clippda: A package for clinical proteomic profiling data analysis

Stephen Nyangoma

May 3, 2016

Cancer Research UK Clinical Trials Unit, Institute for Cancer Studies, School of Cancer Sciences, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

Contents

1 Overview

This package is still under development but it is intended to provide a range of tools for analysing clinical genomics, methylation, and proteomics data. The method used is suitable for analysing data from single-channel microarray experiments, and mass spectrometry data (especially in the situation where there is not one-to-one correspondence between the cases and controls), with non-standard repeated expression measurements arising from technical replicates. Its main application will be to mass spectrometry data analysis. Mass spectrometry approaches, such as the Matrix-Assisted Laser Desorption/Ionization (MALDI) and Surface-Enhanced Laser Desorption and Ionisation Time-of-Flight (SELDI-TOF), are increasingly being used to search for biomarkers of potentially curable early-stage cancer.

Currently, I am more concerned with the problem of finding the number of biological samples required for designing clinical proteomic profiling studies to ensure adequate power for differential peak detection. But we have included a number of tools for the pre-processing of repeated peaks data, including tools for checking the number of replicates for each sample, the consistency of the peaks between replicate spectra, and for data formatting and "averaging". In the future, *clippda* package will offer additional tools, including: functions for differential-expression analysis of peaks from studies with technically replicated assays, and for pre-processing the 3-dimensional, Liquid Chromatography (LC) - Mass Spectrometry (MS) datasets.

To be able to detect clinically-relevant differences between cases and controls with adequate statistical power, a given number of biological samples must be analysed. While mass spectrometry technology is imperfect, and is affected

by multiple sources of variation, little consideration has been given to sample size requirements for studies using this technology. Sample size calculations are typically based on the distribution of the measurement of interest, the between-subject (biological) variability, the clinically meaningful size of the difference in expression values between the cases and controls, the power required to detect this difference, and the p-value (i.e. false positive rate). In proteomic profiling studies, sample size determination needs to consider several additional factors which make sample size calculations less straightforward than sample size determination for many other studies. We need to account for experimental error, and make adjustments for the number of experimental replicates. In addition, proteomic profiling studies seek to simultaneously detect the association of multiple peaks with a given outcome; thus, adjustments, which control for the type I error rate, must be made to the sample size. However, the problem of controlling for the number of false positives is complicated in a proteomic profiling study. The number of peaks (and of differentially-expressed proteins) is normally much lower, often between 45 and 150, than the number of genes considered in microarray studies (typically, 1000's). Thus the parameters, such as the significance level, the power and the proportion of true positives, which control the FDR, must be specially defined for proteomic profiling studies: parameters appropriate to microarray studies cannot be transferred dirrectly to proteomic profiling studies. It is common to find that a significance cut-off of $\alpha = 0.05$ is used. For proteomics datasets this level of significance has been shown to adequately control for the tail probability of the proportion of false positives, (TPPFP) (**?**). TPPFP is a resampling-based empirical Bayes method which controls the ratio of false positives to total rejections. It does not require the use of a fixed value for the clinically important difference for all peaks (as required in microarrays; see, e.g. **?**), and is suitable for comparative analyses where these differences may vary across chip-types, for example.

Our applications focus on the clinical proteomics of cancer. Most of the gene expression and proteomics analyses involving human specimens are observational case-control by design, and this fact immediately raises the key issue of possible biases and confounding factors in the populations from which the samples are drawn (**?**). Moreover, most clinical studies are unbalanced in that the number of biological samples having specific phenotypic attributes differs between the cases and controls. Sample imbalance has been found to have a huge impact on sample size requirements in microarray studies (**?**) but has not been a subject of study in proteomic profiling studies. Unfortunately, in the majority of clinical genomic and proteomic profiling studies, the distinction between observational and experimental designs is not made, and the methods for analyzing experimental (randomized) clinical studies are used, arbitrarily. This software implements a method (**?**) for calculating the sample size required for planning proteomic profiling studies which adjusts for the effect of sample imbalances and confounding factors. Perhaps the main challenge in constructing a sample size formula for planning proteomic profiling studies is the fact that confounders must be appropriately adjusted for. However, typically no confounder information is available at the design stage of a study. In this case, two options are

available:
1. Ignore the confounder information and use the sample size calculation formula such as that adopted by **?**. However, this underestimates the sample size required, which may have an adverse effect on the quality of the conclusions drawn from the study.
2. Introduce a method which makes it simple to include adjustments for expected heterogeneities in sample size calculations.
We use a sample size formula which is based on a modified version of the Generalize Estimating Equations (GEE) method (**?**). It takes into account the experimental design of repetitions of peak measurements, and it adequately adjusts for all sources of variability, heterogeneity and imbalances in the data. In this method, it is assumed that the joint distribution of the peak data and the covariates is known. Thus the usual Fisher information matrix is replaced by the expected (with respect to covariates) Fisher information matrix and the covariate information enters the formula as a function of the multinomial proportions of subjects with specific attributes, while the repeated peak information enters as a linear function of the intraclass correlations between the intensities of the replicates. This method makes it straightfoward for a clinician to plan what proportion of samples with given attributes to include in an experiment. Typically, the covariate information need not be available for the confounding effects to be specified in a sample size calculation. For any proteomic profiling study, the effect of covariates on sample size calculation may be simulated from an appropriate univariate normal distribution with mean $Z = 2 + c$, where $0 \leq c \leq 1$. In these simulations, values of $Z \approx 2$ (and a small variance ≈ 0.03, indicate that the study is balanced. Departures from these values represents various degrees of imbalance. As far as we know, there is only one reference on the sample size requirements for proteomic profiling studies: **?**; but there is an extensive literature for sample size in microarray studies (e.g. **????**).
Defining guidelines for sample size requirements in proteomic profiling studies is complex, since there are multiple ways of setting up experiments: for example, there are many SELDI chip-types (e.g. IMAC, MC10, H50, and Q10), several biofluids (e.g. serum, urine, and plasma), a number of possible control objects (e.g. healthy or diseased controls), and various sample-types (e.g. cancers which are localalized in different body organs). This results in a difficult *multifactorial*, problem and the use of pilot data is warranted.
This method is implemented in the *clippda* package, which offers a number of functions for data pre-processing, formatting, inference, and sample size calculations for unbalanced SELDI or MALDI proteomic data with technical replicates. As opposed to microarrays, it requires multiple parameters, including the intraclass correlations, protein variance, and the clinically important differences. The intraclass correlation is usually fixed in many applications involving repeated measurements, and we follow this approach. We avoid defining the clinically important differences as fold-changes. We rely on a well-defined interval to determine what is clinically important. The sample sizes required for planning clinical proteomic profiling studies are the elements of the plane described by the clinically important differences versus protein variances. There is an option

for plotting sample size parameters from past experiments on the grid described by the the clinically important differences versus protein variances, onto which sample size contours have been superimposed. This information is crucial for establishing general guidelines for sample size calculations in proteomic profiling studies as it gives the bounds for sample size requirements. Thus the grid may be used as tool for integrating sample size information from disparate sources. The control of biological variance is found to be critical in determining sample size. In particular, the parameters used in our calculations are derived from summary statistics of peaks with "medium" biological variation.

Even though, the method for specifying the confounder effect (**?**) is particularly important at the design stage of a clinical proteomics study, it may also be employed when one wants to reuse the data from a previous study (in which there is no complete record of covariates) as the "pilot" data for a new study. The data from previous studies can be used to determine the range for possible values for the intraclass correlation, the within and between sample variances, and the size of differences to be estimated. Then the method given in Nyangoma *et al.* (2009), may be used to to specify the range for possible values for the confounding effects, which can be used in sample size calculations to set guidelines for the required sample size when designing a proteomic profiling study.

From a clinical proteomic profiling study we have protein expression measurements, and separate records of phenotypic characteristics of the samples under study. However, these datasets are stored in formats which cannot be used directly in statistical analyses. Moreover, there is often missing phenotypic information for certain individuals. Thus data management and storage become key issues in proteomic profiling data analyses.

We define a new object class for clinical proteomic profiling data, known as `aclinicalProteomicData`, which can be used to combine both expression data and phenotypic information from a clinical proteomic study into a single coherent and well-documented data structure which can be used directly in analyses. An `aclinicalProteomicData` class `object` has `slots` for storing: "raw" SELDI data, covariates, phenotypic data, classes for phenotypic variables, and the number of peaks of interest (e.g. peaks detected by Biomarkers wizard software).

It should be noted that there are two packages in *Bioconductor* for handling `SELDI` datasets. The *PROcess* package (**?**), provides functions for pre-processing raw `SELDI` mass spectra. *caMassClass* (**?**), provides tools for pre-processing, and classification of `SELDI` datasets. Thus, the mass spectrometry datasets pre-processed using these packages could be analysed using the *clippda* package, to provide information on differential protein expression and the sample size required to plan a proteomic profiling study.

2 Getting started

Installing the package. There are instructions for installing packages at `http://www.bioconductor.org/docs/install/`. For Windows users, first download the appropriate file for your platform (e.g. a .zip file) from the Bioconductor

website `http://www.bioconductor.org/`. Next start R and select the `Packages` menu. Next, select `Install package from local zip file...`. Find and highlight the location of the zip file and click on `open`. Alternatively, you can download the package from your nearest `CRAN` mirror site. Simply set the `CRAN` mirror appropriately, and then use the command `install.packages` from within an R session.

Loading the package. To load the *clippda* package in your R session, type `library(clippda)`.

Help files. Detailed information on *clippda* package functions can be obtained in the help files. For example, to view the help file for the function `clippda` in a browser, use `help.start` followed by `?clippda`.

Case study. We provide data from sera samples from liver patients.

Sweave. This document was generated using the `Sweave` function from the R *tools* package. The source (.Rnw) file is in the `/inst/doc` directory of the *clippda* package.

3 Software Application: proteomic profiling of liver serum

3.1 Introduction

The package for calculating the sample size required for planning a proteomic profiling study is called *clippda*. In it, there are a number of user level functions for pre-processing proteomic data with repeated peak measurements to put them in a format amenable to analysis using conventional tools for repeated measures analysis. The function for sample size calculations is called `sampleSize`. There are plots for visualizing the results of your calculations: the `sampleSizeContourPlots` draws a grid for calculating sample size using parameter values that we have found to be clinically-relevant, while `sampleSize3DscatterPlots` displays the sample sizes corresponding to the range of the clinically important parameters.
We begin by loading the necessary package

```
> library(clippda)
```

We use the `install.packages`, or the function, `bioLite`, to get the necessary analysis and data packages from the R and Bioconductor repositories.

3.2 The *clippda* package and initial data filtering

The dataset used for illustration is from the proteomic profiling of liver cancer patients vs non-cancer controls. There were 60 sera samples from patients with

liver tumors, and 69 from non-cancer controls. In addition to the information on tumor class (exposure) of the samples, gender information was available. The samples were divided into two aliquots, pre-processed using the SELDI protocol, and then assayed on IMAC chips. Although all the samples (except the QC, or quality control, samples) were meant to be assayed in duplicate, there were cases of mislabelling, resulting in some samples having no replicates and others having more than two replicates. So we developed a pre-processing tool to detect samples with incorrect numbers of replicates. The samples without replicates were discarded; while samples having more than two replicates were checked for consistency of peaks. The two replicates with the most similar peak information were used in further analyses. We first illustrate how to pre-process the repeated peak data to get duplicate peak measurements which are used in sample size calculations. The `liverRawData` is the raw data from the Biomarkers wizard, while the `liverdata` is its pre-processed version and `liver.pheno` is a dataframe of sample phenotypic information.

3.2.1 Data pre-processing

```
> data(liverdata)
> data(liverRawData)
> data(liver_pheno)
> liverdata[1:4,]

  SampleTag CancerType Spectrum Peak Intensity
1       156          c        1    1 19.199355
2       156          c        1    2 24.144236
3       156          c        1    3 31.319952
4       156          c        1    4  4.515875
  Substance.Mass
1       1689.272
2       1776.455
3       1863.600
4       1883.988

> liverRawData[1:4,]

  SampleTag CancerType Spectrum Peak Intensity
1       156          c        1    1 19.199355
2       156          c        1    2 24.144236
3       156          c        1    3 31.319952
4       156          c        1    4  4.515875
  Substance.Mass
1       1689.272
2       1776.455
3       1863.600
4       1883.988
```

The short description of the data is

```
> names(liverdata)

[1] "SampleTag"      "CancerType"     "Spectrum"
[4] "Peak"           "Intensity"      "Substance.Mass"

> dim(liverdata)

[1] 13886     6
```

Here are the samples from the "raw" data, for which the number of replicates is not 2 (which is due to mislabelling, in most cases), the intended number of replicates:

```
> no.peaks <- 53
> no.replicates <- 2
> checkNo.replicates(liverRawData,no.peaks,no.replicates)

[1] "250" "25"  "40"
```

These samples must be pre-processed to:
(i) discard the information on samples which have no replicate data, and
(ii) for samples with more than 2 replicate expression data, only duplicates with most similar peak information are retained for use in subsequent analyses.
We can use the wrapper for pre-processing functions, `preProcRepeatedPeakData`, to pre-process the repeated raw mass spectrometry data from the Biomarker wizard. It identifies and discards information on samples that have no replicates. Then for the samples with two or more replicates, it selects and returns data for two replicates with most similar expression pattern. This is done as follows:

```
> threshold <- 0.80
> Data <- preProcRepeatedPeakData(liverRawData, no.peaks, no.replicates, threshold)
```

Only sample with ID 250 has no replicates and has been omitted from the data to be used in subsequent analyses. This fact may varified by using the `R` command, `setdiff`:

```
> setdiff(unique(liverRawData$SampleTag),unique(liverdata$SampleTag))

[1] 250

> setdiff(unique(Data$SampleTag),unique(liverdata$SampleTag))

integer(0)
```

Now filter out the samples with conflicting replicate peak information using the `spectrumFilter` function:

```
> TAGS <- spectrumFilter(Data,threshold,no.peaks)$SampleTag
> NewRawData2 <- Data[Data$SampleTag %in% TAGS,]
> dim(Data)

[1] 13886     6

> dim(liverdata)

[1] 13886     6

> dim(NewRawData2)

[1] 13886     6
```

The output is similar to the liverdata that is included in this package.
In the case of this data (the liver data), all technical replicates have coherent peak information, since no sample information has been discarded by spectra filter.
Let us have a look at what the pre-processing does to samples with more than 2 replicate spectra. Both samples with IDs 25 and 40 have more than 2 replicates.

```
> length(liverRawData[liverRawData$SampleTag == 25,]$Intensity)/no.peaks

[1] 3

> length(liverRawData[liverRawData$SampleTag == 40,]$Intensity)/no.peaks

[1] 4
```

Take correlations of the log-intensities to find which of the 2 replicates have the most coherent peak information.

```
> Mat1 <- matrix(liverRawData[liverRawData$SampleTag == 25,]$Intensity,53,3)
> Mat2 <-matrix(liverRawData[liverRawData$SampleTag == 40,]$Intensity,53,4)
> cor(log2(Mat1))

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.9830000 0.7939772
[2,] 0.9830000 1.0000000 0.7390008
[3,] 0.7939772 0.7390008 1.0000000

> cor(log2(Mat2))

          [,1]      [,2]      [,3]      [,4]
[1,] 1.0000000 0.9917671 0.9669138 0.9859144
[2,] 0.9917671 1.0000000 0.9705106 0.9915617
[3,] 0.9669138 0.9705106 1.0000000 0.9855580
[4,] 0.9859144 0.9915617 0.9855580 1.0000000
```

We see that the first two columns of both `Mat1` and `Mat1`, which contain the the raw intensities for samples with IDs 20 and 40, respectively, have the most similar peaks information (have the most highly correlated spectrum). The function that picks the most similar duplicate spectra is `mostSimilarTwo`. Let us check that it correctly identifies the first two columns of both `Mat1` and `Mat1`, as having the most coherent peak information:

```
> Mat1 <- matrix(liverRawData[liverRawData$SampleTag == 25,]$Intensity,53,3)
> Mat2 <-matrix(liverRawData[liverRawData$SampleTag == 40,]$Intensity,53,4)
> sort(mostSimilarTwo(cor(log2(Mat1))))

[1] 1 2

> sort(mostSimilarTwo(cor(log2(Mat2))))

[1] 1 2
```

Next, check that the pre-processed data, `NewRawData2`, contains similar information to `liverdata` (the allready pre-processed data, included in the *clippda*).

```
> names(NewRawData2)

[1] "SampleTag"      "CancerType"     "Spectrum"
[4] "Peak"           "Intensity"      "Substance.Mass"

> dim(NewRawData2)

[1] 13886     6

> names(liverdata)

[1] "SampleTag"      "CancerType"     "Spectrum"
[4] "Peak"           "Intensity"      "Substance.Mass"

> dim(liverdata)

[1] 13886     6

> setdiff(NewRawData2$SampleTag,liverdata$SampleTag)

integer(0)

> setdiff(liverdata$SampleTag,NewRawData2$SampleTag)

integer(0)

> summary(NewRawData2$Intensity)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 -0.4864   1.8280   4.3880   9.2020  10.2100 123.9000

> summary(liverdata$Intensity)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 -0.4864   1.8280   4.3880   9.2020  10.2100 123.9000
```

3.3 Data formatting

Most computations in this package are performed on data which are arranged in a format which can be averaged by the `limma` package function, `dupcor`, which is generated using the function called `sampleClusteredData`.

```
> JUNK_DATA <- sampleClusteredData(NewRawData2,no.peaks)
> head(JUNK_DATA)[,1:5]

       156      157       158       160      161
1 19.19936 4.277178 0.9830726  7.349703 16.78190
2 21.33371 3.248283 1.7195330  4.211342 10.18573
3 24.14424 6.566648 1.6985200 12.675141 28.03468
4 26.53655 5.966934 2.8839186  7.214974 17.35570
5 31.31995 7.710894 3.0078051 14.281464 49.45436
6 34.02668 6.655040 4.8840995  8.047355 33.70502
```

Two consecutive rows of this dataframe are expression values of same sample. The first column of this dataframe, for example, is given by:

```
> as.vector(t(matrix(liverdata[liverdata$SampleTag %in% 156,]$Intensity,53,2))[,1:5])

 [1] 19.199355 21.333709 24.144236 26.536552 31.319952
 [6] 34.026683  4.515875  4.797836  9.958806  7.276754

> length(as.vector(t(matrix(liverdata[liverdata$SampleTag %in% 156,]$Intensity,53,2))))

[1] 106

> as.vector(t(matrix(NewRawData2[NewRawData2$SampleTag %in% 156,]$Intensity,53,2))[,1:5])

 [1] 19.199355 21.333709 24.144236 26.536552 31.319952
 [6] 34.026683  4.515875  4.797836  9.958806  7.276754

> length(as.vector(t(matrix(NewRawData2[NewRawData2$SampleTag %in% 156,]$Intensity,53,2))))

[1] 106
```

This is the vector of duplicate expression measurements of the 53 peaks on the sample with ID 156.
We recommend that both the expression data and phenotypic information should be combined into a single coherent and well-documented data structure, by storing them in objects of `aclinicalProteomicData` class. We describe this in detail in the next section.

4 Combining expression data and phenotypic information into a single object of aclinicalProteomicData class

A clinical proteomic profiling study results in protein expression data, but there are often available records of phenotypic characteristics for the samples under study. Thus it is convenient to combine the datasets into a single well-annotated data structure with coherrent dimensions across the data slots, e.g. objects of `ExpressionSet` class, in the *Biobase* package. In the *clippda* package, we have defined a new `object` class for clinical proteomic profiling data, known as `aclinicalProteomicData`, which can be used to combine both expression data and phenotypic information from a clinical proteomic study into a single data structure which can be used directly in analyses. An `aclinicalProteomicData` class `object` has `slots` for storing: a matrix of raw SELDI data (`rawSELDIdata`), a character vector of sample covariates (`covariates`), a matrix of phenotypic data (`phenotypicData`), a character vector storing classes for phenotypic variables (`variableClass`), and a numeric value for the number of peaks of interest (`no.peaks`). I will now explain how to construct objects of a `aclinicalProteomicData` class, which are the input data for many generic functions in the *clippda* package. The following steps may be followed:
(i) First, we need to pre-process the "raw" replicate expression data from the Biomarkers wizard software to remove the inconsistencies caused by sample mislabelling. The codes for doing this were given in the last section, and the pre-processed data was stored in the object `NewRawData2`. Below is how to create data objects of a `aclinicalProteomicData` class:

```
> OBJECT=new("aclinicalProteomicsData")
> OBJECT@rawSELDIdata=as.matrix(NewRawData2) #OBJECT@rawSELDIdata=as.matrix(liverdata)
> OBJECT@covariates=c("tumor" ,    "sex")
> OBJECT@phenotypicData=as.matrix(liver_pheno)
> OBJECT@variableClass=c('numeric','factor','factor')
> OBJECT@no.peaks=no.peaks
> OBJECT

clinical proteomics dataType        : aclinicalProteomicsData
rawSELDIdata:
        matrix with: 13886 rows and 6 columns
phenotypicData:
        matrix: containing information on 131 samples and 3 variables
        varLabels: SampleTag tumor sex
variableClass: numeric factor factor
covariates: tumor sex
no.peaks: 53

>
```

An object of `aclinicalProteomicsData` class takes as arguments: a matrix of expression values, a matrix of phenotypic information, a character vector of covariates of interest, a character vector of classes for phenotypic variables, and a numeric value for the number of peaks of interest.

4.1 Extracting data from an `aclinicalProteomicsData` class object

We can extract data from various slots of an object of a `aclinicalProteomicsData` class using the command: `object@slotname`.
The `rawSELDIdata` may be obtained from an `aclinicalProteomicsData` class object using the command: `proteomicsExprsData(object)`. The `phenotypicData` may be obtained from that object using the command: `proteomicspData(object)`. Here is an example of how to extract the expression and phenotypic datasets from an `aclinicalProteomicsData` object. We only show the first few rows of these datasets.

```
> head(proteomicsExprsData(OBJECT))

  SampleTag CancerType Spectrum Peak Intensity
1       156          c        1    1 19.199360
2       156          c        1    2 24.144240
3       156          c        1    3 31.319950
4       156          c        1    4  4.515875
5       156          c        1    5  9.958806
6       156          c        1    6  6.516681
  Substance.Mass
1       1689.272
2       1776.455
3       1863.600
4       1883.988
5       1926.397
6       2020.212

> head(proteomicspData(OBJECT))

   SampleTag tumor sex
1         38     n   M
3        158     c   M
5        196     c   F
7        213     c   M
9         26     n   M
13        71     n   M
```

5 Sample size calculations

5.1 Obtaining the parameters

Now we will use the data from an `aclinicalProteomicsData` class object to calculate the sample size required when planning clinical proteomic profiling studies.
There are two possible sources of the data used in sample size calculations:
1. A pilot clinical proteomic profiling study.
2. *Reuse* data from previous clinical proteomic profiling studies.
Because of the difficulty with reproducibility in proteomic profiling studies, I find it more reasonable to *reuse* data from previous studies (with similar hypotheses to the proposed study) as the pilot data. This is especially useful if it is possibe to obtain data from multiple "previous studies". In this way, there is "replication" and one can use these multiple datasets to determine the range for plausible estimates of the parameters, which may be used to set up guidelines on the sample size required when planning a proteomic profiling study. The down-side of this method is that for some studies, the data on certain risk factors may be missing or incomplete, giving rise to incomplete sets of covariates. In this case, one can employ the method proposed by Nyangoma *et al.* (2009) to estimate the admissible effects of confounders.
First, the data is transformed by taking logarithms-to-base-two, and then used in calculating sample size parameters. The parameters required in sample size calculations are: the biological variance, the technical variance, the difference to be estimated, the intraclass correlation between technical replicates, significance level, and the power that need to be attained. The power and significance level, and the intraclass correlation (if known) are usually pre-specified. The rest of the parameters are computed from the pilot data (or data from previous studies) within the procedure for computing sample size (`sampleSize`) as follows:
(i) The biological variance, the differences to be estimated, and the significance of the peaks, may be obtained using the command:
`betweensampleVariance(object)`.
(ii) The technical variance is calculated using the command:
`sampletechnicalVariance(object)`.

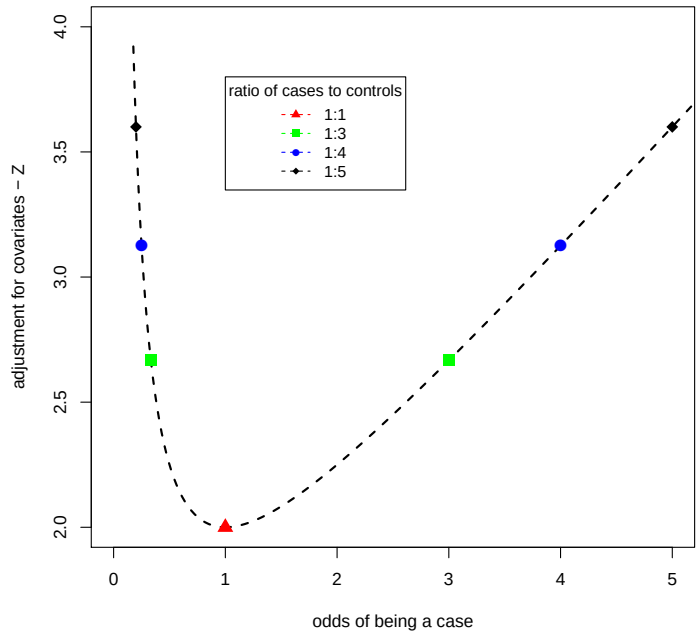
5.2 Graphical representation to aid in selection of adjustment factors for the effect sample imbalance

Some graphics may aid in understanding the consequences of using certain designs when planning a clinical proteomics study. A Plot of the Z values against the ratios of the corresponding proportions of cases in study to the proportions of the controls (i.e. odds of being a case), is the simplest tool that highlights the effects of sample imbalance on the sample size required when planning a proteomics study. The odds is a measure of the fold-change in the proportions of cases to controls. We feel that a choice in the proportions of cases to those

of controls which gives odds greater than a 3-fold change indicate extreme imbalances which are unlikely to be of practical use. This would lead to choices of $Z = 2 + c$, where $0 \leq c \leq 1$. We have plotted several Z values against their corresponding odds and highlighted on this plot some values of odds and corresponding Z from some designs which are likely to be used in practice when planning a profiling study, for example: balanced design (i.e. 1:1) and unbalanced designs (i.e 1:3, 1:4 and 1:5). This plot can be done in `clippda` using the commad:

```
> probs=seq(0,1,0.01) # provide hypothetical proportions of cases vs controls
> ZvaluescasesVcontrolsPlots(probs)

null device
          1
```

This command saves the plot as a pdf file: ZvaluescasesVcontrolsPlots.pdf in a working directory. We append this plot here

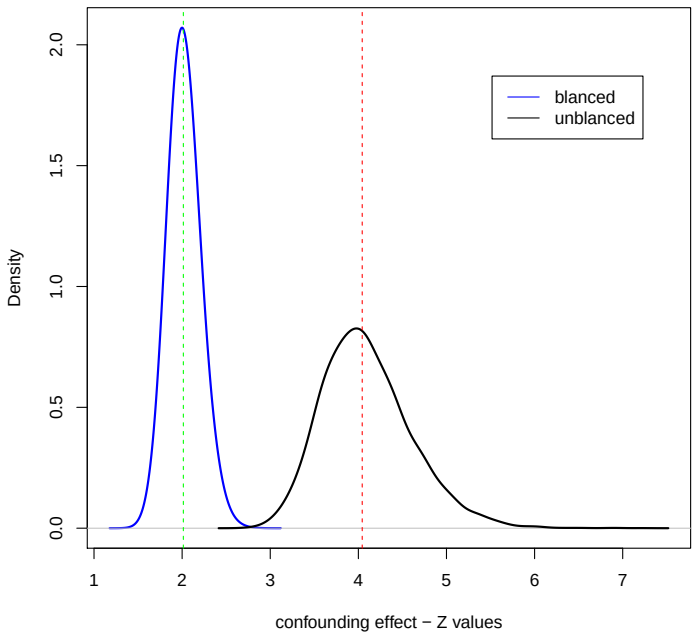


If other parameters (e.g. intraclass correlation, protein variance, significance level) are held constant, a balanced design gives rise to the minimum Z value, $Z = 2$, and hence the smallest sample size compared to unbalanced designs. This means you get same power at a smaller sample size if a study is balanced.

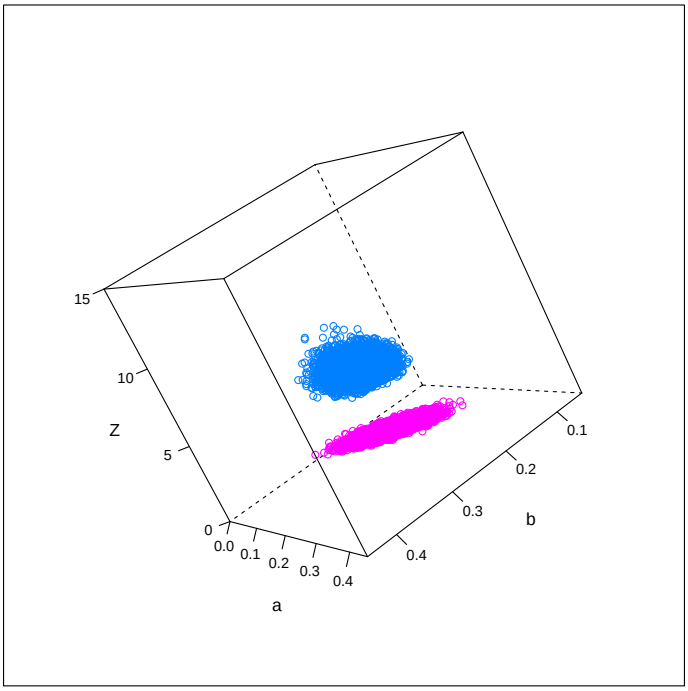
The visualization method becomes more complex as the dimension of the covariates space increases. When we have binary exposure and confounder, then a cross-classification of individuals into these categories gives rise to 3-dimensional (3D) multinomial distribution of the number of cases. When repeated multinomial sampling is done under a specific scheme, then the proportions of elements in various cells vary in the 3D space, and Z values describe ellipsoid with respect to any 2D subspace of the 3D multinomil space. Three hundred individuals were proportionately sampled 10000 times in the ratio 1:1:1:1. The same individuals were again sampled 10000 times following an unbalanced design: 1:2:1:7. We first plotted the densities of the Z values generated from these experiments using the following command

```
> nsim=10000;nobs=300;proposeddesign=c(1,2,1,7);balanceddesign=c(1,1,1,1)
> ZvaluesfrommultinomPlots(nsim, nobs, proposeddesign, balanceddesign)

$Zvalue
[1] 4.043732

$varZ
[1] 0.2569432

>
```

The density plot indicates that the Z values from a balanced design cluster around $Z = 2$, while those of an unbalanced design cluster around $Z = 2 + c$, where here $c \approx 2$.



Thus as in the simplest unbalanced design when there is a binary exposure and one binary confounder, the choice of $Z = 2$ leads to the smallest sample size, which happens to be the case only if a study is balanced. If some degree of unbalancedness is expected, then larger values of Z must be chosen.
The Z values plotted against elements of any 2D subspace of the 3D space of multinomial probabilities lead to ellipsoids shown in the figure below



Again, we can clearly see $Z = 2$ for a balanced study and attains higher values for various degrees of unbalancedness.

5.3 Sample size calculations

The sample size can be computed using the function `sampleSize`. This function, first computes the input parameters for sample size calculation, using the function, `sampleSizeParameters`. The other useful input parameter is the heterogeneity correction factor, Z. If full covariate information is known then Z is the second diagonal element of the output matrix expected Fisher information obtained using the command: `fisherInformation(object)`. However, if there is incomplete set of sample covariates (or if no covariates can be found), it would be useful to derive a range of plausible of Z using the method proposed by Nyangoma *et al.* (2009).
Here, we use the covariates that were available to us (i.e. cancer class, sex, and age) to gauge the degree of data imbalance. This estimate may be used as a guideline when setting up the range of possible values of adjustment factors for data imbalance (i.e. the Z values). In these computations, it is assumed that only duplicate data are available.

```
> intraclasscorr  <-  0.60
> signifcut <- 0.05
```

```
> Data=OBJECT
> sampleSizeParameters(Data, intraclasscorr, signifcut)

$Corr
[1] 0.8802024

$techVar
[1] 0.7510302

$bioVar
[1] 3.211113

$DIFF
[1] 0.3385422

> Z <- as.vector(fisherInformation(Data)[2,2])/2
> Z

[1] 2.075855

> sampleSize(Data, intraclasscorr, signifcut)

$protein_variance
[1] 4.1

$replicate_correlation
[1] 0.8802024

$difference
[1] 0.3385422

$sample_size
        alpha0.001 alpha0.01 alpha0.05 alpha0.001 alpha0.01
beta0.1       1391       990       699       1443       1027
beta0.2       1136       777       522       1179        806
beta0.3        968       639       411       1005        664
        alpha0.05
beta0.1       725
beta0.2       542
beta0.3       426
```

Note that the value of Z is only about 4% above the nominal Z value, i.e. $Z = 2$, which is used in the classical sample size calculations, but it has a noticeable effect on sample size. For example, At 70% power and 5% level of significance, the sample size has increased from 411 to 426. We have seen worse imbalances in many studies where heterogeneities are above 40% of the nominal value. For this data set, the biological variance (`bioVar` = 3.2), is particularly large and

this has inflated the number of biological samples required. We believe that the sample size required could be much lower, if the biological variance could be appropriately controlled.

6 Display of the results

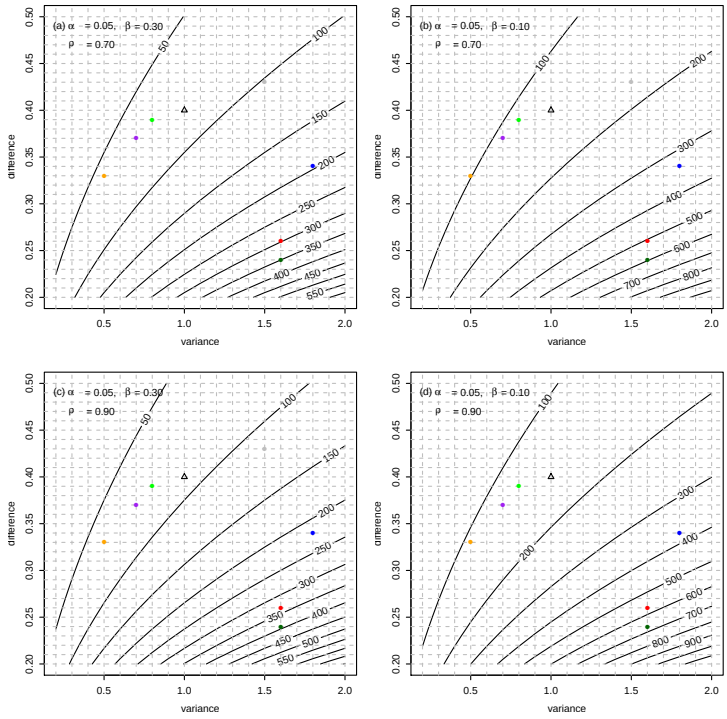
The following commands will produce a contour plot of sample sizes over a range of values for clinically-important differences versus protein variances. The plot will be saved in your working directory. The sample size contours are generated from the outer product of the differences (0.13 - 0.5) and variances (0.2 - 4.0). Combinations of the parameters in this region give sample size values that are likely to be achieved in practice. In contrast, the parameters outside these ranges result in sample sizes that are too large to be of practical use. Intraclass correlations used are 0.70 and 0.90, the powers used are 0.70 and 0.90; and the significance level considered is 0.05. On the grid, we have plotted a number of sample sizes we computed from real-life data from several SELDI and MALDI proteomic profiling studies, for example: late-stage (MALDI - green), early-cancer (SELDI - blue, MALDI - red), colorectal serum (SELDI with different chip-type: IMAC - purple, CM10 - gray, Q10 - orange). The description of the data can be found in **?**. These results can be used to formulate a general guideline for the number of samples required to plan a proteomic profiling study. You will immediately see that using fewer than 50 samples, will not result in any meaningful estimation of differences. For late-stage cancer, you would need the fewest number of samples, even using a very variable sample-types such as urine. You typically need more samples (over 200) to estimate differences between early stage cancer and non-cancer controls. The output will contain four contour plots describing various combinations of the above parameters. You will notice that, as expected, for more power, you need larger sample sizes.

```
> m <- 2
> DIFF <- seq(0.1,0.50,0.01)
> VAR <- seq(0.2,4,0.1)
> beta <- c(0.90,0.80,0.70)
> alpha <-  1 - c(0.001, 0.01,0.05)/2
> Corr <- c(0.70,0.90)
> Z <- 2.4
> Indicator <- 1
> observedPara <- c(1,0.4) #the variance you computed from pilot data
> #observedPara <- data.frame(var=c(0.7,0.5,1.5),diFF=c(0.37,0.33,0.43))
> sampleSizeContourPlots(Z,m,DIFF,VAR,beta,alpha,observedPara,Indicator)

null device
          1
```

You may input parameters from your pilot study into the plot, e.g. the following hypothetical values

```
> observedVAR=1
> observedDIFF=0.4
```

The result will be plotted as a triangle in your plot. You may set these values to 0, if you do not have pilot data, in which case the values computed from my pilot studies (dotted on the plot) may be used to draw guidelines for sample size required to plan a cancer proteomic profiling study. If you have what is believed to be appropriate clinically important differences, you can also determine the required sample sizes from the sample size contour plots. For example, if we assume that the clinically important difference is 0.35 then the sample sizes for given values of protein variances, are the values at the intersection of the sample size contours and the horizontal line with an intercept of 0.35 on the "difference" axis on the grid of differences versus variances (i.e. the blue dotted line). We include the plot generated using the above commands:



You may also visualize your results using 3D scatterplots through the `sampleSize3DscatterPlots` as follows

```
> Z <- 2.460018
> m <- 2
> DIFF <- seq(0.1,0.50,0.01)
> VAR <- seq(0.2,4,0.1)
```

```
> beta <- c(0.90,0.80,0.70)
> alpha  <-  1 - c(0.001, 0.01,0.05)/2
> observedDIFF  <-  0.4
> observedVAR  <-  1.0
> observedSampleSize  <-  80
> Indicator  <-  1
> Angle  <-  60
> sampleSize3DscatterPlots(Z,m,DIFF,VAR,beta,alpha,observedDIFF,observedVAR,observedSampleSi

null device
          1
```

A plot from our examples is given below