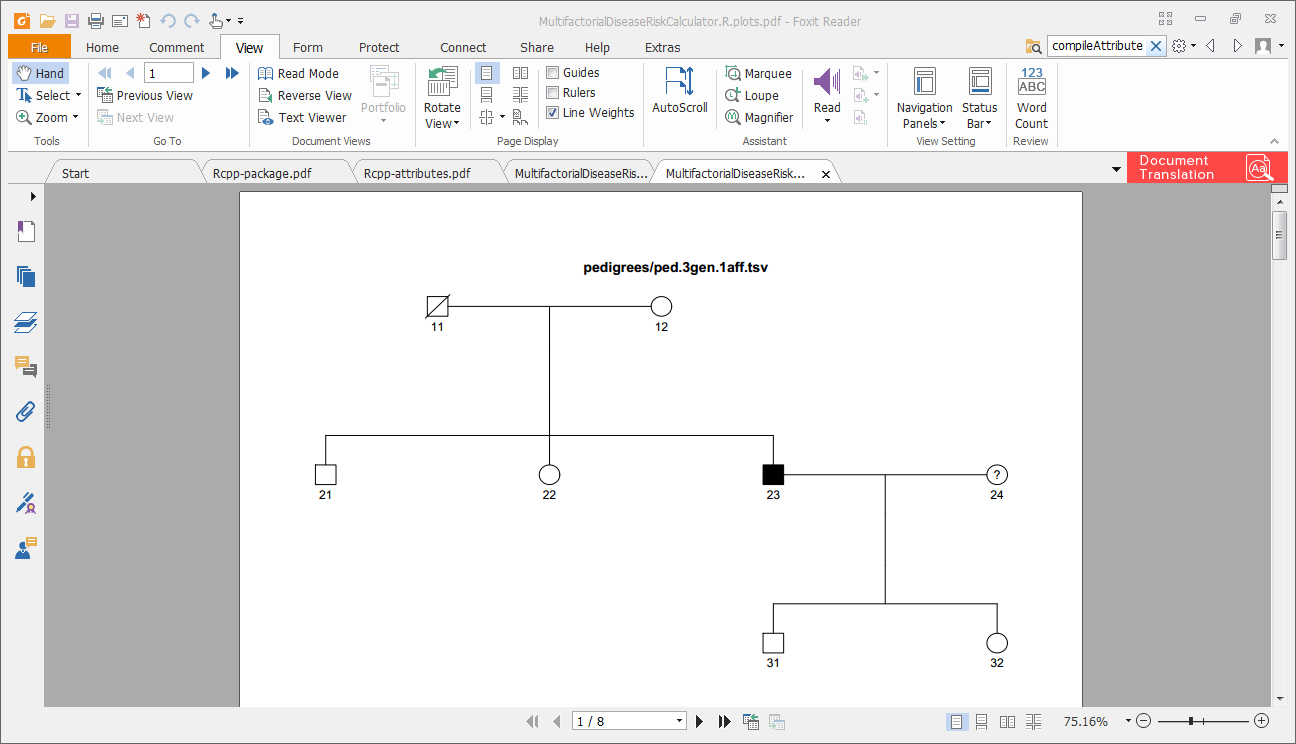
Some other additional columns may be appended depending on the command line options used.

### results.tsv

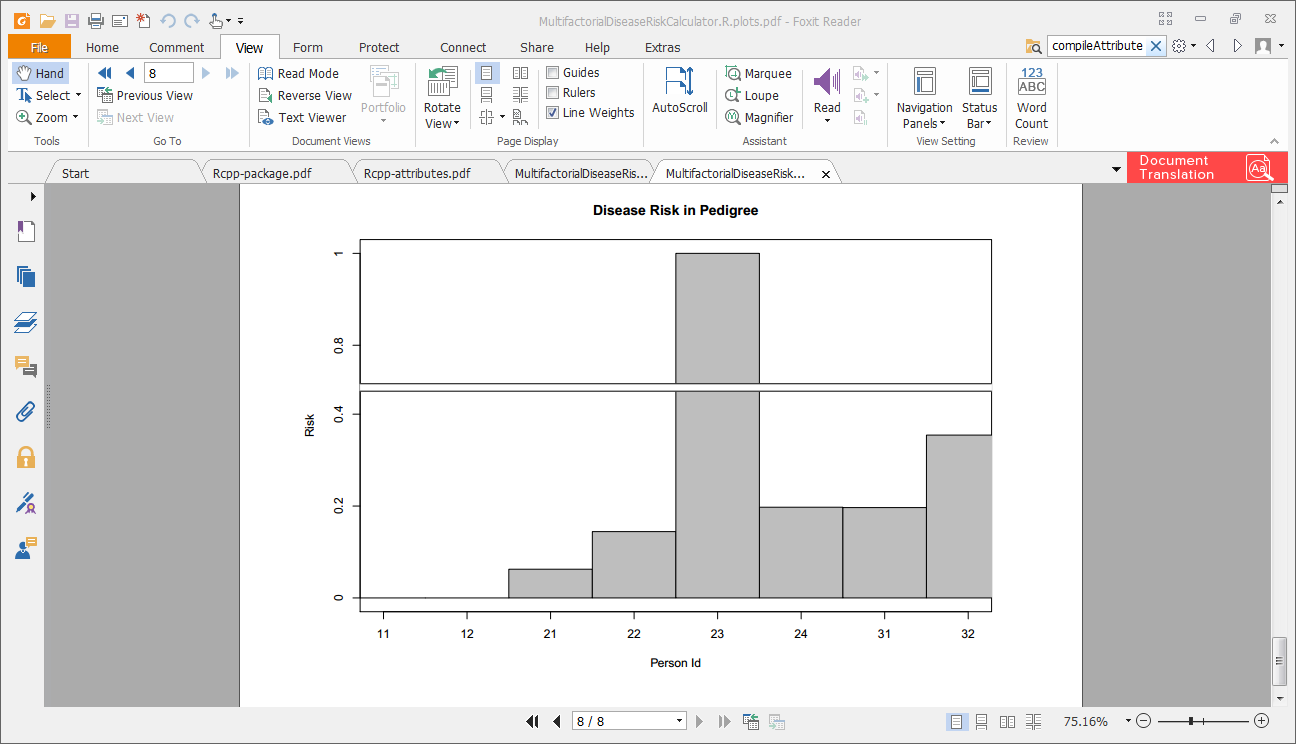
This file contains abbreviated pedigree info along with risk and *n* year risk predictions. The file is in tab separated value format. The columns of this table have been described above.

### MultifactorialDiseaseRiskCalculator.R.plots.pdf

This file contains plots made during the program run. The first and last page are the most interesting.

The first page shows the pedigree diagram. This a pictorial representation of the pedigree specified in the ped file. Here is what that looks like for this example 

The last page contains a (split) bar plot of the predicted disease risks for the pedigree members. Here is what that looks like for this example



In this example, the disease for which risks were predicted was Major Depression (MD). For MD, females have twice the lifetime risk as males. MD is not a congenital disease, instead it manifests throughout adulthood. Both these features are apparent in the risk predictions bar plot.

### pedigreePosteriorLiaSample.tsv

The file pedigreePosteriorLiaSample.tsv contains draws from the pedigree’s posterior joint disease liability distribution, i.e. the liability distribution after conditioning on all disease risk relevant information. It is from this sample that the risk is estimated. An individual’s risk is estimated as the proportion of these liability draws that are above the critical threshold for that individual.

### Prior Variance/Covariance matrix

XXXX - TBD

### Posterior Variance/Covariance matrix

XXXX – TBD

# N Year Risk Prediction Example

N year risk is the risk that a currently unaffected person will become affected within the next *n* years. This is part of the standard output generated by the program. By default *n* is set to 5. The user can change this by providing a --nofYears command line argument.

For instance, enter the following at the command line

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.depression.txt -p pedigrees/ped.3gen.1aff.tsv -y 10

This command differs from the previous example in that it produces predictions of 10 year risk for every unaffected pedigree member (along with each person’s risk). In the previous example (the default) 5 year risk was reported.

The results are reported in the following columns (of results.tsv) appended to the pedigree information

* nofYears - the duration for which the *n* year risk is predicted, i.e. in this case 10
* nYearRisk – the *n* year risk
* f1, f2, a, b – these are parameters used in the calculation of *n* year risk

The method used for predicting *n* year risk is described in

* N Year Risk Prediction.docx

This doc also provides an explanation of the f1, f2, a, b columns.

# Risk Prediction Standard Deviation Example

By default, individual risk is predicted without any estimation of the precision of the prediction being reported. Here we detail how to obtain risk prediction precision estimates. The risk prediction precision is estimated in a very simple way. The risks for the pedigree members are predicted a number of times. Then for each pedigree member, the standard deviation of this sample of their risk predictions is obtained. The user requests risk precision be estimated by setting the –stdDev (-s) option on the command line. By default 10 samples are used to obtain the risk precision estimates. However this can be altered by using the --nofIterations command line argument.

Enter the following at the command line

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.depression.txt -p pedigrees/ped.3gen.1aff.tsv -s -i 20

This will obtain 20 risk predictions for each pedigree member. The mean and standard deviation of these risks will be reported in additional columns appended to the pedigree information

* risk – this is the prime output of interest. It is the predicted disease risk
* riskStdDev – the sample standard deviation on the risk estimates

The pedigreePosteriorLiaSample.tsv file generated will contain only the draws created in the final iteration.

The results.tsv contents (minus some irrelevant columns) are shown below

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **father id** | **mother id** | **sex** | **affected** | **age** | **risk** | **Risk**  **StdDev** | **nofYears** | **nYearRisk** | **nYearRisk**  **StdDev** |
| 11 | 0 | 0 | 1 | 0 | 69 | 0 | 0 | 5 | NA | NA |
| 12 | 0 | 0 | 2 | 0 | 69 | 0 | 0 | 5 | NA | NA |
| 21 | 11 | 12 | 1 | 0 | 34 | 0.062 | 0.0021 | 5 | 0.0170 | 0.00057 |
| 22 | 11 | 12 | 2 | 0 | 34 | 0.141 | 0.0027 | 5 | 0.0385 | 0.00074 |
| 23 | 11 | 12 | 1 | 1 | 36 | 1 | 0 | 5 | 0 | 0 |
| 24 | 0 | 0 | 2 | NA | 34 | 0.199 | 0.0040 | 5 | NA | NA |
| 31 | 23 | 24 | 1 | 0 | 7 | 0.202 | 0.0031 | 5 | 0.0018 | 0.000028 |
| 32 | 23 | 24 | 2 | 0 | 8 | 0.356 | 0.0031 | 5 | 0.0032 | 0.000028 |

As you can see the standard deviation on the risk predictions is low (max a few parts per hundred). The risk standard deviation can be reduced by increasing the number of draws from the pedigree posterior liability distribution used for estimating risk. The --nofDraws command line option controls this. However the risk standard deviation statistic is somewhat misleading. The program predicts risk given the disease model but does not take into account the fact that the disease model is itself generally estimated. Disease model uncertainty is not incorporated into the risk standard deviation.

# XXXX

# XXXX

# Installation

This section describes installation of MultifactorialDiseaseRiskCalculator on a Windows machine. Installation on other platforms may differ in detail, apparently it is straightforward on linux machines.

Download the zip file from the website.

Unzip the zip file.

Move the contained directory and contents to wherever you want. This location will be henceforth referred to as the installation directory.

The code of the program is contained in several files.

* The main program file is
  + MultifactorialDiseaseRiskCalculator.R
* A number of other R files are sourced by the main file, e.g.
  + InitSrc.R, FileReaderSrc.R, DisFileReaderSrc.R, …
* There is also a C++ file
  + rcppFunctions.cpp

These files should all be present in the installation directory.

The program is largely written in R, and was developed under R version 3.1.3 (64bit). A compatible version of R (R version 3.1.3 (64bit) or above) needs to be installed on your computer.

The program requires that a number of R packages are installed. These are

* moments, abind, kinship2, Rcpp, truncnorm, optparse, gplots, shiny, shinysky, shinyAce, Rcpp

XXXX

if (!require(devtools)) install.packages("devtools") #if not already installed

devtools::install\_github("AnalytixWare/ShinySky")

library(shinysky)

The program uses the Rcpp R package to compile the C++ file and link it into the R program. Rcpp uses Rtools to to do this. So Rtools needs to be installed. How to do this is detailed later.

The program can be run from the command line. This is possible because the program’s first line is a hash pling invoking the Rscript program. In order for this to work

* The main file of the program must have execute permissions
* Rscript must be on the path

Another program that must be on the path is g++. This (along with its friends) is needed for compiling and linking the C++ into the R program, which is done via the Rcpp library. Installation of Rtools results in g++ being installed.

## Rtools Installation

This section relates to the installation of Rtools under Windows.

Go to <https://cran.r-project.org/bin/windows/Rtools>

Decide which version you want to install. This will probably be the most recent.

The version I used is - Rtools version 3.2.0.1948, which on the website is called 3.2.

After installation, the version installed can be checked by looking in file

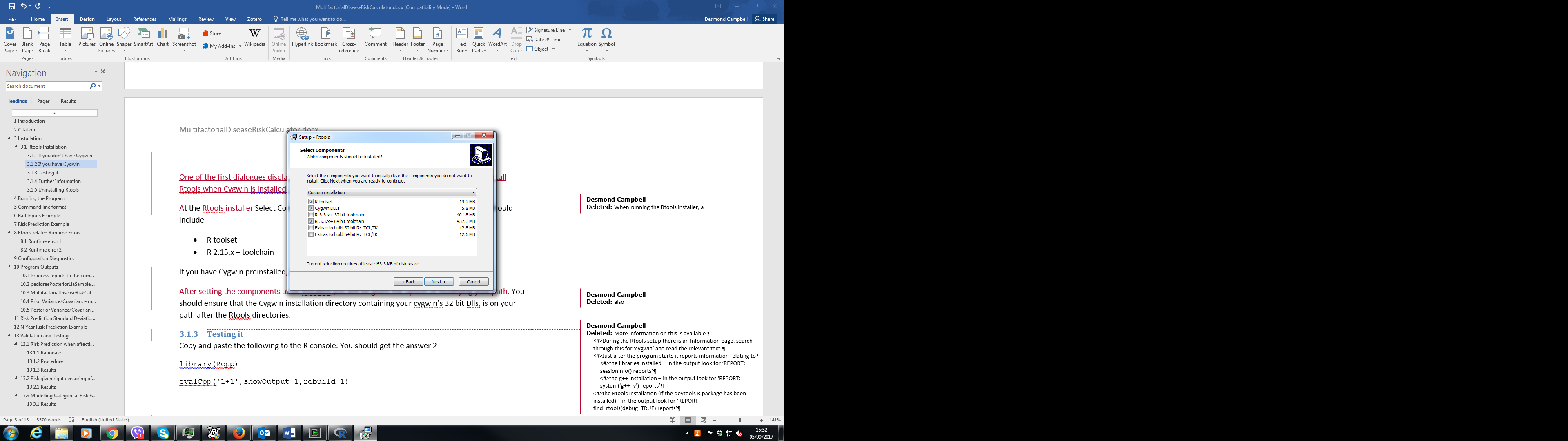
* C:\Rtools\VERSION.txt (presuming you installed Rtools into c:\Rtools)

Download the Rtools installer

### If you don’t have Cygwin

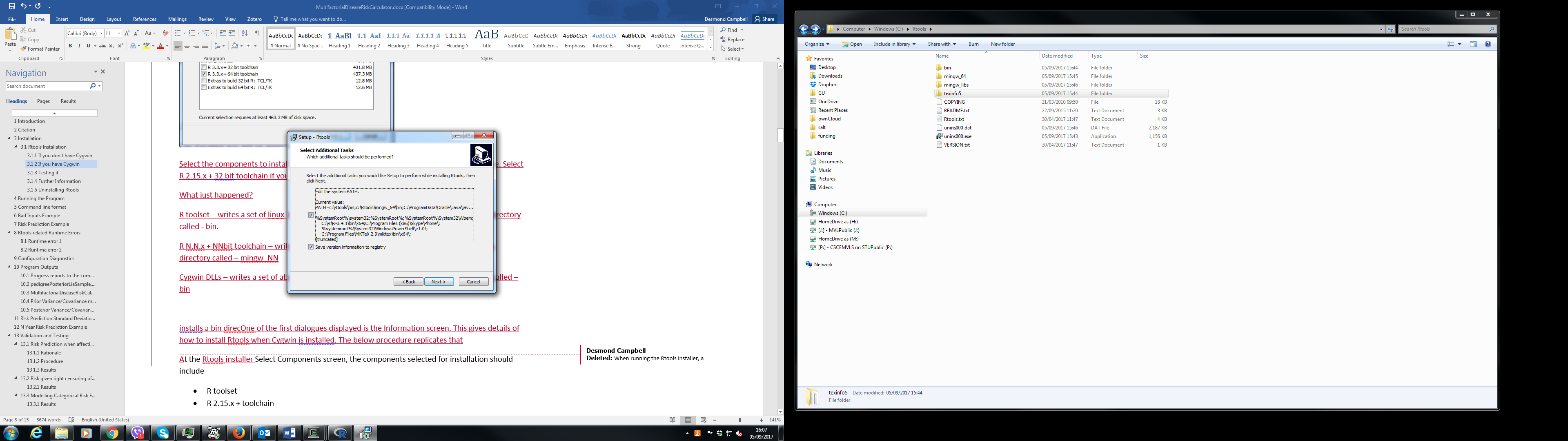
The following installation procedure applies for machine which do not have Cygwin installed. Cygwin is a suite of programs that all you to open a linux like command line prompt and use linux style commands, e.g. cat, grep. If you don’t know what Cygwin is then you probably don’t have it.

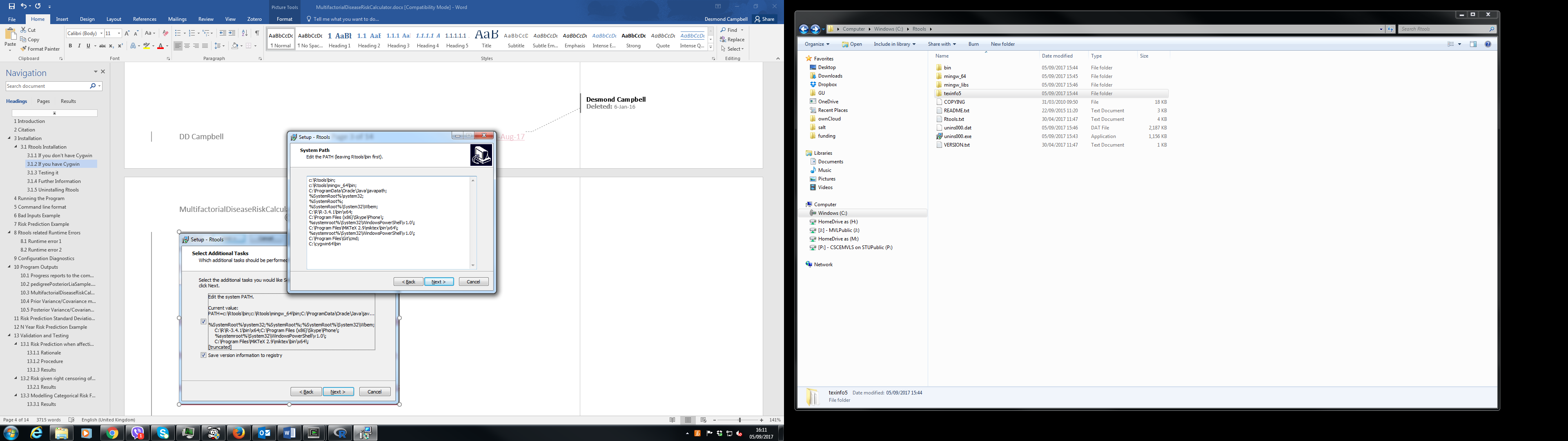
Run the Rtools installer and proceed through the installation procedure until you get to Select Components dialog



Select the components to install as above, if you will only run 64 bit R from the command line. Select R 2.15.x + 32 bit toolchain if you run 32 bit R from the command line. If you are not sure, select both. Press Next.

After selecting components, the installer provides the option of altering the PATH so that R will know where to find Rtools and the g++ compiler automatically. It also allows registration of Rtools so it can be uninstalled easily. I suggest you take advantage of this. Set the checkboxes as below. Press Next.



If you selected the edit system PATH option, you will get something like the following 

The Rtool sub-directories have been prepended to the PATH. Press Next.

The Ready to Install dialog appears summarizing the installation you are about to do. Press Next.

The next dialog reports the Rtools installation progress and completion.

What just happened? The installation created your Rtools installation directory, populating it with a few maintenance files. In addition, the following happened depending on which components you selected for installation

* R toolset –a set of unix like programs (e.g. cat, grep) were written into an Rtools installation sub-directory called - bin.
* R N.N.x + NNbit toolchain – the g++ compiler and linker etc. were written into an Rtools installation sub-directory called – mingw\_NN
* Cygwin DLLs –a set of about 10 cygwin DLLs were written into an Rtools installation sub-directory called – bin

### Testing it

#### Rgui

Open an Rgui console.

Enter the following at the R console prompt.

library(devtools)

find\_rtools()

You should get the answer TRUE

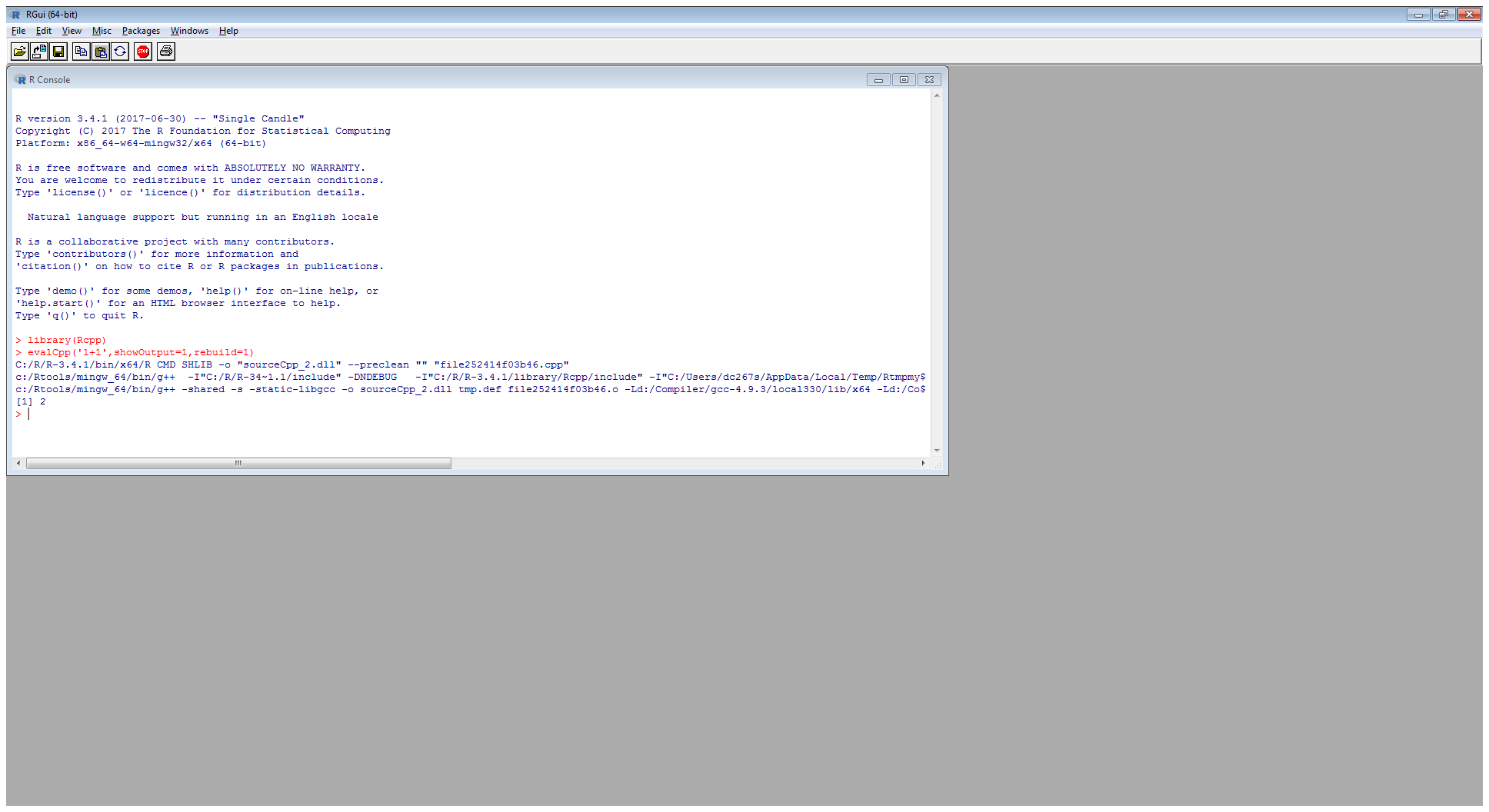
Enter the following at the R console prompt.

library(Rcpp)

evalCpp('1+1',showOutput=1,rebuild=1)

After about a minute, you should get a few lines of output with the last line giving the answer 2

Below is a screenshot of this from my computer.

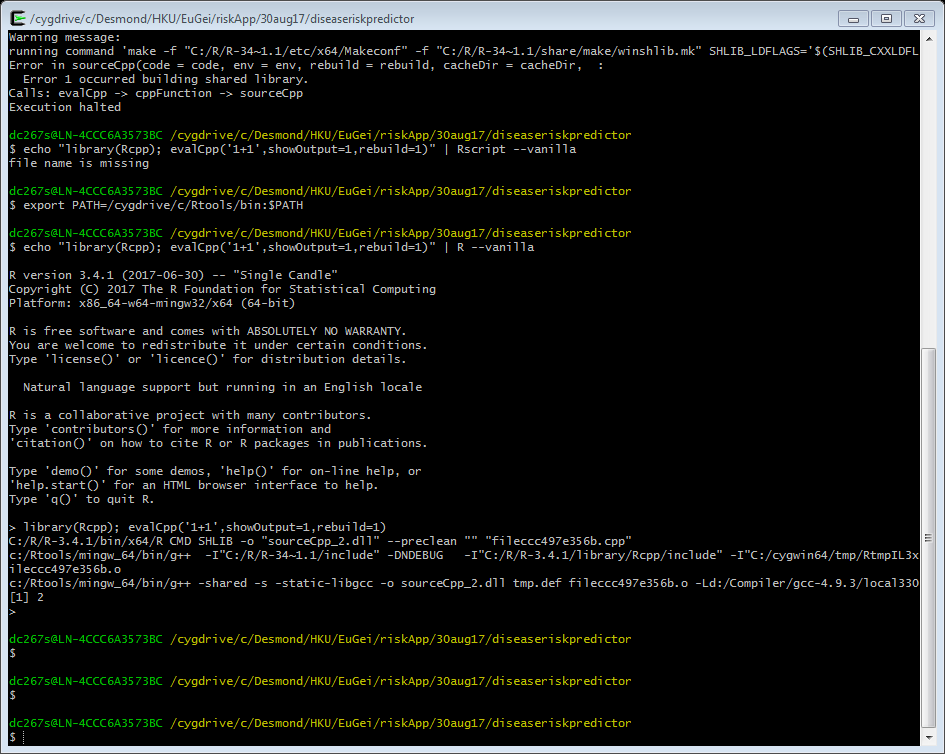


#### Command Line R

Open a command prompt

Enter the following at the command prompt.

echo "library(Rcpp); evalCpp('1+1',showOutput=1,rebuild=1)" | R --vanilla

After about a minute, you should get a few lines of output, with the last line giving the answer 2. See below 

### Further Information

More information on Rtools installation is available

* During the Rtools setup there is an Information dialog, search through this for ‘cygwin’ and read the relevant text.
* Just after the Risk Prediction program starts it reports information relating to
  + the libraries installed – in the output look for ‘REPORT: sessionInfo() reports’
  + the g++ installation – in the output look for ‘REPORT: system('g++ -v') reports’
  + the Rtools installation (if the devtools R package has been installed) – in the output look for ‘REPORT: find\_rtools(debug=TRUE) reports’
* Further details is available in the Rtools and Rcpp online documentation.

### Uninstalling Rtools

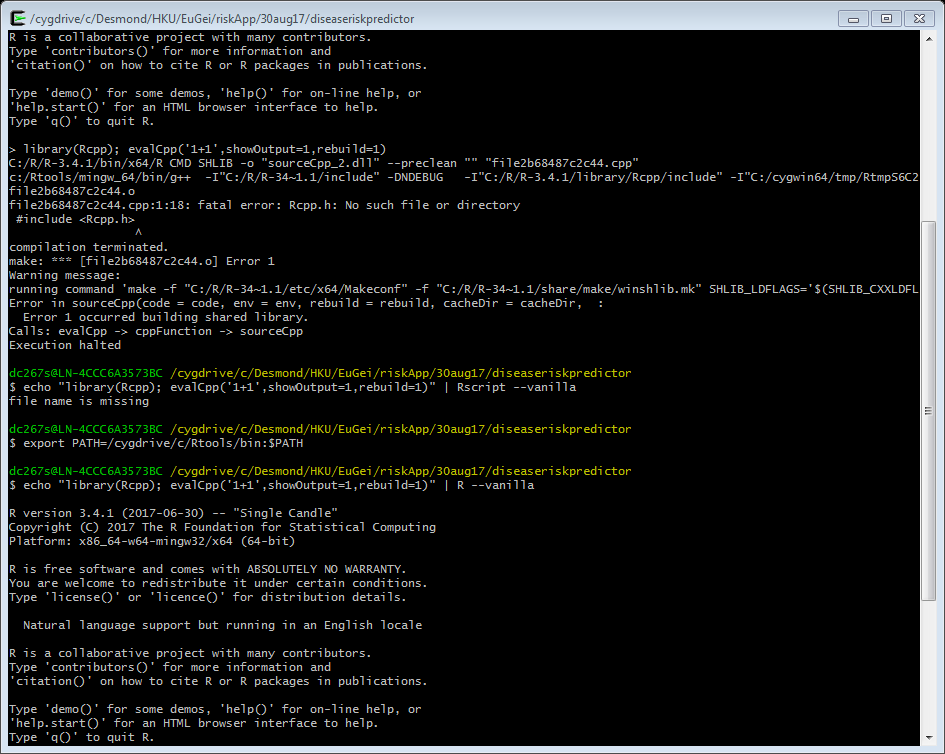
If the Rtools version was saved to the registry (check box on one of the installation dialogs), then it can be uninstalled via Start > Control Panel > Programs > Uninstall a Program

Otherwise go to the Rtools installation directory and run unins000.exe to uninstall it.

### If you have Cygwin

If you have Cygwin installed, then Rtools installation is somewhat more complicated.

If you have Cygwin installed, Rtools recommend not installing the Cygwin Dlls component (deselecting it in the Select Components dialog). Their intention is that the already installed Cygwin Dlls get used. When I followed their recommendation I found upon testing the Rtools installation that it worked (in the R GUI) but failed at the cygwin prompt, see below



The workaround I found for this is as follows

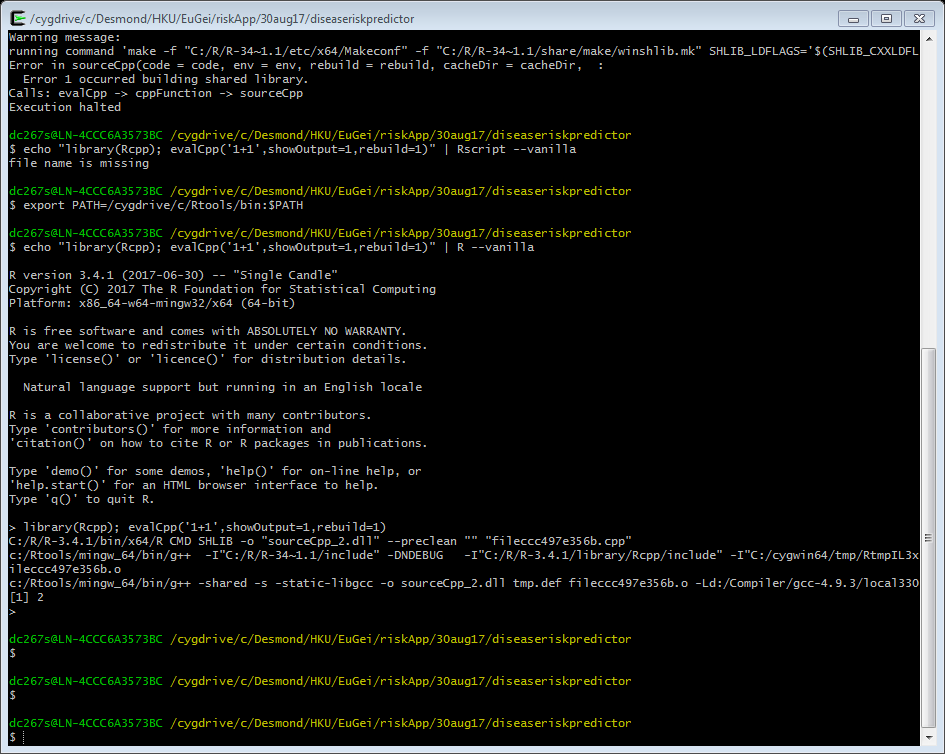
Check Cygwin Dlls in the Select Components dialog during the installation. This will cause Cygwin Dlls to be installed in the Rtools installation subdirectory – bin. This will not interfere with normal Cygwin working because when Cygwin is invoked, the PATH variable is prepended with the path to Cygwin utilities and commands. Therefore when running Cygwin, the Rtools versions of Cygwin utilities will not be invoked.

Open a Cygwin command prompt

Enter the following at the Cygwin prompt

export PATH=/cygdrive/c/Rtools/bin:$PATH

This puts the Rtools Cygwin path in front of cygwin’s own. This means the Rtools versions of Cygwin utilities will be invoked in preference to Cygwin’s own. This could break future use of that Cygwin prompt but it seems to be ok. When I test the Rtools installation at the prompt it works, see below.



For a previous version of R and Cygwin, I followed the Rtools recommendation and everything worked.

# Command line format

The program takes named (not positional) arguments. Arguments can be in long or short form. Some arguments take values, others don’t. The latter are switches, i.e. they provide yes/no information indicated by the argument’s presence or absence. The appropriate format is shown in the previous section. Example of equivalent long and the short form of a valued argument

* --diseaseModel=diseaseModels/dm.Con1.txt
* -d diseaseModels/dm.Con1.txt

Example of equivalent long and the short form of a switch argument

* --stdDev
* -s

If the command line input is not in the appropriate format then the program will abort returning an error message.

# Bad Inputs Example

It may be that the command line is valid but either the disease model file or pedigree file specified is invalid. In such cases an appropriate error message will be reported.

For instance, enter the following command

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.Con1.txt -p pedigrees/ped.p2c3.deceasedNA.tsv

In this case the final report from the program is

Error in fnValidatePedigree(dfPed) :

Problem with pedigree column 'deceased': NA is an invalid value

Execution halted

If you check the pedigree you will see that one of the pedigree member’s deceased status is unknown. For pedigree diagram drawing purposes, unknown deceased status is not allowed. The program detects this problem and halts reporting the problem.

# Rtools related Runtime Errors

The program may start happily but later on hang or report Cygwin Dll related problems. If so something is wrong with your Rtools installation (see above).

Two such errors and what to do about them are described below.

## Runtime error 1

Errors similar to the following are likely due to Cygwin being installed via Rtools and elsewhere.

2 [main] sh (7160) C:\cygwin\bin\sh.exe: \*\*\* fatal error - cygheap base mismatch detected - 0x612B7408/0x612C8408.

This problem is probably due to using incompatible versions of the cygwin DLL.

Search for cygwin1.dll using the Windows Start->Find/Search facility

and delete all but the most recent version. The most recent version \*should\*

reside in x:\cygwin\bin, where 'x' is the drive on which you have

installed the cygwin distribution. Rebooting is also suggested if you

are unable to find another cygwin DLL.

I fixed this by removing all Cygwin dlls (they match cyg\*.dll) from C:\Rtools\bin

## Runtime error 2

Errors similar to the following probably result from there being no Cygwin dlls on your path.

Error 1 occurred building shared library.

The solution is to add the Cygwin 32 bit installation directory containing Cygwin dlls to your path. The Cygwin 32 bit installation directory must be after the R tools directories on the path.

Note, my path originally included the bin directory of a 64 bit version of Cygwin. The program did not work with this path. It started working when I dropped that directory from my path and replaced with the bin directory of a 32 bit version of Cygwin.

The following command will tell you if you're running the 32 bit or 64 bit version:

uname -m

"i686" for the 32-bit version, "x86\_64" if it's 64-bit.

# Configuration Diagnostics

Installing this program with the appropriate libraries etc. is quite difficult. To aid in this, the program reports the session info and various other info immediately after being invoked. Look for the lines following (close to the start of the program reporting)

REPORT: Below is some diagnostic information regarding the setup

REPORT: sessionInfo() reports

For my installation the session info reported is

REPORT: sessionInfo() reports

R version 3.1.3 (2015-03-09)

Platform: x86\_64-w64-mingw32/x64 (64-bit)

Running under: Windows 7 x64 (build 7601) Service Pack 1

locale:

[1] LC\_COLLATE=English\_United Kingdom.1252 LC\_CTYPE=English\_United Kingdom.1252 LC\_MONETARY=English\_United Kingdom.1252 LC\_NUMERIC=C

[5] LC\_TIME=English\_United Kingdom.1252

attached base packages:

[1] stats graphics grDevices utils datasets base

other attached packages:

[1] optparse\_1.3.2 truncnorm\_1.0-7 Rcpp\_0.12.1 kinship2\_1.6.4 quadprog\_1.5-5 Matrix\_1.2-2 abind\_1.4-3 gplots\_2.17.0 moments\_0.14

loaded via a namespace (and not attached):

[1] bitops\_1.0-6 caTools\_1.17.1 gdata\_2.17.0 getopt\_1.20.0 grid\_3.1.3 gtools\_3.4.2 KernSmooth\_2.23-15 lattice\_0.20-33 methods\_3.1.3

# Program Outputs

If the program is running correctly, the outputs produced by the program are the following

* Progress reports to the command line
* pedigreePosteriorLiaSample.tsv
* MultifactorialDiseaseRiskCalculator.R.plots.pdf
* results.tsv

## Progress reports to the command line

The program reports its progress as it runs. It first reports the arguments it has extracted from the command line. It reads the pedigree information from file and reports this to screen. It does the same for the disease model. After the risks have been estimated, the pedigree information is output again but with additional columns appended. These are

* lifetimeRisk – the disease lifetime risk (taken from the disease model)
* thr – the quantile of a standard normal distribution for the lifetime risk
* expressedProportionOfLifetimeRisk – the expressed proportion of lifetime risk is a personal attribute. It expresses the degree of right censoring of an unaffected affection status.
* risk – this is the prime output of interest. It is the predicted disease risk

Some other additional columns may be appended depending on the command line options used.

## pedigreePosteriorLiaSample.tsv

The file pedigreePosteriorLiaSample.tsv contains draws from the pedigree’s joint disease liability distribution after that distribution has been conditioned on all disease risk relevant information. It is from this sample that the risk is estimated. An individual’s risk is estimated as the proportion of these liability draws that are above the critical threshold for that individual.

## MultifactorialDiseaseRiskCalculator.R.plots.pdf

This file contains plots made during the program run

## Prior Variance/Covariance matrix

XXXX - TBD

## Posterior Variance/Covariance matrix

XXXX - TBD

# Risk Prediction Standard Deviation Example

By default, individual risk is predicted without any estimation of the precision of the prediction being reported. Here we detail how to obtain risk prediction precision estimates. The risk prediction precision is estimated in a very simple way. The risks for the pedigree members are estimated a number of times. Then the standard deviation of this sample of each pedigree members risk estimates is obtained. The user requests risk precision be estimated by setting the –stdDev (-s) option on the command line. By default 10 samples are used to obtain the risk precision estimates. However this can be altered by using the --nofIterations command line argument.

Enter the following at the command line

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.Con1.txt -p pedigrees/ped.p2c3.affectedMz.tsv -s -i 20

This will obtain 20 risk estimates for each pedigree member. The mean and standard deviation of these risks will be reported in additional columns appended to the pedigree information

* risk – this is the prime output of interest. It is the predicted disease risk
* riskStdDev – the sample standard deviation on the risk estimates

The pedigreePosteriorLiaSample.tsv file generated will contain only the draws created in the final iteration.

The results.tsv contents (minus some irrelevant columns) are shown below

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **famid** | **id** | **Father**  **id** | **Mother**  **id** | **sex** | **affected** | **age** | **lifetime Risk** | **thr** | **expressed**  **Proportion**  **Of Lifetime Risk** | **risk** | **riskStdDev** |
| 1 | 11 | 0 | 0 | 1 | NA | 49 | 0.05 | 1.6449 | 1 | 0.142 | 0.002 |
| 1 | 12 | 0 | 0 | 2 | NA | 39 | 0.05 | 1.6449 | 1 | 0.142 | 0.002 |
| 1 | 13 | 11 | 12 | 1 | 1 | 20 | 0.05 | 1.6449 | 1 | 1.000 | 0.000 |
| 1 | 14 | 11 | 12 | 2 | NA | 12 | 0.05 | 1.6449 | 1 | 0.143 | 0.003 |
| 1 | 15 | 11 | 12 | 1 | NA | 20 | 0.05 | 1.6449 | 1 | 0.309 | 0.003 |

As you can see the standard deviation on the risk estimates is very low. That is somewhat misleading, the program calculates risk given the disease model. However generally the disease model is itself an estimate. Disease model uncertainty is not incorporated into the risk standard deviation.

# N Year Risk Prediction Example

The n year risk for a currently unaffected person is the risk they will become affected within the next n years. The user can request this to be predicted via the --nofYears command line argument.

Enter the following at the command line

./MultifactorialDiseaseRiskCalculator.R -d ./validationNYearRisk/dm.schizophrenia.txt -p ./validationNYearRisk/testPedDes.tsv -y 10

This command causes the 10 year risk to be predicted for every unaffected person, along with each person’s risk. The results are reported in the following columns appended to the pedigree information

* nofYears - the duration for which the n year risk is predicted, i.e. in this case 10
* nYearRisk – the n year risk
* f1, f2, a, b – related columns

The method used for calculating the n year risk is described in

* N Year Risk.docx

This doc also provides an explanation of the f1, f2, a, b columns.

# Validation and Testing

Extensive simulation studies were conducted to characterise the performance of the methodology (Campbell et al. 2010). To validate this program much more rudimentary testing was performed. Several aspects were tested

* Risk Prediction when affection status is unknown (i.e. when right censoring of affection status can be ignored)
* Risk Prediction given right censoring of affection status

## Risk Prediction when affection status is unknown

When affection status is unknown the effects of right censoring of affection status can be ignored.

The following pedigree was used

* ./pedigrees/ped.p2c3.affectedMz.tsv

In this pedigree there is one affected individual, the affection status for the rest of the pedigree is unknown.

Pedigree members’ risk was predicted via two methods

* the program
* Pearson Aitken Selection formulae method (described below)

The risk predictions obtained by the two methods were then compared for each pedigree member.

### Rationale

For a set of correlated variables, conditioning on a subset of the variables will induce distributional changes in the remaining variables. If the variables jointly follow a multivariate normal distribution, then the Pearson Aitken selection formulae relate summary statistics (mean vector and variance/covariance matrix) regarding

* the full multivariate distribution, prior to conditioning
* a subset of conditioning variables, post conditioning
* the remaining (i.e. conditioned upon) variables, post conditioning

The pedigree joint liability distribution (in terms of a mean vector and a variance/covariance matrix) can be derived given disease model information on how liability variance is partitioned across genetic and environmental components.

The liability of an affected person is distributed as a truncated standard normal with all their liability lying above the critical threshold determined from the disease’s lifetime risk. The mean and variance of the truncated normal can be calculated.

We can use the Pearson Aitken selection formulae to condition the pedigree’s prior joint liability distribution on the affection status information. The result is summary statistics (mean vector and covariance matrix) for the pedigree’s posterior joint liability distribution.

Individual risk can then be estimated. This is done by applying the disease’s critical threshold to a Gaussian distribution with the person’s posterior mean and variance. Risk is the proportion of the distribution above the disease’s critical threshold.

This procedure can be used to provide approximate estimates of disease risk.

### Procedure

The program was used to predict risk for pedigree members for schizophrenia

Enter the following command

./MultifactorialDiseaseRiskCalculator.R -d diseaseModels/dm.schizophrenia.txt -p pedigrees/ped.p2c3.affectedMz.tsv

We calculate the mean and variance of the affected person’s liability (truncated normal) distribution.

Then use the Pearson-Aitken selection formulae to calculate the mean and variance of the liability distribution of several other pedigree members conditional on the affected person’s liability mean and variance.

We then calculate risk for those persons by applying the disease’s critical threshold to a Gaussian distribution of the appropriate mean and variance.

The person’s we do this for are

* The affected person’s MZ twin (correlated by A and C)
* The affected person’s other sibling (correlated by 0.5A and C)
* The affected person’s parent (correlated by 0.5A)

An R script was used for doing this. It is

* validateRiskPredForUnknownAffectionStatus.R

### Results

Risk as estimate via the above method was

> # affected's MZ twin

$postRiskB

[1] 0.6021358

> # affected's sib

$postRiskB

[1] 0.1396804

> # affected's parent

$postRiskB

[1] 0.08834543

This matches very well risks estimated via the program.

REPORT: Estimated Risks for the pedigree members are given below

famid id fatherid motherid sex affected ... relation.MZ ... lifetimeRisk ... risk

1 1 11 0 0 1 NA ... 0 ... 0.01 ... 0.08715

2 1 12 0 0 2 NA ... 0 ... 0.01 ... 0.09330

3 1 13 11 12 1 1 ... 1 ... 0.01 ... 1.00000

4 1 14 11 12 2 NA ... 0 ... 0.01 ... 0.13900

5 1 15 11 12 1 NA ... 1 ... 0.01 ... 0.59475

In conclusion, there is no evidence the program’s basic risk estimation implementation is misbehaving.

## Risk given right censoring of affection status

TBD

We will estimate disease risk in the following pedigree - XXXX

For the disease specified in – XXXX

In this pedigree there is one unaffected individual, the affection status for the rest of the pedigree is unknown.

The affection status of the individual is right censored by age.

We run the program via entering the command

XXXX

The expressed proportion of lifetime risk for the unaffected person is a function of their age and the disease model is output by the program.

Whether their expressed proportion of lifetime risk is correct can be determined by manual inspection of the age of onset curve in the disease model and their personal attributes (age).

Their risk of disease is the proportion of their liability distribution that is above threshold. Their liability distribution is a mixture distribution of two truncated normal distributions.

However it is easier to think in terms of part of their liability distribution being explained by their unaffected affection status.

The proportion explained is

where

* = lifetime risk
* = expressed proportion of lifetime risk

That leaves as the proportion unexplained

The above threshold part unexplained is

The disease risk is the proportion of all unexplained liability that is above threshold, i.e.

The following code snippet can be used to calculate risk

XXXX

# lifetime risk

r <- 0.1

# expressed proportion of lifetime risk

e <- 0.2

# calc risk for an unaffected

risk <- (r\*(1-e))/(1-r\*e)

risk

### Results

Risk for unaffected individuals was calculated via the above and also estimated via the program.

The risks obtained by the two methods matched over a range of ages.

In conclusion there is no evidence the program’s is improperly accounting for right censoring of affection status.

## Modelling Categorical Risk Factors

The disease is modelled by a liability threshold model in which the population liability distribution is a mixture distribution of risk factor strata liability distributions. Liability distribution within each risk factor stratum is assumed to be Gaussian. The means of these liability distributions are allowed to differ across risk factor strata but the variances are constrained to be equal. The population distribution, a mixture distribution of the risk factor strata distributions, is specified to have a mean of zero and a variance of 1. These constraints allow the risk strata liability means (and thence the variance they share) to be calculated from the information provided in the disease model, namely

* Risk factor categories – frequencies (must sum to 1)
* Risk factor categories – relative risks (relative to some arbitrarily chosen baseline category)
* Disease lifetime risk

Doing this is not trivial. It involves solving a set of non-linear simultaneous equations. This was done via an optimisation algorithm. Details of how categorical risk factors are handled are to be found in

* Categorical Risk Factors.docx

This document also describes how the methodology and its implementation were validated.

### Results

The solution we developed works well over a wide range of possible categorical risk factors.

# References

Campbell, Desmond D., Pak C. Sham, Jo Knight, Harvey Wickham, and Sabine Landau. 2010. ‘Software for Generating Liability Distributions for Pedigrees Conditional on Their Observed Disease States and Covariates’. *Genetic Epidemiology* 34 (2): 159–70. doi:10.1002/gepi.20446.

XXXX (Campbell et al. 2017) – Gen Epi