Alternative Dimensionality Reduction Methods to find Population Structure in Genetic Data

Submitted in partial fulfillment of the requirements of the degree of Master's in Statistical Data Science.

Tatiana Hammond-Antrobus

Student ID: 23044157

School of Mathematics, Statistics and Actuarial Science

The University of Kent, Canterbury

September 2023

Abstract

Improving population structure terms will reduce spurious correlations between Single Nucleotide Polymorphisms (SNPs) and affection status variables, improving predictions in Genome-Wide Associaton Studies. Each simulation in this project used principal(), factanal(), and fastICA() functions in R to generate population structure terms derived through Principal Component Analysis, Factor Analysis, and Independent Component Analysis, respectively. Then glm() was applied to produce disease-association models for all SNPs. The disease was not associated with the SNPs by design, so the sum of coefficients from the glm() indicated inflation; for each simulation, the best dimensionality reduction method was the one that consistently produced association models with the lowest coefficient sums. Of the 12 simulations, 11 favoured population structure terms derived through Independent Component Analysis. In conclusion, the simulations in this project suggest Independent Component Analysis is a better dimensionality reduction method than Principal Component Analysis for Genome-Wide Association Studies.

Keywords Section 1.2 · genome · genotype · phenotype · nucleotide · SNPs · polymorphic · monomorphic · population structure · affection status variable · phenotype of interest · manhattan plot · qq plot · linkage disequilibrium · test statistics inflation · genomic inlfation factor · λ_{1000} · PCA · principal components · scores · FA · factors · manifest variables · latent variables · communality · uniqueness · ICA · independent components · independence · supergaussian/Laplacian · nonlinear uncorrelatedness · decorrelation · unmixing matrix · maximal nongaussianity · kurtosis · Central Limit Theorem

Contents

1 Introduction

Principal Component Analysis has been the dominant, if not sole, dimensionality reduction technique in the field of genomics thus far. With an intent to improve dimensionality reduction technqiues in Genome-Wide Association Studies for more efficient disease-association models, this project aims to compare Principal Component Analysis (PCA) with two other potentially suitable dimensionality reduction methods: Factor Analysis (FA) and Independent Component Analysis (ICA).

1.1 Context

Genomics is a relatively young field of research already widely applied: from trying to identify migratory patterns of our ancestors [40], to predicting the diseases to which an infant might be predisposed [7]. Over a year ago, I submitted an uncomfortable scraping of cheek cells to CRI Genetics, and have since been informed that I am predisposed to hyperuricemia, that 45% of my ancestors are German, and that centuries ago, an ancestor of mine was no doubt revolutionising potato farming in Peru; it is easy to get carried away taking such revelations for fact, but in reality all these insights and more are likelihoods, calculated as well as modern techniques allow, through Genome-Wide Association Studies (GWAS).

A **genome** is the sequence of genes an individual has in the vast majority of his cells. While **genotype** refers to what genes a unique individual has in his genome, **phenotype** refers to the expression of those genes as traits, for example, eye colour or heritable conditions like Sickle Cell Anemia. Each gene is a string of **nucleotides**, the four ‘encoding’ chemicals often referred to by their initials: A, T, C, and G.

In genomics, one simple approach to identifying what makes a genome unique is to look for **Single nucleotide polymorphisms (SNPs)**. SNPs describe genetic mutations that are the result of swapping one nucleotide in a sequence for another. For example, suppose the eight-hundred-and-fifth nucleotide on one particular chromosome is Adenosine in most people, but it can sometimes also be Guanine; the eight-hundred-and-fifth single nucleotide would thus be **polymorphic**, as it can take on multiple forms. With roughly 10^7 nucleotides in a single human genome, there are many opportunities for SNPs to create meaningful differences in gene expression; however, if a particular SNP doesn’t result in any phenotypical diversity, it still sits as a marker for genetic diversity.

Population structure aims to describe the genetic diversity within ethnic groups by summarising the variance of SNPs between people; this is typically achieved by applying

Principal Component Analysis to the genomes of many people from a single ethnic group. However, *other dimensionality reduction methods might be more suitable.*

A GWAS might be carried out to identify population structure for many reasons; one is to correct for spurious correlations in a disease association model. A disease association model can be a generalised linear model (GLM) in which the **affection status variable** (also known as the response variable, the **phenotype of interest (POI)**, or the **disease status**) is predicted by a SNP or SNPs, plus one or more correction terms.

To understand why the correction term is necessary, temporarily assume we don't have one in the model. Since the SNPs contribute to genetic variation within a population, the covariance matrix of SNPs will not be an identity matrix; in other words, SNPs have at least some association with each other. Now suppose only one particular SNP, call it SNPa, truly impacts the probability of someone developing the phenotype of interest. Because of natural population structure, another SNP, SNPb, happens to be positively correlated with SNPa. When we run a GLM for the affection status variable against SNPb, we "discover" that SNPb is associated with the disease! Extrapolate the process and suddenly you have 10^7 potential spurious correlations with the affection status variable.

The example above also illustrates how a correction term can be identified.The correction terms represent population structure, so generally, they are derived from the axes of maximal variance, known as principal components in PCA. By incorporating such terms, a GLM will then only assign a high coefficient (indicating a strong association) to a SNP if its influence on the affection status variable is significantly greater than what could be expected, given that SNP's position (score) on the axis of maximal variance.

1.2 Terminology

This project will apply multiple dimensionality reduction methods. The **principal components** of PCA, the **factors** of FA, and the **sources** of ICA are all **vectors** that define new **axes** that somehow better encapsulate the data's commonalities and uniquenesses[1] than did the available measures. In the genomic context, these new axes represent the population structure.

Scores refer to the projections of each data point onto each axis. The **scores** value is a consistent term across all three dimensionality reduction functions (PCA, FA, and ICA).

From genomics, the terms **linkage disequilibrium** and **genomic inflation factor** also appear in this project. Arbitrarily named ([44]), linkage disequilibrium is "the nonrandom

[1]I use these terms colloquially though they have technical definitions in FA.

association of alleles at different loci" and can indicate the presence of genuine causal mutations (SNPs that are truly associated with a phenotype) or spurious correlations due to population structure, or both, according to [53]. The genomic inflation factor, λ, is a measure of “the bulk inflation and the excess false positive rate” [3] from a genomic association study. λ is further discussed in sections 2.1 and 3.3. In this paper, ‘inflation’ can also apply to the overestimation of the test statistics and parameters as a result of a model’s unsuitability.

1.3 Main Goals

The overarching goal is to compare the suitability of PCA, FA, and ICA for generating population structure in GWAS. In service of that overall question, this project will:

- Source real genome data from the multiple hapmap projects using Plink. (The 1000 Genomes from [12] project is also a good source of real genomes, but the size and formats of the data are beyond the scope of this project.)
- Select and clean subsets of SNPs from the genome data into formats and dimensions that fit the functions related to the tasks below. In the following tasks, I refer to these data subsets as the SNPs.
- Each simulation will then contain the following stages:
  - Carry out Principal Component Analysis on each dataset, and calculate corresponding scores.
  - Carry out Factor Analysis on each dataset, and calculate scores.
  - Carry out Independent Component Analysis on each dataset, and calculate scores.
  - Generate an affection status variable, called ’disease status’, in order to simulate a disease-association model.
  - Construct a data frame containing the SNPs, the affection status variable, and all scores; this step is essential for the following glm() function.
  - Calculate coefficients for GLMs. Each GLM will have disease status as the response variable, with predictors of one SNP[2], and a number of sources from one of the three dimensionality reduction methods. Call all GLM with different SNPs but the same sources part of the same “model structure.”

[2]See section 7 for an example of an earlier attempt at GLM code simultaneously regressed over all SNPs of a small dataset.

  - For each distinct model structure, sum the coefficients of the SNPs. The model structure with the lowest sum should indicate the model structure with the least inflation, and thus identifying the sources that most accurately describe the population structure.

- Repeat the simluation steps for multiple SNP data subsets and make a note of which dimensionality reduction methods produce the best sources in each instance.

2 Preliminary Research and Tasks

2.1 Literature Review

To clarify the foundational knowledge of genetic biology required of this project, and to find images for this project, I read [52], and [38]. For an introduction to using Plink, I watched [34]. Further advice on accessing and processing genome data came from [12], [13] and [33].

To develop skill and understanding with dimensionality reduction methods, I revisited lectures ([45]) for PCA and FA, and the foremost book on the subject ([24]) for ICA. I used [18], [3], [9], [48] and discussions with my project advisor, Dr. James Bentham to develop an understanding of λ_{1000}; I am especially grateful to [54] and [26] for putting into laymen's terms the genomics approaches for checking goodness of fit. To clarify my writing on various statistical concepts, I revisited lectures [46] and read [4].

The following sources expanded and refined my understanding of the applications of dimensionality reduction in GWAS:

- For clearer protocols and history surrounding the use of PCA to find population structure, I checked [43] and the educational guidelines, [11].
- To further explore an application of GWAS more similar to the ancestry tracing websites, I found [40] and [36], which highlighed new interpretations of principal components of 'spatial data.'
- A discussion of statistical power, error rates, and methods, new and old, for selecting the number of principal comonents and assessing the strength of SNP associations can be found in [16], which also serves as an example of a simulation-based means of testing GWAS methods.
- Another potentially novel improvement on the current GWAS approaches, [31] proposes stepwise linear regression mixed models as a unified method of GWAS and genomic selection.

- A chapter in [53] confirmed the interpretation of Manhattan Plots and QQ Plots in GWAS, and contributes to a wider discussion on dimensionality reduction and interpretability.
- [42] examines the quality control behind the 1000 Genomes Project.

2.2 Working around Plink

Developed by [39], Plink is the foremost program used by genomists to carry out GWAS; therefore, my initial expectation for this project was to learn to code entirely in Plink, running all the GWAS there as one might write and run many statistical functions in R. To this end, I started the tutorial found at [5]. However, since Plink only runs one form of dimensionality reduction, this approach would be insufficient for the central question of this project. The next approach was simply using Plink to convert between file formats, as shown in the table below, and then reading the suitable file into, and subsequently working solely in, R. Ultimately, [5] and investigation in further sources ([2], [28], [14], [17]) enabled me to work entirely in R using the read.pedfile function from the package trio, or later, from the package LEA [48], using the ped2geno() and read.geno() functions.

file pipeline	code in CMD Prompt
downloading...	>cd C:YourChosenDirectory
.ped and .map	>plink –file hapmap1
.bed, .bim, .fam, .irem	>plink –file hapmap1 –make-bed –out hapmap1

2.3 Procuring Data Sets

In order to save time and computer memory while fine-tuning the code, I initially downloaded and cleaned (Section 4.1) a small subset of the hapmap genome downloaded from [5], which covers a population of Chinese and Japanese individuals. For reproducability, I also developed and applied the code to other genomes, including another subset of the hapmap dataset shared by Dr. James Bentham, and another sourced from [32].

3 Functional Foundations

In this section we examine novel functions applied in the project, both in a mathematical context, as well as any essential information provided in R Documentation. [48].

3.1 Dimensionality Reduction Methods

In a dataset, p can stand for the number of parameters (or measures taken for each data point), and n most often stands for the number of samples (or data points, or individuals) in the dataset. A covariance matrix between p explanatory variables can be calculated. If the number of samples is significantly greater than the number of variables, $n > p$, then it's more likely that statistically meaningful conclusions about the data can be drawn from various approaches. However, if Exploratory Data Analysis (EDA) reveals high-dimensional data ($p \geq n,$ or even $p \to n$), as well as a highly correlated covariance matrix of the explanatory variables, then dimension reduction is one approach to improving tractability and interpretability of any further analysis of the data ([45]).

Dimensionality Reduction methods can vary in complexity, from a simple 'total score' approach on an exam, to a multi-step cleaning and decorrelation process which calculates and manipulates kurtosis across all p parameters in order to construct statistical independence. Some dimensionality reduction approaches identify new axes as linear combinations of the original parameters that more efficiently describe some quality of the data as a whole, in which case the number of new axes should *generally* be less than p. Other approaches may re-group the data along axes of maximal *co*variance (as opposed to maximal variance), or with separating lines (an approach known as Linear Discriminant Analysis). Since this project is about GWAS, with each genome drawn from a member of the same ethnic group, yet carrying genes entirely unique to the individual (apart from twins), the most suitable methods to identify population structure will be conceptually similar to PCA; that is, rather than seek to regroup the data, suitable methods will, as [24] put it, boil down to finding a linear model of the data as it relates to some lower-dimensional variables (these vectors otherwise known as "sources" in ICA, "factors" in FA, and "principal components" in PCA).

3.1.1 Principal Component Analysis

The main approach for dimensionality reduction in PCA is to identify *uncorrelated* linear combinations of the original parameters. That is:

- a_i is the ith eigenvector of the covariance matrix of the data, S_x. Each a_i represents another axis of variation.
- $z_i = a_{ik} \cdot x_k$ are the projections of the original data onto each new axis of variation.
- The axes are constructed so that

$$Cor(z_i, z_j) = 0 \quad \forall i, j \in [1, p]$$

and

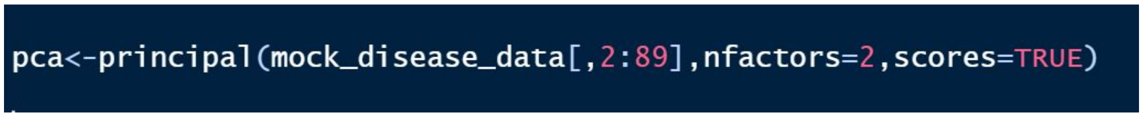
$$z_i \perp z_{i+1}$$

The numbering i is dictated by each eigenvalue, with the largest indicating the new axis captures the most variance of the original data. As each subesequent eigenvector is incorporated into the new parameter space, more of the variance of the data is accounted for. Hueristically, once the eigenvectors cumulatively summarise 80% of the original variance of the dataset, the model sufficiently describes the data; thus, even though p vectors are calculated, not all need be included in the final model. One additional constraint is each vector must be a Euclidean unit vector, to prevent an artificial inflation of the "variance captured" by each.

In summary, PCA constructs eigenvectors a_i of the covariance matrix of the data (x_i), such that the scores, $z_i = a_i \cdot x_i$ are uncorrelated and orthogonal. Ordered by the eigenvalues of a_i, the first q of the z_i are selected such that they capture a sufficient proportion of the variance of the original x_i. The scores are the projections of each of the n samples onto the new axes.

R has "values" (outputs from the code which can be called with a $) and "arguments" (inputs). With respect to PCA, the function principal(), from the *psych* package, uses the following terminology [48]:

Figure 1: Arguments of the function principal()

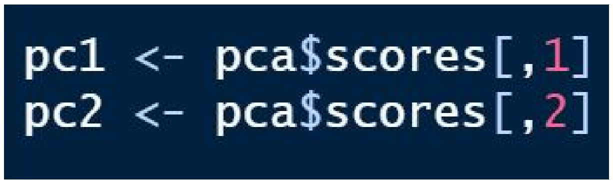


```
pca<-principal(mock_disease_data[,2:89],nfactors=2,scores=TRUE)
```

- Arguments: See figure 1.
  - The first entry, "'mock_disease_data[,2:89]" is the dataset to be analysed.
  - nfactors dictates how many axes to calculate (rather than outputting all and leaving the user to select based on the aforementioned hueristic), and by extension, how many column vectors of scores will be projected.
  - "scores" dictates whether scores will be produced as a value. Since we want to use the scores as the population structure, we select TRUE in this example.
- Values:
  - Of primary interest, in figure 2, you can see how scores are called, and in figure 20 you can see how I will separate each score to its own vector of length 89.

- "loadings" indicates the contribution of each sample to each principal component. See figure 22.
- A few of them are displayed in figures 21 and 23.

Figure 2: This is how scores are generated.



```
pc1 <- pca$scores[,1]
pc2 <- pca$scores[,2]
```

3.1.2 Factor Analysis

The key approach for FA stems from a subtly different interpretation: the observed relationships between **manifest** (i.e. observable and potentially informative) variables are the results of relationships between the manifest variables and their unobservable **latent** variables, also known as factors. In contrast to PCA, which is essentially "model free" ([45]), FA constructs a linear model to describe these relationships.

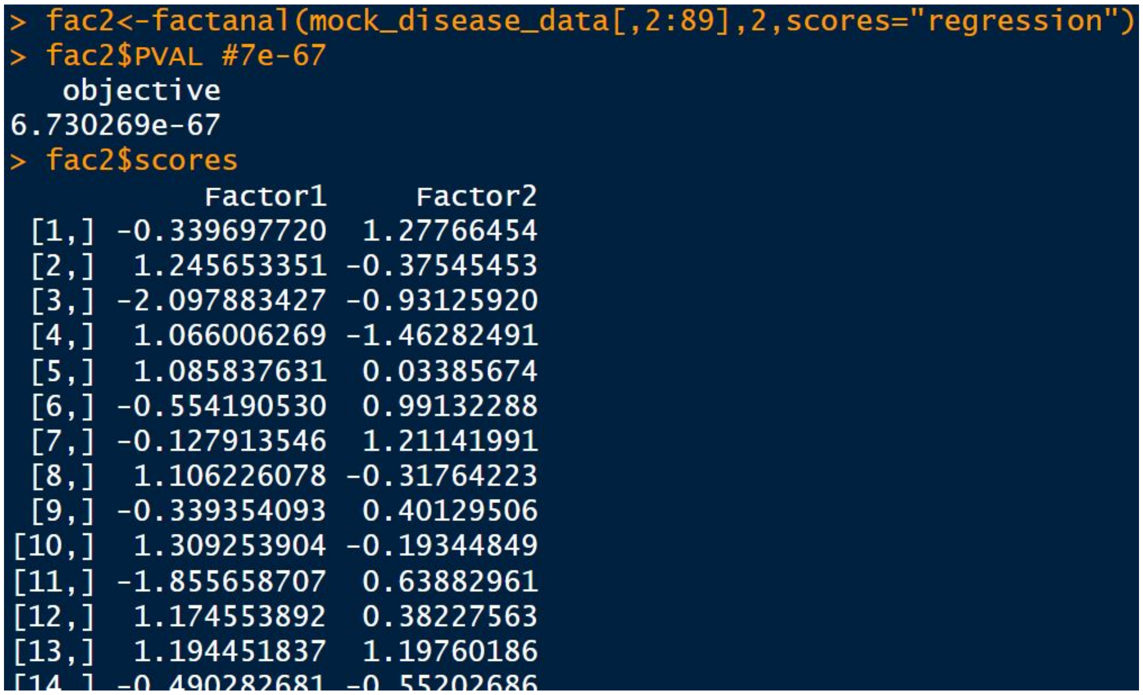
Though **communality** and **uniqueness** are also possible values in the principal() function, they are foundational to understanding FA. The communality is the variance common to all X_i data, the result of the manifest variables' shared latent variables, Y_j. The uniqueness is the variance unique to each individual X_i; in other words, it is $Var(e_i)$, where e_i is the error term in the presumed linear model for X_i with respect to the Y_j. In total, $Var(X_i) = communality(X_i) + uniqeness(X_i)$. In matrix notation, this is summarised as:

$$\Sigma = \Lambda\Lambda^T + \Psi$$

where Σ is the covariance matrix, Λ is a matrix with dimension $p \times q$ (where p is the number of manifest variables, q is the presumed number of factors, and $q < p$) such that $\Lambda\Lambda^T$ is a communality matrix (with rank limited by q), and Ψ is a diagonal matrix of uniquenesses. Stochastically reducing the uniqueness of all X_i, FA attempts to reach the most "saturated" model, such that *as much as possible* of X_i is explained by Y_i. Using Maximum Likelihood Estimation (MLE), a maximiser of the uniqueness matrix is found analytically, and then used to numerically calculate the uniqueness matrix; this process assumes $\vec{X}^{(i)} \sim MN(\vec{0}, \Sigma)$, and thus, the MLE is robust if the manifest data is reasonably close to Normal.

In the R package *stats*, the function factanal() works as shown in figure 3. The following facets of FA are of primary interest to this project:

- Arguments:
  - matrix of manifest variables
  - number of factors to build into the model
  - "scores" indicates how the scores will be calculated
- Values:
  - As in PCA, the scores serve as our dimensions representing population structure.
  - The PVAL in Factor Analysis is evidence to reject the null hypothesis that the current q-factor model sufficiently describes the manifest data[3]. Communalities greater than 1 indiciate spurious solutions that may result from high dimensional data, or from a selection of too many, or too few, factors.



```
> fac2<-factanal(mock_disease_data[,2:89],2,scores="regression")
> fac2$PVAL #7e-67
   objective
6.730269e-67
> fac2$scores
              Factor1     Factor2
 [1,] -0.339697720  1.27766454
 [2,]  1.245653351 -0.37545453
 [3,] -2.097883427 -0.93125920
 [4,]  1.066006269 -1.46282491
 [5,]  1.085837631  0.03385674
 [6,] -0.554190530  0.99132288
 [7,] -0.127913546  1.21141991
 [8,]  1.106226078 -0.31764223
 [9,] -0.339354093  0.40129506
[10,]  1.309253904 -0.19344849
[11,] -1.855658707  0.63882961
[12,]  1.174553892  0.38227563
[13,]  1.194451837  1.19760186
[14,] -0.490282681 -0.55202686
```

Figure 3: Factor Analysis arguments and values

While PCA simply identifies new dimensions as the eigenvectors of a covariance matrix, FA presumes a relationship between the obeserved data and some 'sources' (also known as factors or latent variables). Given that genes evolve from different environmental challenges along with the random happenstance of parentage, it makes at least philosophical sense

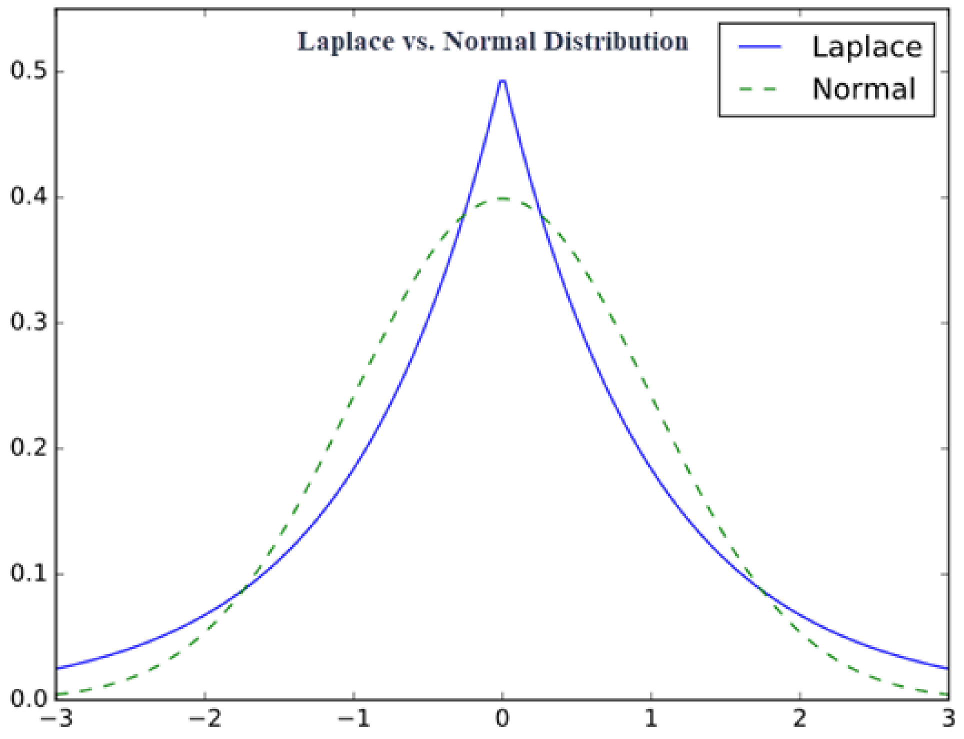
[3]For such high dimensional data as in our SNP simulations, there was strong evidence to reject every null hypothesis for every size of model proposed. Thus, the hypothesis testing feature of factanal() was not applied in these simulations. You can see examples of widespread null model rejection for FA in section 7.1

that a factor model might produce better scores for population structure than would simple principal components. However, FA is not necessarily a robust model if the SNPs don't follow a normal distribution.

3.1.3 Independent Component Analysis

With a different process and different assumptions regarding the data, Independent Component Analysis (ICA) is a possible alternative to FA and PCA. Factor Analysis holds the assumption that the manifest variables are Normally distributed; because uncorrelated components of Gaussian data are always independent, this stipulation is essential for drawing meaningful conclusions from Factor Analysis. In contrast, ICA aims to find statistically independent components (meaning the new vectors are "as independent as possible" according to constraints) for **nongaussian** data. For instance, real-world datasets often have **Laplacian** (also known as supergaussian) distributions, as shown in figure 4.

Figure 4: Taken from [6], the Laplacian distribution has the same mean as the Gaussian, but a higher peak and heavier tails.



Similarly to FA, ICA begins with the assumption that there is some matrix, A that "mixes" the unknown sources, s_j (latent variables) to produce the observed data, x_i. The model A can be estimated if and only if the components, s_j are nongaussian.

Recall the orthogonality constraint of PCA; for normal data, orthogonality gives rise to uncorrelatedness. However, **independence** is a stronger assumption than uncorrelatedness. The following summary of ICA estimation comes from [24].

A key concept in the process of ICA is:

Independence implies nonlinear uncorrelatedness.

ICA seeks out the unmixing matrix, A^{-1} such that for any $i \neq j$, $Cor(s_i, s_j) = 0$ and some suitable non-linear transformations of the sources, $g(s_i)$ and $h(s_j)$, are also uncorrelated; this principle of estimation is called **nonlinear decorrelation.**

A key issue with ICA estimation comes from the Central Limit Theorem:

The sum of nongaussian distances approaches as Gaussian distribution.

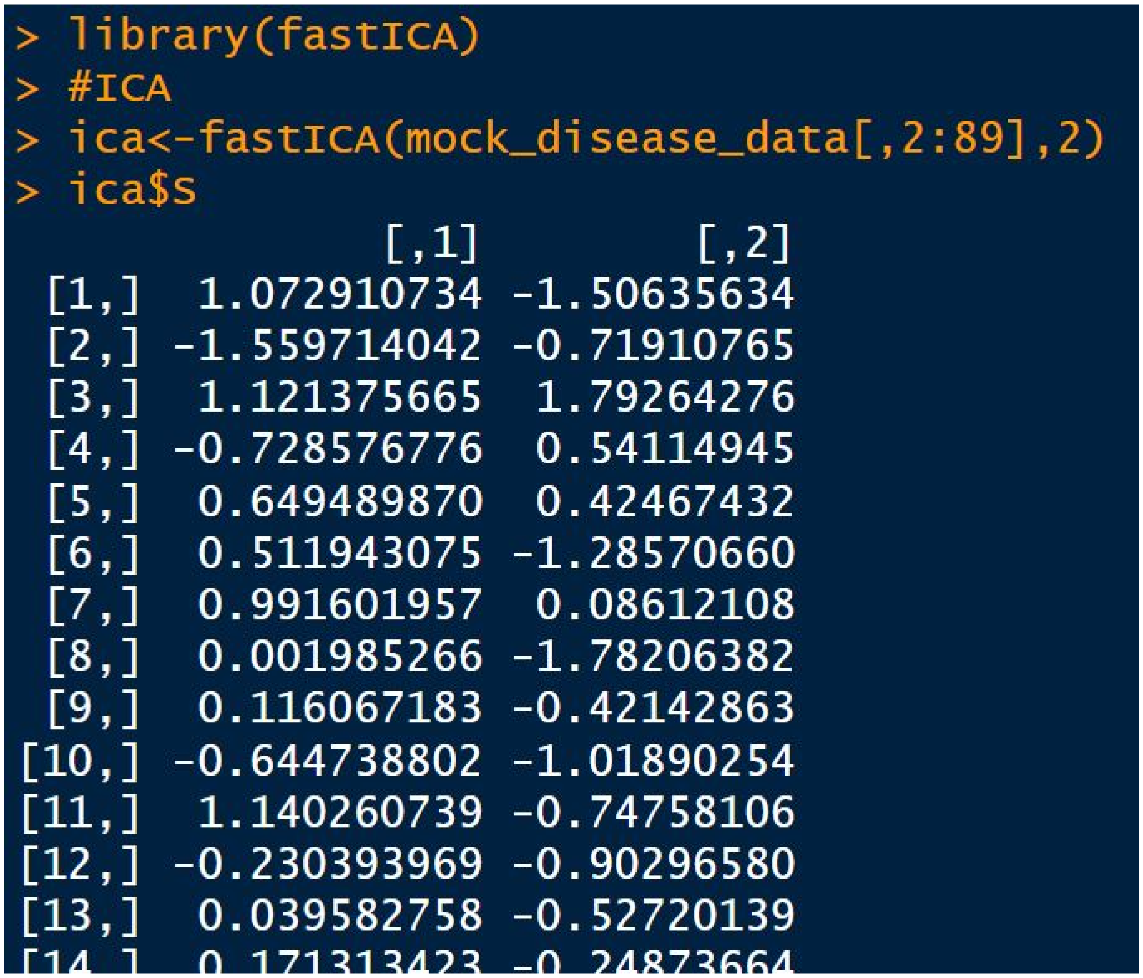
Thus, the second principle of ICA estimation is **maximal nongaussianity**: Using the unmixing matrix, every source is a linear combination of the data, i.e. $s = \sum_i b_i x_i$. By finding a local maximum of nongaussianity for s under the constraint that $Var(s)$ is constant, each local maximum gives one independent component. In practise, nongaussianity is measured by kurtosis, which is not contained in Σ, the correlation matrix, but instead is calculated through method of moments.

There are many approaches for computation of ICA estimates. In theory, linear algebra is insufficient for ICA, since its estimations are nonlinear, and often non-quadratic. However, typical numerical approaches for ICA optimisation include gradient descent, and a fixed point algorithm called fastICA, which is used in this project.

From the R package *fastICA*, implementation of the function fastICA() is as simple as for PCA and FA. See figure 5 for an example:

- Arguments:
  - The data to be ’unmixed’
  - The number of sources (aka components) the algorithm should produce.
- Values:
  - Diverging slightly, all values in fastICA are single letters. $\$S$ is the value for scores.

Figure 5: In order to allow some of our dimensionality reduction methods to work, it was essential to limit the SNP columns to the first 88, so that $n > p$. Thus, mock_disease_data[,2:89] is the 88 manifest variables. 2 is the number of sources to be produced. Logically, each source, $\$S$, is of length 89, to correspond with the 89 sampled individuals.



```
> library(fastICA)
> #ICA
> ica<-fastICA(mock_disease_data[,2:89],2)
> ica$S
              [,1]        [,2]
 [1,]  1.072910734 -1.50635634
 [2,] -1.559714042 -0.71910765
 [3,]  1.121375665  1.79264276
 [4,] -0.728576776  0.54114945
 [5,]  0.649489870  0.42467432
 [6,]  0.511943075 -1.28570660
 [7,]  0.991601957  0.08612108
 [8,]  0.001985266 -1.78206382
 [9,]  0.116067183 -0.42142863
[10,] -0.644738802 -1.01890254
[11,]  1.140260739 -0.74758106
[12,] -0.230393969 -0.90296580
[13,]  0.039582758 -0.52720139
[14,]  0.171313423 -0.24873664
```

3.2 Generalised Linear Models

The glm() function performs regression such that some link function, η, is a linear function of the input variables, or predictors ([47]).

$$\eta = \beta X \quad , \quad \eta = f(y)$$

The exact link function is determined by the distribution of the response variable ([46]). Since y, the presence of a phenotype of interest is binary, the "family" argument for glm() will be binary in this case; this means η is the log-odds of y. There are many output values from glm(), and this project will focus on the coefficients value.

Since our project methodology is to identify the coefficient of each SNP one GLM at a time, we need a fast and simple way to run the GLM for each model structure on every indiviudal SNP; thus, R's apply() function ([48]) will apply three arguments:

- A matrix of SNP data as the input data
- Margin = 2 to indicate the function should run over every column (SNPs are in columns in this matrix)
- A pre-designed function that calls the SNP coefficient from a glm with a set number of sources.

The value from each apply() will thus be a vector of SNP coefficients.

3.3 Assessing a GWAS

According to [53], linkage disequilibrium (LD) can be attributed to "linkages between a marker and the causal mutation," but also to population structure. In other words, each SNP *may* have an impact on the affection status variable, or identified linkages may be due to spurious correlations. Many tools are used to assess a GWAS:

- **Manhattan plots** show the strength of linkage disequilibrium, as in the example in figure 6a, and also show locations of significantly associated SNPs, which can be clustered together or scattered across the genome.
- Used to quantify the extent of the bulk inflation and the excess false positive rate ([9], the genomic inflation factor, λ, compares the median of a test statistic with the expected median ([3]), and therefore can be calculated many ways ([26]), including an estimation using Regression Analysis. From [54], one common approach appears to be to convert the p-values of the data to a χ^2 distribution, then to compare with the median of a standard χ^2 with the same degrees of freedom. According to [26],

$E(\lambda) = 1$, and values of λ greater than one suggest some systemic bias may remain to be corrected in the data. The formula for the genomic inflation factor is very simply:

$$\lambda = \frac{median(observed)}{\chi^2_v(0.5)}$$

where v is the degrees of freedom in the data, and $observed$ is a χ^2 test statistic, for example, z^2 where z are Normal quantiles from the data. λ is commonly calculated as the median of the χ^2 test statistic divided by 0.454 cited for $v = 1$.

- The genomic inflation factor scales with size. Thus, λ_{1000} is a rescaling of λ with 1000 cases and 1000 controls, often used in reporting ([15]). Leveraging the χ^2 distribution, the conversion formula from [3] is:

$$\lambda_{1000} = 1 + (\lambda_{observed} - 1) \times (\frac{1}{n_{cases}} + \frac{1}{n_{controls}}) \div (\frac{1}{n_{cases_{new}}} + \frac{1}{n_{controls_{new}}})$$

  and since the new numbers are both 1000, this simplifies to

$$\lambda_{1000} = 1 + (\lambda_{observed} - 1) \times (\frac{1}{n_{cases}} + \frac{1}{n_{controls}}) \times 500$$

- **QQ plots** are commonly used to check how closely the quantiles of the data resemble their theoretical counterparts; in a sense, a QQplot is a visual extension of λ from the median to multiple quantiles. See an example in figure 6b.

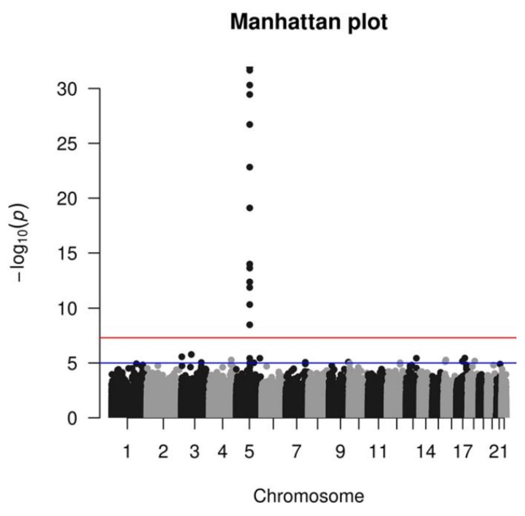
- For an assessment method free from assumptions with regards to the distribution of the genomic data, consider the sum of SNP coefficients, as applied in this project. See sections 4.1 and 4.7 for further details. In short, the simulations in this project ascribe no association between the affection status variable and any of the SNPs; therefore, all LD will be the result of systemic bias. Therefore, the best GLM will be the one that most reduces the coefficient inflation for all SNPs. By extension, the best dimensionality reduction method will produce GLMs with lower sums of SNP coefficients using fewer components.

4 Main Method and Example Code

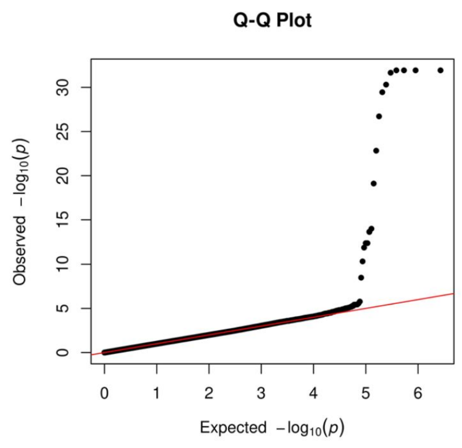
There are many communalities among all the simulations, so the briefest of them, modelling with a maximum of four sources, is displayed here to demonstrate the process, and segments of the rest of the simulations can be found section 6.

4.1 Data Preparation

The following code was used to read in and clean the data for multiple simulations.



(a) This Manhattan plot illustrates how many of the SNPs highly associated with the affection status variable are at the same locus, a phenomenon called linkage disequilibrium. Their common location indicates the SNPs may be part of the same gene, but are they all responsible for the phenotype with which they are associated?



(b) As in any regression model, a GWAS QQ-plot compares the distribution of the data to the expected distribution.

Figure 6: From [11], both of these diagrams exemplify common graphical outputs of a GWAS, intended to help select “candidate genes using LD extent” [53].

4.1.1 Read In the SNPs data

This stage only needed to occur twice; once to obtain and clean the original read1 file, and again, when the read1 file had degraded.

```
> setwd("C:/Users/user/Desktop/PROJECT/PCA_FA_ICA_code")
> test<-ped2geno("hapmap3_r2_b36_fwd.consensus.qc.poly.ped")

  - number of detected individuals: 1184
  - number of detected loci:    1440616
> #This took many minutes, but reduced the file size from
> # 6.6 million KB for the .ped
> #to 1.6 million KB for the .geno
> read1<-read.geno("hapmap3_r2_b36_fwd.consensus.qc.poly.geno")
Read 1440616 items
> save.image("hapmap3_full_code.R")
```

4.1.2 Clean the SNPs data so that dimensionality reduction methods will apply

This first process ensures that each column of SNPs is in the correct format. The SNPs should have been read in as 0, 1, or 2 (indicating the number of alleles in the polymorphic alternative), but sometimes there are columns that represent values other than SNPs (for example, a code indicating to which family the sample belonged), and possible entry errors.

```
    > setwd("C:/Users/user/Desktop/PROJECT/PCA_FA_ICA_code")
> load("hapmap3_full_code.R")
> max.col <-NA
> for(i in 1:ncol(read1)) max.col[i]<-max(read1[,i])
> clean1<-read1[,max.col==2]
> range(clean1) #confirms the range is now 0 to 2
[1] 0 2
> #to save space, periodically remove now-cleaned data frames
> rm(test, read1, max.col)
```

The next stage removes **monomorphic** SNPs from the dataset; since we are looking for causal SNPs for a polymorphic condition, and working to identify population structure (which summarises or explains the variety in a population), we don't need any SNPs with a variance of 0 in our study.

```
    > var.col <- NA
> for (i in 1:ncol(clean1))
+   var.col[i] = var(clean1[,i])
> clean2 <- clean1[,var.col!=0]
> rm(clean1, var.col)
```

Finally, in the following lines of code, you will see two steps accomplished:

- Identify pair-wise correlations greater than 0.9, and remove the variables from each pair with larger mean absolute correlation. findCorrelation() comes from the *caret* package ([30]). This step enables the dimensionaltiy reduction methods to work efficiently without noisy SNPs to bias the eigenvectors of PCA or mixing matrices of FA and ICA.
- Limit p to 12000 SNP columns; 12000 was chosen after experimentation demonstrated that any more would significantly slow or break the dimensionality reduction functions that need to be applied later. The selection of which SNPs to use is arbitrary, since the affection status variable in this project is artificially assigned to each individual; for repeatability, see repeat simulations on different subsets of SNPs in section 6, and the Discussion, Section 5.

```
> corr.col <- findCorrelation(cor(clean2[,c(1:12000)]), cutoff=0.9)
>
> clean3 <- clean2[,c(1:12000)]
> clean3 <- clean3[,-corr.col]
>
> rm(clean2, corr.col,i)
```

4.1.3 Assign affection status variable

```
>#Randomly Assigning Disease Status
> disease.index <- sample(1:nrow(clean3), nrow(clean3)/2, replace=FALSE)
> disease.status <- rep(0,nrow(clean3))
> for (i in disease.index)
+   disease.status[i] = 1
> mock_disease_data <- cbind(disease.status, clean3)
> mock_disease_data[1:10,1:5] #just to check it looks right.
      disease.status
 [1,]               1 2 2 2 2
 [2,]               0 1 2 0 2
 [3,]               1 2 2 2 2
 [4,]               0 2 2 1 2
 [5,]               1 2 2 0 2
 [6,]               0 2 2 1 2
 [7,]               1 2 2 1 2
 [8,]               0 2 2 2 2
 [9,]               0 2 2 1 2
[10,]               0 2 2 0 2
> dim(mock_disease_data) #1184 samples, one affection status variable and
    9170 SNPs  remaining after cleaning
[1] 1184 9171
> sum(mock_disease_data[,1]==1) #592 out of 1184 is roughly half the
    population
[1] 592
> rm(clean3, disease.index, disease.status, i)
>save.image("hapmap3_full_code.R")
```

At this stage, the coding environment contains just one large matrix with 1184 samples, one affection status variable, and 9170 SNPs. Using the rm() function, the coding environment is kept sparse as a matter of course, to ensure that the available data is sufficient for the project's purposes, while also conserving computer memory. Using the save.image() function enables quick reengagement with the project from this point using the load() function.

In section 4.5 we will add different forms of population structure to the large data frame, in order to set up different model structures for our GLM experiments. First, we need to generate the sources that represent population structure.

4.2 PCA code

4.2.1 PCA code Summary

In this simulation, SNPs 1:1149 were used because the number of SNPs analysed for dimensionality reduction was limited by the capacity of the factanal() function, identified through trial and error.

```
> pca2<-principal(mock_disease_data[,2:1150],nfactors=4,scores=TRUE)
'
The determinant of the smoothed correlation was zero.
This means the objective function is not defined.
Chi square is based upon observed residuals.
The determinant of the smoothed correlation was zero.
This means the objective function is not defined for the null model either.
The Chi square is thus based upon observed correlations.
'
#still slow and warning still appeared, but scores not inhibited
> pc1 <- pca2$scores[,1]
> pc2 <- pca2$scores[,2]
> pc3 <- pca2$scores[,3]
> pc4 <- pca2$scores[,4]
> head(pc1)
[1]  0.13891117 -1.25444564 -1.42132360
[4] -0.04916858 -1.29381648 -0.09684196
> length(pc1)
[1] 1184
```

4.3 FA code

For various reasons[4] the standard approach to selecting the best number of factors is dropped, in favour of running FA for q-factor models from $q = 1$ to 4, leaving the selection of the best population structure scores to later in this project.

[4]Namely, the goal of each simulation is to compare the dimensionality reduction methods to each other, rather than to follow the standard hueristics for selecting the best model from all options within a particular dimensionality reduction method. Just as the cumulative variance hueristic for PCA was not prioritised, neither is they hypothesis testing method for FA. To see an earlier attempt to select q based on the hypothesis testing method for FA, see section 7.

The limit on the number of SNPs that allowed PCA to be approximated with principal() produced an error for FA with factanal(), and so the whole dataset was further reduced.

```
> fac1<-factanal(mock_disease_data[,2:1184],1,scores="regression")
'
Error in solve.default(cv) :
  system is computationally singular: reciprocal condition number = 9.48829
    e-19
'
> fac1<-factanal(mock_disease_data[,2:1150],1,scores="regression")
#no error this time
```

Like principal(), the function factanal() took between five seconds and five minutes to run for each model.

```
> fac2<-factanal(mock_disease_data[,2:1150],2,scores="regression")
> fac3<-factanal(mock_disease_data[,2:1150],3,scores="regression")
> fac4<-factanal(mock_disease_data[,2:1150],4,scores="regression")
```

The following confirms that the q^{th}-factor score is different for different q-factor models; in this case, the 1st vector of scores in fac1 does not match the 1st vector of scores in fac2:

```
> head(fac1$scores,4)
         Factor1
[1,]  0.11768477
[2,] -1.19414731
[3,] -1.36872670
[4,] -0.05526246
> head(fac2$scores,4)
         Factor1     Factor2
[1,]  0.19324579  1.12171120
[2,] -1.21260131 -0.08005193
[3,] -1.37466522  0.13351712
[4,] -0.01890914  0.54636965
```

Thus, generating and keeping track of scores in different model structures requires extra care in the case of FA:

```
> fa1.1 <- fac1$scores[,1]
> fa2.1 <- fac2$scores[,1]
> fa2.2 <- fac2$scores[,2]
> fa3.1 <- fac3$scores[,1]
> fa3.2 <- fac3$scores[,2]
> fa3.3 <- fac3$scores[,3]
> fa4.1 <- fac4$scores[,1]
> fa4.2 <- fac4$scores[,2]
> fa4.3 <- fac4$scores[,3]
> fa4.4 <- fac4$scores[,4]
```

4.4 ICA code

In contrast to principal() and factanal(), fastICA() ran very quickly with no errors; this is due to the apply()-like design of the fastICA() function ([51], but also reflects how ICA is designed to be more robust to high dimensional data. In earlier experiments ([19]), it was noteworthy that fastICA() would even work on highly correlated data, while factanal() would not; the findCorrelation() step in 4.1.2, while advisable for all three dimensionality reduction methods, is only *required* by factanal().

```
> ica<-fastICA(mock_disease_data[,2:1150],4)
```

Below those components are saved as scores:

```
> ic1 <- ica$S[,1]
> ic2 <- ica$S[,2]
> ic3 <- ica$S[,3]
> ic4 <- ica$S[,4]
```

4.5 Constructing the full data frame

The glm() function needs to draw from a single data frame for its predictor and response variables. Hence, we add the population structure terms and appropriate labels as follows:

```
##Create data frame with pop structure sources
#dim redux has only occurred for the first 1149 SNPs.
#this experiment could be conveniently repeated for the next 1100 or so.

dat <- as.data.frame(mock_disease_data[,2:1150]) #this is the first 1149
    SNPs, no disease (i.e. inputs)
snp_names <- as.character(1:1149)
colnames(dat) <- snp_names
fix(dat) #can use this to quickly check if column names are in place.

dis <- mock_disease_data[,1]


###In order to run glm on a single data frame,
'I need to add the disease status and scores (above)
to the data frame called dat, with appropriate column names.'
```

```

data_full <- cbind(dis,dat,pc1,pc2,pc3,pc4,ic1,ic2,ic3,ic4,fa1.1,fa2.1,fa2
    .2,fa3.1,fa3.2,fa3.3,fa4.1,fa4.2,fa4.3,fa4.4)
```

4.6 GLM code

In practise, we use the apply() function to quickly run GLMs and total the SNP coefficients for each model structure; the following is merely an example of a single SNP GLM, run with the third SNP as the predictor.

```
> #Model Structure 0 : no corrective components
> glm0.1<-glm(dis ~ dat$"3", data = data_full, family = binomial)
> glm0.1$coefficients
'
(Intercept)      dat$"3"
 0.04594174 -0.03833276
 '
```

For all simulations, I performed the following steps for up to q sources per dimensionality reduction method:

- Write a function that runs any SNP through a particular model structure, and saves the coefficient of the SNP, which is the second coefficient.
- Use apply() to run that function over all SNPs. From [48], note that Margin = 2 indicates we are running each function of the columns of *dat* (i.e. the SNPs), rather than the rows (i.e. the samples).
- Calculate the sum of the coefficients of the SNPs for each model structure; these sums will be used in section 4.7.

```
#this function returns SNP coefficients for the model structure with no
    population structure
coef_glm0 <- function(x) coef(glm(dis ~ x, data = data_full, family =
    binomial))[2]

coef_glm0(dat$"3")

#apply for every column (MARGIN = 2 for columns)
coef0_list <- apply(dat, MARGIN = 2, coef_glm0)

sum0 <- sum(abs(coef0_list))
#sum0 will give sum of coefficients of SNPs when not controlled with any
    pop struc (i.e. glm0)
```

```


#Write the functions...
#GLMs with PCs as pop struc:
coef_glm1.1 <- function(x) coef(glm(dis ~ x + pc1,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.2 <- function(x) coef(glm(dis ~ x + pc1 + pc2,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.3 <- function(x) coef(glm(dis ~ x + pc1 + pc2 + pc3,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.4 <- function(x) coef(glm(dis ~ x + pc1 + pc2 + pc3 + pc4,
                                    data = data_full, family = binomial))
    [2]
#GLMs with Factors as pop struc:
coef_glm2.1 <- function(x) coef(glm(dis ~ x + fa1.1,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.2 <- function(x) coef(glm(dis ~ x + fa2.1 + fa2.2,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.3 <- function(x) coef(glm(dis ~ x + fa3.1 + fa3.2 + fa3.3,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.4 <- function(x) coef(glm(dis ~ x + fa4.1 + fa4.2 + fa4.3 + fa4
    .4,
                                    data = data_full, family = binomial))
    [2]

#GLMs with ICs as pop struc:
coef_glm3.1 <- function(x) coef(glm(dis ~ x + ic1,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.2 <- function(x) coef(glm(dis ~ x + ic1 + ic2,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.3 <- function(x) coef(glm(dis ~ x + ic1 + ic2 + ic3,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.4 <- function(x) coef(glm(dis ~ x + ic1 + ic2 + ic3 + ic4,
                                              data = data_full, family = binomial))
    [2]


#Use apply() on each function and calculate sum of SNP coefficients

#PCA:
coef1.1_list <- apply(dat, MARGIN = 2, coef_glm1.1)
sum1.1 <- sum(abs(coef1.1_list))

coef1.2_list <- apply(dat, MARGIN = 2, coef_glm1.2)
sum1.2 <- sum(abs(coef1.2_list))

coef1.3_list <- apply(dat, MARGIN =2, coef_glm1.3)
sum1.3 <- sum(abs(coef1.3_list))

coef1.4_list <- apply(dat, MARGIN =2, coef_glm1.4)
sum1.4 <- sum(abs(coef1.4_list))

#FA:
coef2.1_list <- apply(dat, MARGIN =2, coef_glm2.1)
sum2.1 <- sum(abs(coef2.1_list))

coef2.2_list <- apply(dat, MARGIN =2, coef_glm2.2)
sum2.2 <- sum(abs(coef2.2_list))

coef2.3_list <- apply(dat, MARGIN =2, coef_glm2.3)
sum2.3 <- sum(abs(coef2.3_list))

coef2.4_list <- apply(dat, MARGIN =2, coef_glm2.4)
sum2.4 <- sum(abs(coef2.4_list))

#ICA:
coef3.1_list <- apply(dat, MARGIN =2, coef_glm3.1)
sum3.1 <- sum(abs(coef3.1_list))

coef3.2_list <- apply(dat, MARGIN =2, coef_glm3.2)
sum3.2 <- sum(abs(coef3.2_list))

coef3.3_list <- apply(dat, MARGIN =2, coef_glm3.3)
sum3.3 <- sum(abs(coef3.3_list))

coef3.4_list <- apply(dat, MARGIN =2, coef_glm3.4)
```

```r
sum3.4 <- sum(abs(coef3.4_list))
```

4.7 Assessing Quality of Corrective Terms

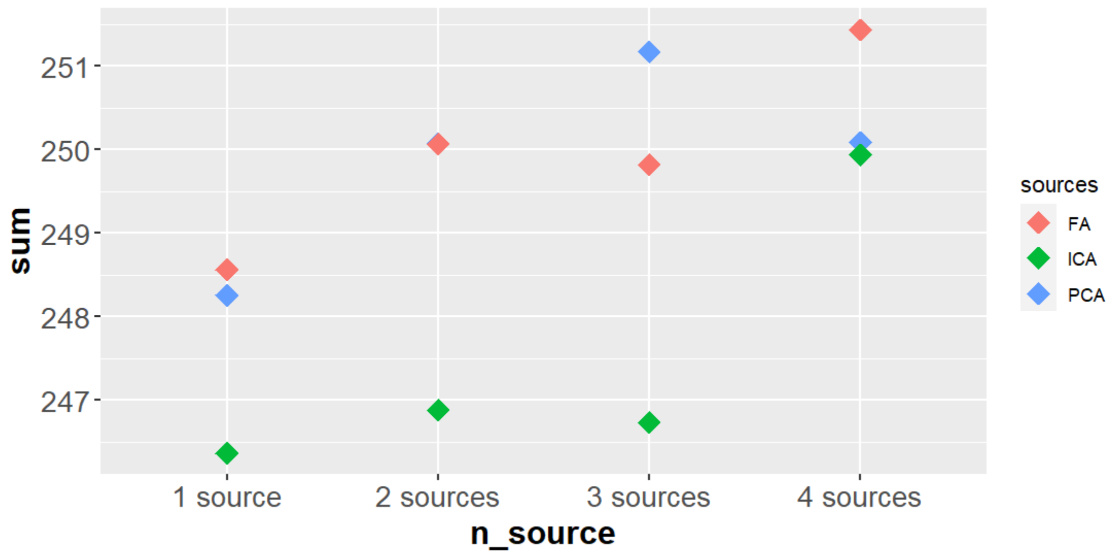
Once the sums have been calculated for each model structure in section 4.6, we seek to identify the "best" model structure as that which most reduces the sum of SNP coefficients[5]. Due to the random assignment of our disease, no SNPs are actually associated with the disease, so all coefficients should approach zero. However, applying parsimony, we also want to find the dimensionality reduction method that allows us to find the smallest sum of SNP coefficients using the *fewest* number of sources.

```r
#mini table- we should see the SNP coef
#sums descend with the addition of each pc

n_sources <- c("1 source", "2 sources", "3 sources", "4 sources")
PC_sums1 <- c(sum1.1,sum1.2,sum1.3,sum1.4)
FA_sums1 <- c(sum2.1, sum2.2, sum2.3, sum2.4)
IC_sums1 <- c(sum3.1, sum3.2, sum3.3, sum3.4)

display_results <- data.frame(n_population_structure_terms = n_sources,
                              PC_SNP_coef_total = PC_sums1,
                              FA_SNP_coef_total = FA_sums1,
                              IC_SNP_coef_total = IC_sums1)
> display_results
'
  n_population_structure_terms PC_SNP_coef_total FA_SNP_coef_total IC_SNP_
    coef_total
1                     1 source          248.2595          248.5643
     246.3642
2                    2 sources          250.0744          250.0708
     246.8843
3                    3 sources          251.1716          249.8228
     246.7326
4                    4 sources          250.0860          251.4368
     249.9387
'
#ICA shows better sums on average,
#but none of the sums decreases monotonically
#I expect we have not yet found the best number of sources
#for population structure.
```

[5]Before the methodology crystallised, I carried out other exploratory analyses of GLMs of SNP data using confusion tables. You can see an example of this work in section 7 .

```r


###A quick visual comparison of coefficient sums

bar_stac <- data.frame(sources=rep(c("PCA","FA","ICA"), each = 4),
                       n_source=rep(n_sources, times = 3),
                       sum = c(PC_sums1 , FA_sums1, IC_sums1))

ggplot(bar_stac, aes(col=sources, y=sum, x=n_source)) +
  geom_point(shape = 18, size = 5) +
  theme(axis.text=element_text(size=14),axis.title=element_text(size=16,
    face="bold"))
```

Figure 7: As you can see, the results were not striking, though ICA took an initial lead in reduction. Back to section 4.7.



According to display_results above, there was a general upward trend with the addition of each source, especially for FA. Especially visible in figure 7, it appears as though corrective terms derrived through ICA result in the lowest coefficient sums. Based on this small simulation, it could be the case that while ICA produces the least amount of inflation, the addition of each source increased inflation of the SNP coefficients rather than decreased it; this would run counter to the expectation that population structure terms correct for spurious correlations between each SNP and the affection status variable.

However, the initial use of four sources was arbitrary, allowing for comparisons between PCA, FA, and ICA, while succinctly developing a coding structure for all simulations. Thus, another possible explanation for the poor performance of all the population structure terms is that the inflation as a function of the number of sources is nonlinear. In other words, like the sum of coefficients, the mean squared error (MSE) is a success criterion to be minimised to identify ideal regression models. For many regression methods, the MSE plotted against the number of predictors in the model results in a U-shaped [25]; is it the case that the addition of more sources would show a dip in the sum of SNP coefficients?

After the results of the max-four-source model structures proved inconclusive, it seemed as though larger model structures[6] would provide more discernible results. The simulations following the example in this section were all run with more than four sources, and you can see their results in section 6 as well as a reference table in the Discussion, section 5.1.

5 Discussion

5.1 Conclusions of Analysis

After twelve simulations on data of over 1000 people with varying numbers of sources, numbers of SNPs, disease prevalences, and nucleotide locations with the genome, two patterns appear to emerge.

- ICA consistently produces the lowest sum of SNP coefficients, and FA generally produces the highest; under the assumption that a lower sum of SNPs indicates that the arguments of the GLM better accounted for variance in the data, this would suggest that ICA is a better dimensionality reduction method than PCA, for the purposes of GWAS.

- **However**, the sum of SNP coefficients (almost monotonically!) increased with the addition of each source, suggesting **no** dimensionality reduction method actually *improved* the GLM; this pattern calls to mind the bias-variance tradeoff (Pages 217-218 of [25]): as a model's flexibility increases, its bias decreases, but its variance increases. Perhaps the sources in my simulations introduced more variance without reducing bias because they were drawn from data that was too small to justify population structure terms in the first place. Nevertheless, the results of these simulations remind us to approach any dimensionality reduction method (even the conveniently Plink-embedded PCA) with caution.

The table below displays the different simulations and their features:

[6]For full genome studies, along the lines of 20 sources are used, as suggested in [32].

Section	Figure	Affect Frequency	N. of SNPs	Location of SNPs	Maximum Sources
4.7	7	50%	1150	1:15000	4
6.1.6	8	50%	1150	1:15000	16
6.2	9	50%	650	1:15000	7
6.3.1	10	10%	800	15000:25000	7
6.3.2	11	10%	800	285000:295000	7
6.3.3	12	30%	800	285000:295000	7
6.3.4	13	10%	800	15000:25000	7
6.3.5	14	30%	940	15000:25000	7
6.4	15	20%	2000	65000:75000	7
6.4	16	20%	2000	15000:25000	7
6.4	17	20%	2000	285000:295000	7
6.4	18	20%	2000	115000:125000	7

5.2 Project Achievements

This project set out to explore one central question: between PCA, FA, and ICA, which dimensionality reduction method produces the best population structure sources for a GWAS? I ran twelve simulations with different parameters to fully explore the question, and to address possible explanations for the unexpected effect of population structure terms in those simulations. Despite the poor performance most likely caused by limitations on the size of data I could process, there does appear to be a clear winner. The table below displays which dimensionality reduction method generally produced the lowest sum of SNP coefficients for each simulation, and which produced the highest. A $*$ in the rightmost column indicates the rare occasions that the best model had less coefficient inflation than the source-free model.

Section	Figure	Best Method	Worst Method	Better than sum0
4.7	7	ICA	FA	
6.1.6	8	ICA	FA	
6.2	9	ICA	FA	
6.3.1	10	ICA	FA	
6.3.2	11	ICA	FA	
6.3.3	12	ICA	FA	$*$
6.3.4	13	PCA	FA	
6.3.5	14	ICA	FA	$*$
6.4	15	ICA	PCA (FA not an option)	
6.4	16	ICA	PCA (FA not an option)	
6.4	17	ICA	PCA (FA not an option)	
6.4	18	ICA	PCA (FA not an option)	

In pursuit of answering the central question, as well as exploring new domains of data science, I refreshed and developed:

- an introduction to data processing with Plink and CMD Prompt
- my knowledge of genomics
- algorithmic thinking through the use of apply()
- a better understanding of data cleaning and ensuring data is fit for purpose
- troubleshooting skills for code development, research, and data analysis, particularly of genomic data
- best practises for writing clear, functional, transferable code

5.3 Possibilities for Further Research

Two new questions come directly from the results of my simulations:

- Why were the simulations in sections 6.3.3 and 6.3.5 the only successful demonstrations of population structure *reducing* coefficient inflation? Is it a coincidence that those were the only two simulations to use a disease prevalence of 30%, or does that frequency hit some "sweet spot" for variance of the response variable, similar to the Rule of Three ([27]?
- Why does ICA work better? In the end, I did not have time to derive a means of checking the distribution of the genome data against a Laplacian; it seems clear from most qqplots of genome data (many of which look like figure 6b ([34], [13],

[11], [54], [53])), that the tails of these distributions don't fit the Normal Quantiles. While this doesn't guarantee the true distribution is Laplacian, I don't imagine it would be difficult to derive a similar qqplot comparing the genome data to the supergaussian ([24]).

For decades, population structure has been a useful tool in genomics for explaining the inflation of test statistics [8], so why would my simulations, which are at least 6 times smaller, show the opposite effect? I believe the resemblance of my simulations to true GWAS was significantly curtailed by the size of my simulations; therefore:

- One avenue of interest would be an examination of the scalability of dimensionality reduction methods from one size of data to data of dimensions many factors of ten greater or smaller. There are already well-defined methods for scaling the genomic inflation factor, but there does not appear to be one consistent metric of 'inflation' for associated data. Each dimensionality reduction method has its own proposed methods for determining the suitability of its sources, mostly designed to select the best number of sources with parsimony. Thus, while answers on source-identification-method scalability might seem within reach, I believe the question opens up possibilities for developing new, more consistent metrics and methodologies.
- Another approach would be to run the same experiment of this project on larger and larger subsets of SNPs[7]; various solutions to the arduous size of genomic data are rife in genomics, as exemplified in [3], which explains one solution for identifying causal SNPs spread out across multiple chromosomes: "Parallelization on a multi-node cluster can be achieved by splitting jobs up across chromosomes and into different sample subsets but keeping cases and controls together to avoid differential bias." Because factanal() in particular[8], but all the dimensionality reduction functions in R, struggled with even 1000 parameters, for a re-creation of these simulations on larger datasets, it might be necessary to develop algorithms for PCA, FA, and ICA that could operate on these larger subsets.

It would also be interesting to run simulations against an alternate model in which the disease is constructed to be associated with a small subset of the SNPs, as described in section 7.2; such an approach would facilitate the use of alternative measures of success, for instance, the prediction accuracy, as demonstrated in section 7.3, which would apply cross

[7]After the submission of this paper, I may attempt this using cloud computing.

[8]see section 6.4 for some larger simulations that didn't involve FA

validation. How accurately can we predict the presence of a condition, given someone's genes?

On the topic of real world applications, the use of GWAS is likely to become far reaching. Not only is genomics already being used to predict some diseases ([7]), but the potential to precisely identify phenotypical traits from a person's genome could have massive implications for positive drug outcomes[9]. Though it may sound like science fiction[10], the use of GWAS to better understand what human beings are made of (including all the genomes we use that *don't* reside in our nuclei ([22])) will become increasingly accurate and powerful as we collect more and more data about ourselves.

Thus, even the ethics of genomics and analytics should be held under the microscope ([49]). At what point, under the auspicious guise of societal security or psychological research, will profiling of our online data enmesh with our genomes ([1]), rendering ever-more-detailed versions of our identities commodities? What efforts are currently made to maintain anonymity in genome data?

6 Further Simulations

6.1 Simulation: 16 sources

This section contains the simulation run on the largest number of sources. Initially intended to run on up to 25 sources (inspired by [32]), this simulation began with PCA, and the maximum number of sources was adjusted to accommodate FA (section 6.1.2).

6.1.1 PCA for 25 sources

```r
##Population Structure (Dimensionality Reduction)
#PCA
pca25<-principal(mock_disease_data[,2:1150],nfactors=25,scores=TRUE)
#plot(test.pca$scores[,1],test.pca$scores[,2], col = 4, pch = 15)

pc1 <- pca25$scores[,1]
pc2 <- pca25$scores[,2]
pc3 <- pca25$scores[,3]
pc4 <- pca25$scores[,4]
pc5 <- pca25$scores[,5]
pc6 <- pca25$scores[,6]
pc7 <- pca25$scores[,7]
```

[9]For instance, see my reflections on how we could better select contraception methods at [21], inspired by [23].

[10]In the latest James Bond movie, compounds can be tailor-made to assassinate genetically distinct individuals! [41]

```r
pc8 <- pca25$scores[,8]
pc9 <- pca25$scores[,9]
pc10 <- pca25$scores[,10]
pc11 <- pca25$scores[,11]
pc12 <- pca25$scores[,12]
pc13 <- pca25$scores[,13]
pc14 <- pca25$scores[,14]
pc15 <- pca25$scores[,15]
pc16 <- pca25$scores[,16]
pc17 <- pca25$scores[,17]
pc18 <- pca25$scores[,18]
pc19 <- pca25$scores[,19]
pc20 <- pca25$scores[,20]
pc21 <- pca25$scores[,21]
pc22 <- pca25$scores[,22]
pc23 <- pca25$scores[,23]
pc24 <- pca25$scores[,24]
pc25 <- pca25$scores[,25]

pca_sources <- data.frame(pc1, pc2, pc3, pc4, pc5,
                          pc6, pc7, pc8, pc9, pc10,
                          pc11, pc12, pc13, pc14, pc15,
                          pc16, pc17, pc18, pc19, pc20,
                          pc21, pc22, pc23, pc24, pc25)
```

6.1.2 FA for 16 sources

Generating $q = 25$ sources for PCA is fairly straightforward; however, as each run of factanal() took an increasing amount of computing power and memory, I instead ran an arbitrary selection of factor analyses for $q \in [6, 25]$ in order to estimate any "turning point" in the bias-variance trade-off as q increased; as you will see in the sum of SNPs in figure 8, this adjustment proved fruitful, and saved time and computer memory in the process, since the turning point appeared early.

```r
#FA #since each q-factor model is separately generated...
fac1<-factanal(mock_disease_data[,2:1150],1,scores="regression")
fac2<-factanal(mock_disease_data[,2:1150],2,scores="regression")
fac3<-factanal(mock_disease_data[,2:1150],3,scores="regression")
fac4<-factanal(mock_disease_data[,2:1150],4,scores="regression")
fac5<-factanal(mock_disease_data[,2:1150],5,scores="regression")
fac6<-factanal(mock_disease_data[,2:1150],6,scores="regression")
  # honestly it became painful watching the computer try to run EVERY
    factor analysis,
  # so I skipped a few at this stage....
#fac7<-factanal(mock_disease_data[,2:1150],7,scores="regression")
fac8<-factanal(mock_disease_data[,2:1150],8,scores="regression")
#fac9<-factanal(mock_disease_data[,2:1150],9,scores="regression")
fac10<-factanal(mock_disease_data[,2:1150],10,scores="regression")
#fac11<-factanal(mock_disease_data[,2:1150],11,scores="regression")
#fac12<-factanal(mock_disease_data[,2:1150],12,scores="regression")
#fac13<-factanal(mock_disease_data[,2:1150],13,scores="regression")
#fac14<-factanal(mock_disease_data[,2:1150],14,scores="regression")
#fac15<-factanal(mock_disease_data[,2:1150],15,scores="regression")
fac16<-factanal(mock_disease_data[,2:1150],16,scores="regression")
#fac17<-factanal(mock_disease_data[,2:1150],17,scores="regression")
#fac18<-factanal(mock_disease_data[,2:1150],18,scores="regression")
#fac19<-factanal(mock_disease_data[,2:1150],19,scores="regression")
#fac20<-factanal(mock_disease_data[,2:1150],20,scores="regression")
##fac21<-factanal(mock_disease_data[,2:1150],21,scores="regression")
#fac22<-factanal(mock_disease_data[,2:1150],22,scores="regression")
#fac23<-factanal(mock_disease_data[,2:1150],23,scores="regression")
#fac24<-factanal(mock_disease_data[,2:1150],24,scores="regression")
##fac25<-factanal(mock_disease_data[,2:1150],25,scores="regression")


fa_sources <- data.frame(fac1$scores,fac2$scores,fac3$scores,fac4$scores,
    fac5$scores,
                         fac6$scores,#fac7$scores,
                         fac8$scores,#fac9$scores,
                         fac10$scores,
                         #fac11$scores,fac12$scores,fac13$scores,fac14$
    scores,fac15$scores,
                         fac16$scores#,#fac17$scores,fac18$scores,fac19$
    scores,fac20$scores,
                        # fac21$scores,#fac22$scores,fac23$scores,fac24$
    scores,
                        # fac25$scores
                        )
names(fa_sources) <- c("fa1.1",
                       "fa2.1", "fa2.2",
                       "fa3.1","fa3.2", "fa3.3",
                       "fa4.1","fa4.2","fa4.3","fa4.4",
                       "fa5.1","fa5.2","fa5.3","fa5.4","fa5.5",
                       "fa6.1","fa6.2","fa6.3","fa6.4","fa6.5",
                       "fa6.6",
                       #"fa7.1","fa7.2","fa7.3","fa7.4","fa7.5",
                       #"fa7.6","fa7.7",
                       "fa8.1","fa8.2","fa8.3","fa8.4","fa8.5",
                       "fa8.6","fa8.7","fa8.8",
                       #"fa9.1","fa9.2","fa9.3","fa9.4","fa9.5",
                       #"fa9.6","fa9.7","fa9.8","fa9.9",
                       "fa10.1","fa10.2","fa10.3","fa10.4","fa10.5",
                       "fa10.6","fa10.7","fa10.8","fa10.9","fa10.10",
                       #
                       "fa16.1","fa16.2","fa16.3","fa16.4","fa16.5",
                       "fa16.6","fa16.7","fa16.8","fa16.9","fa16.10",
                       "fa16.11","fa16.12","fa16.13","fa16.14","fa16.15",
                       "fa16.16"#,
                       #
                      # "fa21.1","fa21.2","fa21.3","fa21.4","fa21.5",
                      # "fa21.6","fa21.7","fa21.8","fa21.9","fa21.10",
                      # "fa21.11","fa21.12","fa21.13","fa21.14","fa21.15",
                      # "fa21.16","fa21.17","fa21.18","fa21.19","fa21.20",
                      # "fa21.21",
                       #
                      # "fa25.1","fa25.2","fa25.3","fa25.4","fa25.5",
                      # "fa25.6","fa25.7","fa25.8","fa25.9","fa25.10",
                      # "fa25.11","fa25.12","fa25.13","fa25.14","fa25.15",
                      # "fa25.16","fa25.17","fa25.18","fa25.19","fa25.20",
                      # "fa25.21","fa25.22","fa25.23","fa25.24","fa25.25"
                      )
```

6.1.3 ICA for 16 sources

As in the four-source code, fastICAI() ran quickly and smoothly. To save time, I adopted a different naming convention from section 4.

```
#ICA
ica<-fastICA(mock_disease_data[,2:1150],25)
#plot(test.ica$S, col = 6, pch = 17)

ica_sources <- data.frame(ica$S)
names(ica_scores) #so now I know to refer to ic's as X, i.e. X9 is the 9th
    independent component.
```

6.1.4 Data Frame for 16 sources

This process is nearly identical, save that, in order to reduce the amount of copy/pasting required to input every possible source into the data frame, I pre-framed them, as you saw, above, with names ‘pca_sources,’ ‘fa_sources,’ and ica_sources.’

```r
##Create data frame with pop structure sources
#dim redux has only occurred for the first 1149 SNPs.
#this experiment could be conveniently repeated for the next 1100 or so.

dat <- as.data.frame(mock_disease_data[,2:1150]) #this is the first 1149
    SNPs, no disease (i.e. inputs)
snp_names <- as.character(1:1149)
colnames(dat) <- snp_names
fix(dat) #can use this to quickly check if column names are in place.

dis <- mock_disease_data[,1]


###In order to run glm on a single data frame,
'I need to add the disease status and scores (above)
to the data frame called dat, with appropriate column names.'

data_full <- cbind(dis,dat,pca_sources,fa_sources, ica_sources)
data_full <- as.data.frame(data_full)
```

6.1.5 GLM with up to 16-sources

Here we set up each GLM generically, so that is can be run on every SNP, one at a time:

```r

'Single SNP GLMs with varying model structures'

#Model Structure 0 : no corrective components
glm0.1<-glm(dis ~ dat$"3", data = data_full, family = binomial)

glm(dis ~ dat$"3", data = data_full, family = binomial)$coefficients[2]
glm(dis ~ dat$"4", data = data_full, family = binomial)$coefficients[2]

scoef1 <- coef(glm(dis ~ dat$"1", data = data_full, family = binomial))[2]
scoef2 <- coef(glm(dis ~ dat$"2", data = data_full, family = binomial))[2]

coef_SNPs.0 <- c(scoef1,scoef2)

```

```
sum(coef_SNPs.0)


glm0_func <- function(x) glm(dis ~ x, data = data_full, family = binomial)

glm0_func(dat$"3") #works

####start here

#this function returns SNP coefficients for the model structure with no
    population structure
coef_glm0 <- function(x) coef(glm(dis ~ x, data = data_full, family =
    binomial))[2]

coef_glm0(dat$"3")

#apply for every column (MARGIN = 2 for columns)
coef0_list <- apply(dat, MARGIN = 2, coef_glm0)

sum0 <- sum(abs(coef0_list))
#sum0 will give sum of coefficients of SNPs when not controlled with any
    pop struc (i.e. glm0)

#model structure naming system:
#coef_glmA.B, coefA.B_list, and sumA.B
#A is 1 for PCA, 2 for FA, and 3 for ICA
#B is the number of added components/factors

#just recall what those term names are:
names(data_full[,1185:1255])
#faSIZE.FACTOR
#X12 = ic12
names(data_full[,1150:1185])
#pcCOMPONENT

##First I'll set up each function, for model structures where B=1, 2, 3, 4,
     5, 6, 8, 10, or 16

#A=1 for PCA:
coef_glm1.1 <- function(x) coef(glm(dis ~ x + pc1,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.2 <- function(x) coef(glm(dis ~ x + pc1+pc2,
                                    data = data_full, family = binomial))[2]
coef_glm1.3 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3,
                                    data = data_full, family = binomial))[2]
coef_glm1.4 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4,
                                    data = data_full, family = binomial))[2]
coef_glm1.5 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5,
                                    data = data_full, family = binomial))[2]
coef_glm1.6 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6,
                                    data = data_full, family = binomial))[2]
coef_glm1.8 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6+pc7+pc8,
                                    data = data_full, family = binomial))[2]
coef_glm1.10 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6+pc7+pc8+pc9+pc10,
                                    data = data_full, family = binomial))[2]
coef_glm1.16 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6+pc7+pc8+pc9+pc10+pc11+pc12+pc13+pc14+pc15+pc16,
                                    data = data_full, family = binomial))[2]

#A=2 for FA:
coef_glm2.1 <- function(x) coef(glm(dis ~ x + fa1.1,
                                    data = data_full, family = binomial))[2]
coef_glm2.2 <- function(x) coef(glm(dis ~ x + fa2.1+fa2.2,
                                    data = data_full, family = binomial))[2]
coef_glm2.3 <- function(x) coef(glm(dis ~ x + fa3.1+fa3.2+fa3.3,
                                    data = data_full, family = binomial))[2]
coef_glm2.4 <- function(x) coef(glm(dis ~ x + fa4.1+fa4.2+fa4.3+fa4.4,
                                    data = data_full, family = binomial))[2]
coef_glm2.5 <- function(x) coef(glm(dis ~ x + fa5.1+fa5.2+fa5.3+fa5.4+fa5.5,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.6 <- function(x) coef(glm(dis ~ x + fa6.1+fa6.2+fa6.3+fa6.4+fa6
    .5+fa6.6,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.8 <- function(x) coef(glm(dis ~ x + fa8.1+fa8.2+fa8.3+fa8.4+fa8
    .5+fa8.6+fa8.7+fa8.8,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.10 <- function(x) coef(glm(dis ~ x + fa10.1+fa10.2+fa10.3+fa10.4+
    fa10.5+fa10.6+fa10.7+fa10.8+fa10.9+fa10.10,
                                     data = data_full, family = binomial))
    [2]
coef_glm2.16 <- function(x) coef(glm(dis ~ x + fa16.1+fa16.2+fa16.3+fa16.4+
    fa16.5+fa16.6+fa16.7+fa16.8+fa16.9+fa16.10+fa16.11+fa16.12+fa16.13+fa16
    .14+fa16.15+fa16.16,
                                     data = data_full, family = binomial))
    [2]

#A=3 for ICA:
coef_glm3.1 <- function(x) coef(glm(dis ~ x + X1,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.2 <- function(x) coef(glm(dis ~ x + X1+X2,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.3 <- function(x) coef(glm(dis ~ x + X1+X2+X3,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.4 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.5 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.6 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.8 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6+X7+X8,
                                    data = data_full, family = binomial))
    [2]
```

```r
coef_glm3.10 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6+X7+X8+X9+
    X10,
                                    data = data_full, family = binomial))
    [2]
coef_glm3.16 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6+X7+X8+X9+
    X10+X11+X12+X13+X14+X15+X16,
                                    data = data_full, family = binomial))
    [2]
```

Next we apply the GLM for each model structure to all SNPs, and calculate SNP coefficient sums:

```r

#Apply and get sums for each model structure!
#PCA
coef1.1_list <- apply(dat, MARGIN = 2, coef_glm1.1)
sum1.1 <- sum(abs(coef1.1_list))
coef1.2_list <- apply(dat, MARGIN = 2, coef_glm1.2)
sum1.2 <- sum(abs(coef1.2_list))
coef1.3_list <- apply(dat, MARGIN = 2, coef_glm1.3)
sum1.3 <- sum(abs(coef1.3_list))
coef1.4_list <- apply(dat, MARGIN = 2, coef_glm1.4)
sum1.4 <- sum(abs(coef1.4_list))
coef1.5_list <- apply(dat, MARGIN = 2, coef_glm1.5)
sum1.5 <- sum(abs(coef1.5_list))
coef1.6_list <- apply(dat, MARGIN = 2, coef_glm1.6)
sum1.6 <- sum(abs(coef1.6_list))
coef1.8_list <- apply(dat, MARGIN = 2, coef_glm1.8)
sum1.8 <- sum(abs(coef1.8_list))
coef1.10_list <- apply(dat, MARGIN = 2, coef_glm1.10)
sum1.10 <- sum(abs(coef1.10_list))
coef1.16_list <- apply(dat, MARGIN = 2, coef_glm1.16)
sum1.16 <- sum(abs(coef1.16_list))

#FA
coef2.1_list <- apply(dat, MARGIN = 2, coef_glm2.1)
sum2.1 <- sum(abs(coef2.1_list))
coef2.2_list <- apply(dat, MARGIN = 2, coef_glm2.2)
sum2.2 <- sum(abs(coef2.2_list))
coef2.3_list <- apply(dat, MARGIN = 2, coef_glm2.3)
sum2.3 <- sum(abs(coef2.3_list))
coef2.4_list <- apply(dat, MARGIN = 2, coef_glm2.4)
sum2.4 <- sum(abs(coef2.4_list))
coef2.5_list <- apply(dat, MARGIN = 2, coef_glm2.5)
```

```r
sum2.5 <- sum(abs(coef2.5_list))
coef2.6_list <- apply(dat, MARGIN = 2, coef_glm2.6)
sum2.6 <- sum(abs(coef2.6_list))
coef2.8_list <- apply(dat, MARGIN = 2, coef_glm2.8)
sum2.8 <- sum(abs(coef2.8_list))
coef2.10_list <- apply(dat, MARGIN = 2, coef_glm2.10)
sum2.10 <- sum(abs(coef2.10_list))
coef2.16_list <- apply(dat, MARGIN = 2, coef_glm2.16)
sum2.16 <- sum(abs(coef2.16_list))

#ICA
coef3.1_list <- apply(dat, MARGIN = 2, coef_glm3.1)
sum3.1 <- sum(abs(coef3.1_list))
coef3.2_list <- apply(dat, MARGIN = 2, coef_glm3.2)
sum3.2 <- sum(abs(coef3.2_list))
coef3.3_list <- apply(dat, MARGIN = 2, coef_glm3.3)
sum3.3 <- sum(abs(coef3.3_list))
coef3.4_list <- apply(dat, MARGIN = 2, coef_glm3.4)
sum3.4 <- sum(abs(coef3.4_list))
coef3.5_list <- apply(dat, MARGIN = 2, coef_glm3.5)
sum3.5 <- sum(abs(coef3.5_list))
coef3.6_list <- apply(dat, MARGIN = 2, coef_glm3.6)
sum3.6 <- sum(abs(coef3.6_list))
coef3.8_list <- apply(dat, MARGIN = 2, coef_glm3.8)
sum3.8 <- sum(abs(coef3.8_list))
coef3.10_list <- apply(dat, MARGIN = 2, coef_glm3.10)
sum3.10 <- sum(abs(coef3.10_list))
coef3.16_list <- apply(dat, MARGIN = 2, coef_glm3.16)
sum3.16 <- sum(abs(coef3.16_list))
```

6.1.6 Comparison of results for up to 16 sources

First, a table:

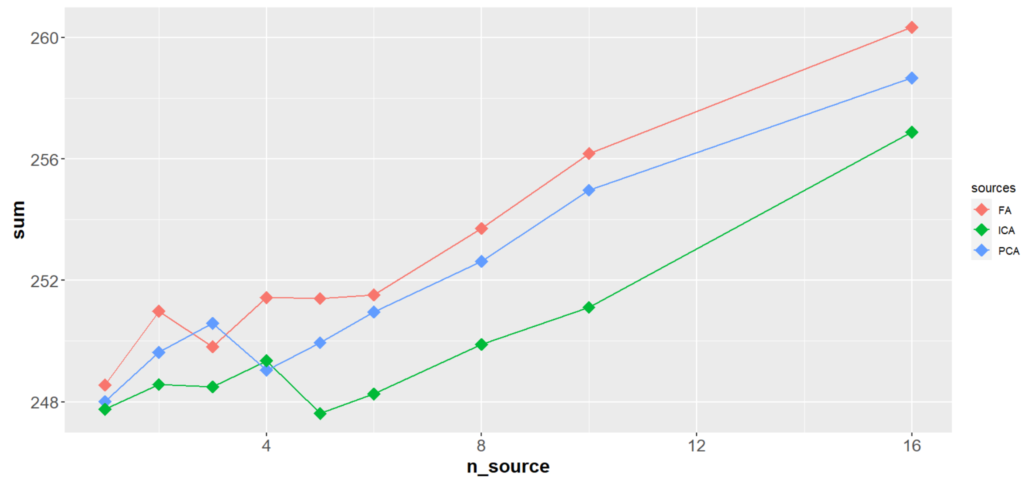
```r
#mini table- we should see the SNP coef
#sums descend with the addition of each pc

n_sources <- c(1, 2, 3, 4,5,6,8,10,16)
PC_sums <- c(sum1.1,sum1.2,sum1.3,sum1.4,sum1.5,sum1.6,sum1.8,sum1.10,sum1
    .16)
FA_sums <- c(sum2.1, sum2.2, sum2.3, sum2.4,sum2.5, sum2.6,sum2.8,sum2.10,
    sum2.16)
IC_sums <- c(sum3.1, sum3.2, sum3.3, sum3.4,sum3.5,sum3.6,sum3.8,sum3.10,
    sum3.16)
```

```

display_results <- data.frame(n_population_structure_terms = n_sources,
                              PC_SNP_coef_total = PC_sums,
                              FA_SNP_coef_total = FA_sums,
                              IC_SNP_coef_total = IC_sums)
> display_results
  n_population_structure_terms PC_SNP_coef_total FA_SNP_coef_total IC_SNP_
   coef_total
1                            1          248.0122          248.5643
    247.7565
2                            2          249.6374          250.9878
    248.5764
3                            3          250.5889          249.8228
    248.4955
4                            4          249.0452          251.4368
    249.3580
5                            5          249.9525          251.4080
    247.6166
6                            6          250.9628          251.5271
    248.2564
7                            8          252.6315          253.7145
    249.8910
8                           10          254.9730          256.1840
    251.1163
9                           16          258.6689          260.3405
    256.8875
```

Then, a plot, shown in figure 8, coded below:

```
###A quick visual comparison of coefficient sums

bar_stac <- data.frame(sources=rep(c("PCA","FA","ICA"), each = 9),
                       n_source=rep(n_sources, times = 3),
                       sum = c(PC_sums, FA_sums, IC_sums))

ggplot(bar_stac, aes(col=sources, y=sum, x=n_source)) +
  geom_point(shape = 18, size = 5) +
  #geom_line(aes(group = sources))+
  geom_smooth(aes(group = sources), method = "loess", se = FALSE)+
  theme(axis.text=element_text(size=14),axis.title=element_text(size=16,
   face="bold"))
```

Figure 8: Here is a rendering of the ggplot from section 6.1.6, demonstrating that coefficient inflation markedly increases after roughly 6 sources are incorporated into a GLM, and that ICA consistently produces better sources than the other two approaches. Back to section 6.1.6.



6.2 Simulation: different genome region, 7 sources

Could the stark ranking of ICA, then PCA, then FA, have been a fluke? Multiple simulations on a different subset of SNPs suggest not. The code in this section is identical to that in section 6.1, except the initial data input for dimensionality reduction and glm data frames is as shown below.

```
#Recall that there were far more SNPs we could use for dimensionality
    reduction...
> dim(mock_disease_data)
[1] 1184 9171
```

Based on this, I ran the experiment again on the next 650 SNPs[11]; this is a little over half the number of SNPs from the largest simulation. Additionally, since the graph in figure 8 showed dimensionality reduction methods beyond $q = 6$ didn't seem to improve the SNP coefficient total, to save memory and computing power, I limited the subsequent simulation to the first seven (just to be safe) sources.

[11]Aiming for another 1100, I gradually reduced this number until factanal() stopped returning the "system is computationally singular" error.

First, generating the population structure sources of this new subset of SNPs, starting with PCA[12]:

```r
##Population Structure (Dimensionality Reduction)
#PCA
pca7<-principal(mock_disease_data[,1150:1800],nfactors=7,scores=TRUE)

pc1 <- pca7$scores[,1]
pc2 <- pca7$scores[,2]
pc3 <- pca7$scores[,3]
pc4 <- pca7$scores[,4]
pc5 <- pca7$scores[,5]
pc6 <- pca7$scores[,6]
pc7 <- pca7$scores[,7]


pca_sources <- data.frame(pc1, pc2, pc3, pc4, pc5,
                          pc6, pc7)
```

Then FA:

```r
#FA 7 times since each q-factor model is separately generated...
fac1<-factanal(mock_disease_data[,1150:1800],1,scores="regression")
fac2<-factanal(mock_disease_data[,1150:1800],2,scores="regression")
fac3<-factanal(mock_disease_data[,1150:1800],3,scores="regression")
fac4<-factanal(mock_disease_data[,1150:1800],4,scores="regression")
fac5<-factanal(mock_disease_data[,1150:1800],5,scores="regression")
fac6<-factanal(mock_disease_data[,1150:1800],6,scores="regression")
fac7<-factanal(mock_disease_data[,1150:1800],7,scores="regression")

fa_sources <- data.frame(fac1$scores,fac2$scores,
                         fac3$scores,fac4$scores,fac5$scores,
                         fac6$scores,fac7$scores)

names(fa_sources) <- c("fa1.1",
                       "fa2.1", "fa2.2",
                       "fa3.1","fa3.2", "fa3.3",
                       "fa4.1","fa4.2","fa4.3","fa4.4",
                       "fa5.1","fa5.2","fa5.3","fa5.4","fa5.5",
                       "fa6.1","fa6.2","fa6.3","fa6.4","fa6.5",
                       "fa6.6",
```

[12]Running principal() returned a new warning which speaks to λ as a measure of fitness: “The determinant of the smoothed correlation was zero. This means the objective function is not defined for the null model either. The Chi square is thus based upon observed correlations.”

```
                              "fa7.1","fa7.2","fa7.3","fa7.4","fa7.5",
                              "fa7.6","fa7.7"
                              )
```

Then ICA:

```
#ICA
ica<-fastICA(mock_disease_data[,1150:1800],7)
#plot(test.ica$S, col = 6, pch = 17)

ica_sources <- data.frame(ica$S)
#names(ica_scores) #X5 instead of ic5

save.image("repeat_big.R")
```

Then setting up the data frame for the GLM:

```
##Create data frame with pop structure sources

dat <- as.data.frame(mock_disease_data[,1150:1800])
snp_names <- as.character(1149:1799)
colnames(dat) <- snp_names

#dis <- mock_disease_data[,1]
        #### already coded in from last time; no need to replace


data_full <- cbind(dis,dat,pca_sources,fa_sources,ica_sources)
data_full <- as.data.frame(data_full)
```

Then the GLM functions:

```

'Single SNP GLMs with varying model structures'

#this function returns SNP coefficients for the model structure with no
    population structure
coef_glm0 <- function(x) coef(glm(dis ~ x, data = data_full, family =
    binomial))[2]

#apply for every column (MARGIN = 2 for columns)
coef0_list <- apply(dat, MARGIN = 2, coef_glm0)

sum0 <- sum(abs(coef0_list))
sum0 # for SNPs 1:1149 it was 247.0496
```

```
    # this time, for SNPs 1149:2249, it's 128.6101, since there are far
fewer SNPs
```

Notice the smaller coefficient sums for this simulation, because these are coefficients of almost half the number SNPs run in previous simulations.

Identically to section 6.1, continue setting up the glm functions to be applied:

```
#PCA:
coef_glm1.1 <- function(x) coef(glm(dis ~ x + pc1,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.2 <- function(x) coef(glm(dis ~ x + pc1+pc2,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.3 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.4 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.5 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.6 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6,
                                    data = data_full, family = binomial))
    [2]
coef_glm1.7 <- function(x) coef(glm(dis ~ x + pc1+pc2+pc3+pc4+pc5+pc6+pc7,
                                    data = data_full, family = binomial))
    [2]

#FA:
coef_glm2.1 <- function(x) coef(glm(dis ~ x + fa1.1,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.2 <- function(x) coef(glm(dis ~ x + fa2.1+fa2.2,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.3 <- function(x) coef(glm(dis ~ x + fa3.1+fa3.2+fa3.3,
                                    data = data_full, family = binomial))
    [2]
coef_glm2.4 <- function(x) coef(glm(dis ~ x + fa4.1+fa4.2+fa4.3+fa4.4,
                                    data = data_full, family = binomial))
    [2]
```

```
coef_glm2.5 <- function(x) coef(glm(dis ~ x + fa5.1+fa5.2+fa5.3+fa5.4+fa5
    .5,
                                     data = data_full, family = binomial))
    [2]
coef_glm2.6 <- function(x) coef(glm(dis ~ x + fa6.1+fa6.2+fa6.3+fa6.4+fa6
    .5+fa6.6,
                                     data = data_full, family = binomial))
    [2]
coef_glm2.7 <- function(x) coef(glm(dis ~ x + fa7.1+fa7.2+fa7.3+fa7.4+fa7
    .5+fa7.6+fa7.7,
                                     data = data_full, family = binomial))
    [2]

#ICA:
coef_glm3.1 <- function(x) coef(glm(dis ~ x + X1,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.2 <- function(x) coef(glm(dis ~ x + X1+X2,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.3 <- function(x) coef(glm(dis ~ x + X1+X2+X3,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.4 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.5 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.6 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6,
                                     data = data_full, family = binomial))
    [2]
coef_glm3.7 <- function(x) coef(glm(dis ~ x + X1+X2+X3+X4+X5+X6+X7,
                                     data = data_full, family = binomial))
    [2]
```

GLMs applied over each SNP, and coefficient sums calculated:

```
#Apply and get sums for each model structure!
#PCA
coef1.1_list <- apply(dat, MARGIN = 2, coef_glm1.1)
sum1.1 <- sum(abs(coef1.1_list))
coef1.2_list <- apply(dat, MARGIN = 2, coef_glm1.2)
sum1.2 <- sum(abs(coef1.2_list))
```

```r
coef1.3_list <- apply(dat, MARGIN = 2, coef_glm1.3)
sum1.3 <- sum(abs(coef1.3_list))
coef1.4_list <- apply(dat, MARGIN = 2, coef_glm1.4)
sum1.4 <- sum(abs(coef1.4_list))
coef1.5_list <- apply(dat, MARGIN = 2, coef_glm1.5)
sum1.5 <- sum(abs(coef1.5_list))
coef1.6_list <- apply(dat, MARGIN = 2, coef_glm1.6)
sum1.6 <- sum(abs(coef1.6_list))
coef1.7_list <- apply(dat, MARGIN = 2, coef_glm1.7)
sum1.7 <- sum(abs(coef1.7_list))

#FA
coef2.1_list <- apply(dat, MARGIN = 2, coef_glm2.1)
sum2.1 <- sum(abs(coef2.1_list))
coef2.2_list <- apply(dat, MARGIN = 2, coef_glm2.2)
sum2.2 <- sum(abs(coef2.2_list))
coef2.3_list <- apply(dat, MARGIN = 2, coef_glm2.3)
sum2.3 <- sum(abs(coef2.3_list))
coef2.4_list <- apply(dat, MARGIN = 2, coef_glm2.4)
sum2.4 <- sum(abs(coef2.4_list))
coef2.5_list <- apply(dat, MARGIN = 2, coef_glm2.5)
sum2.5 <- sum(abs(coef2.5_list))
coef2.6_list <- apply(dat, MARGIN = 2, coef_glm2.6)
sum2.6 <- sum(abs(coef2.6_list))
coef2.7_list <- apply(dat, MARGIN = 2, coef_glm2.7)
sum2.7 <- sum(abs(coef2.7_list))

#ICA
coef3.1_list <- apply(dat, MARGIN = 2, coef_glm3.1)
sum3.1 <- sum(abs(coef3.1_list))
coef3.2_list <- apply(dat, MARGIN = 2, coef_glm3.2)
sum3.2 <- sum(abs(coef3.2_list))
coef3.3_list <- apply(dat, MARGIN = 2, coef_glm3.3)
sum3.3 <- sum(abs(coef3.3_list))
coef3.4_list <- apply(dat, MARGIN = 2, coef_glm3.4)
sum3.4 <- sum(abs(coef3.4_list))
coef3.5_list <- apply(dat, MARGIN = 2, coef_glm3.5)
sum3.5 <- sum(abs(coef3.5_list))
coef3.6_list <- apply(dat, MARGIN = 2, coef_glm3.6)
sum3.6 <- sum(abs(coef3.6_list))
coef3.7_list <- apply(dat, MARGIN = 2, coef_glm3.7)
sum3.7 <- sum(abs(coef3.7_list))
```

Finally, compare sum of SNPs:

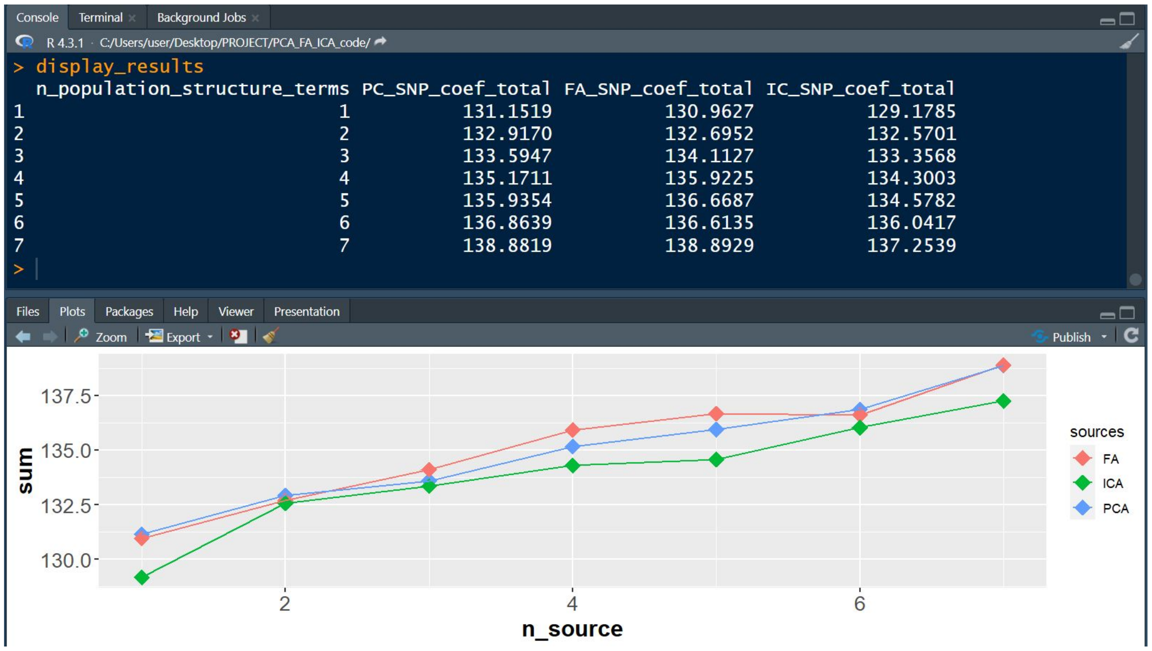
```r
#mini table

n_sources <- c(1, 2, 3, 4,5,6,7)
PC_sums <- c(sum1.1,sum1.2,sum1.3,sum1.4,sum1.5,sum1.6,sum1.7)
FA_sums <- c(sum2.1, sum2.2, sum2.3, sum2.4,sum2.5, sum2.6,sum2.7)
IC_sums <- c(sum3.1, sum3.2, sum3.3, sum3.4,sum3.5,sum3.6,sum3.7)

display_results <- data.frame(n_population_structure_terms = n_sources,
                              PC_SNP_coef_total = PC_sums,
                              FA_SNP_coef_total = FA_sums,
                              IC_SNP_coef_total = IC_sums)

###A quick visual comparison of coefficient sums

bar_stac <- data.frame(sources=rep(c("PCA","FA","ICA"), each = 7),
                       n_source=rep(n_sources, times = 3),
                       sum = c(PC_sums, FA_sums, IC_sums))

ggplot(bar_stac, aes(col=sources, y=sum, x=n_source)) +
  geom_point(shape = 18, size = 5) +
  geom_line(aes(group = sources))+
  #geom_smooth(aes(group = sources), method = "loess", se = FALSE)+
  theme(axis.text=element_text(size=14),axis.title=element_text(size=16,
    face="bold"))
```

The results are shown in figure 9.

6.3 Simulations with SNPs drawn from different regions of the genome and assigned different true positive rates

In previous simulations (four-source, q-source, and the 650-SNP 7-source experiments), the approximate probability of the affection status variable has remained consistent at 50%; perhaps it is the ubiquity of this hypothetical disease that makes it so difficult to disentangle causal mutations (truly associated SNPs, of which there should be none) from population structure, which has thus far been so noisy as to generally *increase* SNP coefficient totals rather than decrease them.

Thus, the inspiration for this simulation: what happens if I reduce the initial probability of having the phenotype in question from a popular 50% to a meagre 10%? Will the *weight* of causes of linkage disequilibrium noticeably shift from causal mutations to population structure ([53]), thereby rendering the corrective terms more useful?



```
> display_results
  n_population_structure_terms PC_SNP_coef_total FA_SNP_coef_total IC_SNP_coef_total
1                            1          131.1519          130.9627          129.1785
2                            2          132.9170          132.6952          132.5701
3                            3          133.5947          134.1127          133.3568
4                            4          135.1711          135.9225          134.3003
5                            5          135.9354          136.6687          134.5782
6                            6          136.8639          136.6135          136.0417
7                            7          138.8819          138.8929          137.2539
> 
```

Figure 9: Recall that the sum of SNP coefficients without any corrective terms was 128.6201. Ideally the sum of SNPs would go *down* with the addition of suitable corrective terms (sources). Rather than suggesting ICA produces the best corrective terms for linkage disequilibrium, these results seem to suggest that ICA generates the *least noisy* (but still unhelpful) sources. Back to section 6.2.

At the outset of this simulation it also occurred to me that using the first available nucleotides from the .geno file might mean I'm using only telomeres ([29]); since the"aglets" ([10]) of genes may not vary or reproduce to the same degree that expressed genes do, I thought working deeper within the genome might produce more meaningful results for dimensionality reduction.

First, working with SNPs from a deeper section of the genome meant starting the data cleaning over from the very beginning (reading in the .geno, and as it turns out, .ped, files). To work with a different sections of the genome, clean1, clean2, clean3, and clean4 now represent different cleaned subsets of the original .geno file.

```
##Data Setup
#Load and Clean

#test<-ped2geno("hapmap3_r2_b36_fwd.consensus.qc.poly.ped")
read1<-read.geno("hapmap3_r2_b36_fwd.consensus.qc.poly.geno")
#for some unknown reason, my LEA:read.geno() function is producing read1 as
    a list instead of a matrix!
#very frustration
```

```
#the .geno file in wd() has also mysteriously shrunk from 1.6 million KB (
    last I checked) to 600 thousand KB today!
#it does seem to have lost information, as well, reading in 1440616 items
    in the past, and recently, just 530572.


#instead, I'm starting from .ped scratch:
test<-ped2geno("hapmap3_r2_b36_fwd.consensus.qc.poly.ped")
'
  - number of detected individuals: 1184
  - number of detected loci:    1440616
'
#this refreshsed the .geno to its original 1.6 million KB.
read1<-read.geno("hapmap3_r2_b36_fwd.consensus.qc.poly.geno")
#and read1 came in as a large matrix this time!


max.col <-NA
for(i in 1:ncol(read1)) max.col[i]<-max(read1[,i])
clean1<-read1[,max.col==2]
#range(clean1) #confirms the range is now 0 to 2

#to save space, periodically remove now-cleaned data frames
rm(test, read1, max.col) #even though it took ages to get read1 back,
#it also takes ages to save.image whenever read1 is still present. it's
    enormous!!

var.col <- NA
for (i in 1:ncol(clean1))
  var.col[i] = var(clean1[,i])
clean2 <- clean1[,var.col!=0]

rm(clean1, var.col)

#this time, let's select a subset of SNPs from the middle of the genome;
#if these are ordered by location on chromosomes, we may have been working
#on telomeres all this time! (this is why I had to re-load read1)
#due to their un-expressed-ness, I don't know if telomeres would
#vary in a comparable manner to verbose genes.

dim(clean2) # 1184 464817 # Ostensibly could easily save 45 different
    cleaned SNP datasets.

```

```r
corr.col <- findCorrelation(cor(clean2[,c(15000:25000)]), cutoff=0.9)

clean3 <- clean2[,c(15000:25000)]
clean3 <- clean3[,-corr.col]

save.image("quiet_disease.R")

corr.col <- findCorrelation(cor(clean2[,c(115000:125000)]), cutoff=0.9)

clean1 <- clean2[,c(115000:125000)]
clean1 <- clean1[,-corr.col]

corr.col <- findCorrelation(cor(clean2[,c(65000:75000)]), cutoff=0.9)

clean4 <- clean2[,c(65000:75000)]
clean4 <- clean4[,-corr.col]

corr.col <- findCorrelation(cor(clean2[,c(285000:295000)]), cutoff=0.9)

clean2 <- clean2[,c(285000:295000)]
clean2 <- clean2[,-corr.col]
#save clean1 through clean2 and you have 4 cleaned data sets of <10,000
    SNPs to play with.

rm(corr.col)
```

6.3.1 Simulation: clean1

Then, all the same dimensionality reduction and glm steps as for the 7-source simulation took place on clean1.

```r
#Randomly Assigning Disease Status
#I've used clean 1 but I could equivalently replace with clean2,3, or 4 and
     get different results.

disease.index <- sample(1:nrow(clean1), nrow(clean1)/10, replace=FALSE)

disease.status <- rep(0,nrow(clean1))

for (i in disease.index)
  disease.status[i] = 1

mock_disease_data <- cbind(disease.status, clean1)

```

```
mock_disease_data[1:50,1:3] #just to check it looks right.
dim(mock_disease_data) #1184 7701
sum(mock_disease_data[,1]==1) #118 is as close to perfectly 10% as you can
    get


save.image("quiet_disease.R")


##Population Structure (Dimensionality Reduction)
#PCA          ### (on a further reduced SNP dataset)
pca7<-principal(mock_disease_data[,1150:1950],nfactors=7,scores=TRUE)

pc1 <- pca7$scores[,1]
pc2 <- pca7$scores[,2]
'etc...
'
```

The results did not seem to be affected by the application of SNPs from a different region, nor from the quietening of the disease from a 50% to a 10% true positive rate. You can see the results below, and in figure 10.

```
> sum0
[1] 178.2907
> display_results
  n_population_structure_terms PC_SNP_coef_total FA_SNP_coef_total IC_SNP_
    coef_total
1                             1          184.9270          184.8696
     179.2463
2                             2          190.2060          190.6781
     182.4553
3                             3          191.5545          191.7453
     183.5689
4                             4          193.2681          194.0190
     185.8412
5                             5          194.6280          195.4045
     194.8508
6                             6          195.8271          195.8073
     196.5005
7                             7          198.2682          198.0318
     198.3857
```

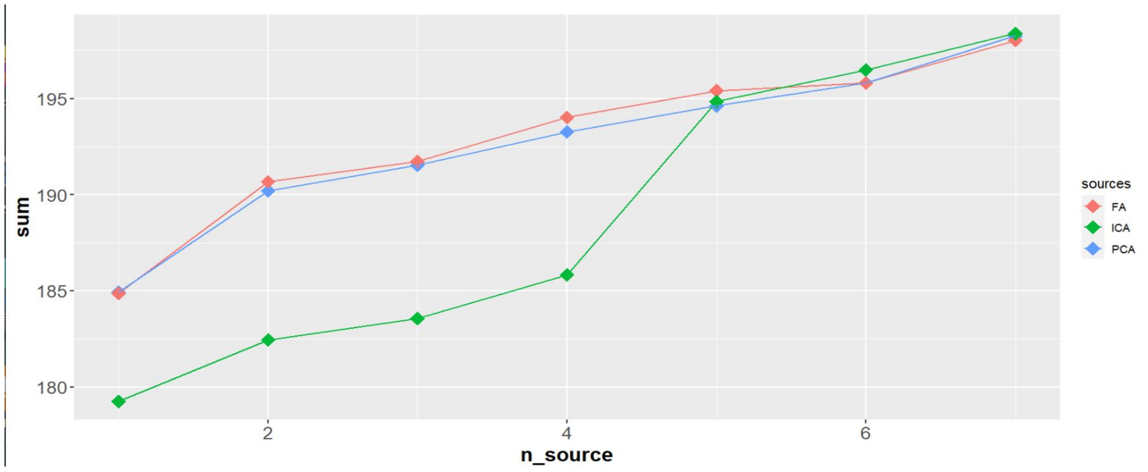


Figure 10: ICA still largely results in lower SNP coefficient sums, and adding sources overall seems to increase sums, with sum0 for this simulation being 178.2907; this indicates that even with a less prevalent affection status variable, and drawing from SNPs that may have a very different population structure, the addition of dimensionality reduction sources seems only to introduce more variance to the model. Back to section 6.3.1.

6.3.2 Simulation: clean2

The above experiment was entirely repeated for the clean2 subset of SNPs from even deeper within the genome.

Results are displayed below, and in figure 11

```
> sum0
[1] 285.6903
> n_sources <- c(1, 2, 3, 4,5,6,7)
> PC_sums <- c(sum1.1,sum1.2,sum1.3,sum1.4,sum1.5,sum1.6,sum1.7)
> FA_sums <- c(sum2.1, sum2.2, sum2.3, sum2.4,sum2.5, sum2.6,sum2.7)
> IC_sums <- c(sum3.1, sum3.2, sum3.3, sum3.4,sum3.5,sum3.6,sum3.7)
>
> display_results <- data.frame(n_population_structure_terms = n_sources,
+                               PC_SNP_coef_total = PC_sums,
+                               FA_SNP_coef_total = FA_sums,
+                               IC_SNP_coef_total = IC_sums)
> display_results
  n_population_structure_terms
1                            1
2                            2
3                            3
4                            4
5                            5
```

```
6                                6
7                                7
  PC_SNP_coef_total
1           294.0758
2           299.0050
3           302.2168
4           303.8494
5           305.8859
6           307.0437
7           309.0332
  FA_SNP_coef_total
1           294.3132
2           299.6900
3           302.9252
4           304.2691
5           305.3281
6           306.4886
7           309.6854
  IC_SNP_coef_total
1           294.0487
2           295.7103
3           296.5184
4           300.7327
5           302.2843
6           304.6772
7           306.5961
```

6.3.3 Simulation: clean2, 30%

Another simulation on a different section of the clean2 SNP region was carried out with a disease prevalence of 30% instead of 10%. In this case, the one source ICA model *did* appear to reduce inflation of the SNP coefficients; only one other simulation, in section 6.3.3 achieved this.

```
disease.index <- sample(1:nrow(clean2), 3*nrow(clean2)/10, replace=FALSE)
```

```
dat <- as.data.frame(mock_disease_data[,6729:7529])
```

Results are displayed below, and in figure 12

```
> sum0
[1] 114.5034
> display_results
  n_population_structure_terms
```

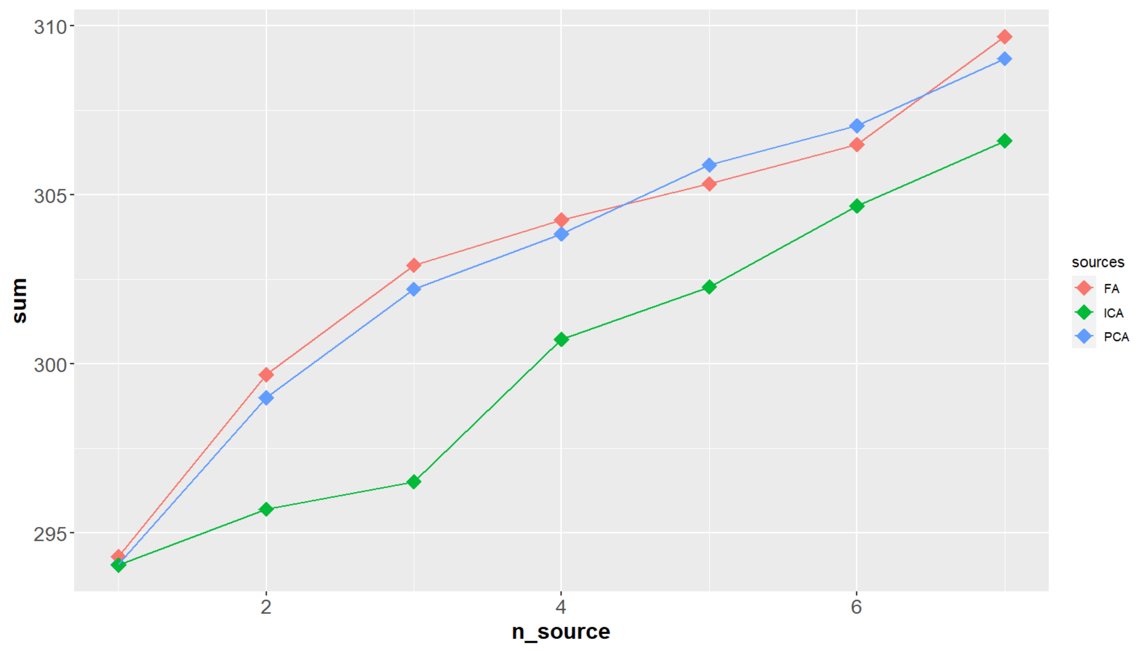


Figure 11: ICA sources clearly provide a better adjustment for coefficient inflation. Bearing in mind that sum0 was 285.6903, however, none of the additions of sources up to $q = 7$ show an improvement over the model structure with no population structure corrective terms. Back to section 6.3.2.

```
1                              1
2                              2
3                              3
4                              4
5                              5
6                              6
7                              7
  PC_SNP_coef_total
1          119.9968
2          119.2739
3          118.6170
4          118.0401
5          117.1470
6          117.9829
7          118.2253
  FA_SNP_coef_total
1          118.1264
2          117.8360
3          118.2070
4          119.4384
```

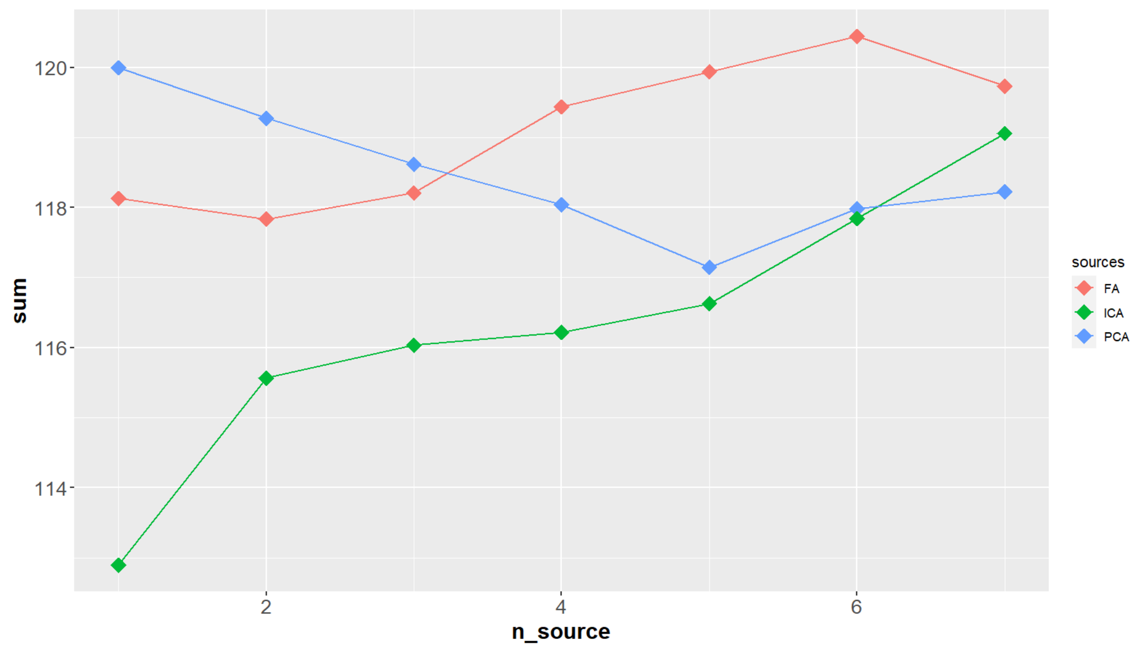


Figure 12: In this simulation, the one-source ICA model improved on glm0. Back to section 6.3.3.

```
5           119.9378
6           120.4416
7           119.7317
  IC_SNP_coef_total
1           112.8925
2           115.5650
3           116.0357
4           116.2203
5           116.6276
6           117.8432
7           119.0594
```

6.3.4 Simulation: clean3, 10%

Results are displayed below, and in figure 13

```
> sum0
[1] 288.4223
> display_results
  n_population_structure_terms
1                            1
2                            2
3                            3
```

```
4                             4
5                             5
6                             6
7                             7
  PC_SNP_coef_total
1          301.7582
2          306.5475
3          310.6156
4          307.8404
5          308.0687
6          308.9575
7          310.9693
  FA_SNP_coef_total
1          302.7957
2          307.8225
3          305.2602
4          307.3025
5          309.9794
6          308.8738
7          311.8760
  IC_SNP_coef_total
1          302.9953
2          305.3637
3          307.3613
4          309.1714
5          305.9587
6          309.6638
7          310.2864
```

6.3.5 Simulation: clean3, 30%

Results are displayed below, and in figure 14

```
> sum0
[1] 365.6647
> display_results <- data.frame(n_population_structure_terms = n_sources,
+                               PC_SNP_coef_total = PC_sums,
+                               FA_SNP_coef_total = FA_sums,
+                               IC_SNP_coef_total = IC_sums)
> display_results
  n_population_structure_terms
1                             1
2                             2
3                             3
```

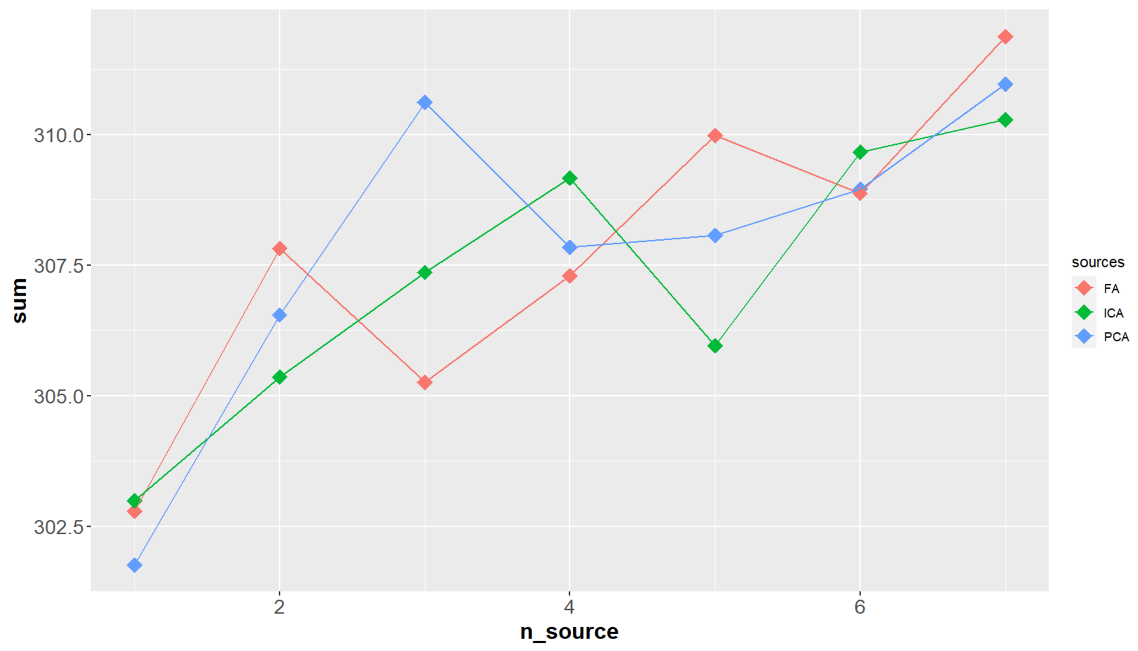


Figure 13: In this simulation, as in many others, none of the sources explained SNP coefficient inflation. However, it's also intriguing to notice that the usual obvious hierarchy, in which ICA largely having the lowest sum of SNPs and FA most often having the highest, is not present in this simulation: instead, all three dimensionality reductions take the highest sum of coefficients in turns. Back to section 6.3.4.

```
4                             4
5                             5
6                             6
7                             7
  PC_SNP_coef_total
1          370.5226
2          371.6784
3          373.1622
4          375.1309
5          376.1938
6          376.9110
7          378.4594
  FA_SNP_coef_total
1          371.2313
2          372.2337
3          374.6376
4          375.5314
5          378.1775
6          380.3026
```

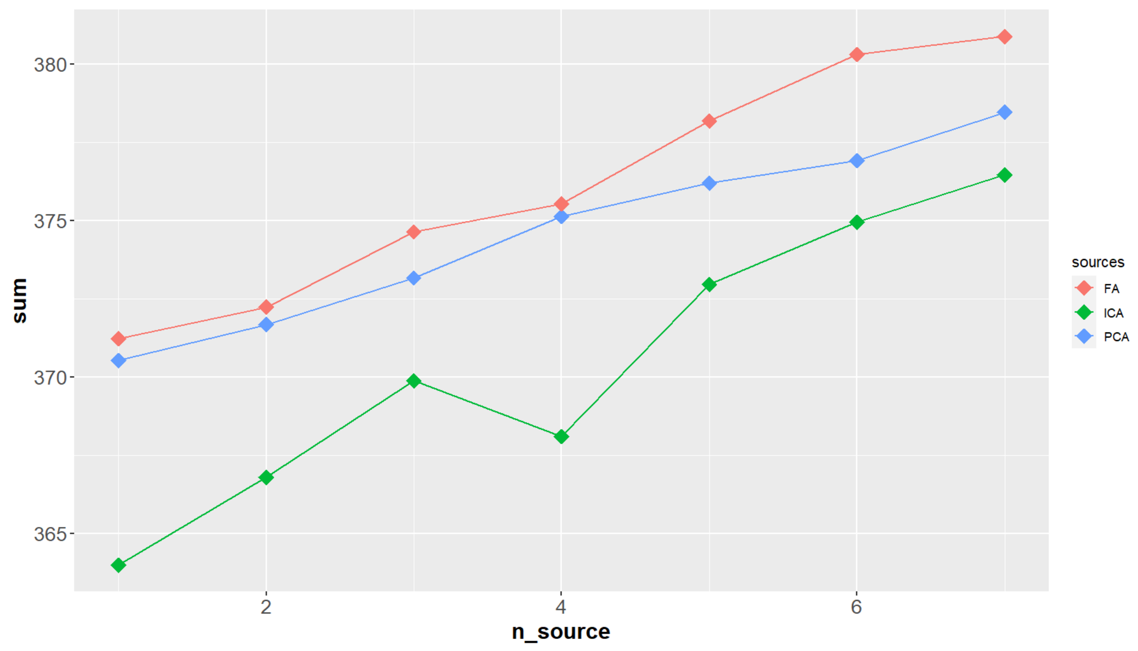


Figure 14: With sum0 = 365.6647, the only model structure that improves on SNP coefficient sums is the one with one corrective term generated by ICA. Back to section 6.3.5.

```
7           380.8751
  IC_SNP_coef_total
1           363.9879
2           366.7997
3           369.8792
4           368.0980
5           372.9520
6           374.9384
7           376.4605
```

6.4 Larger Simulations without Factor Analysis

Based on previous simulations, Factor Analysis seems to be the worst candidate for producing sources that correct for inflation. Furthermore, the lowest ceiling for the number of SNPs that can be run through the simulations is produced by the error that factanal() returns for high-dimensional (or even close to high-dimensional) data. Therefore, the following simulations are run only on PCA's principal() and ICA's fastICA() functions, on larger subsets of SNPs, to see if the sources continue perform poorly when there is a greater amount of genetic variation to caputre.

For each simulation, the start of the code looks like this, all other steps being largely the same, leaving out FA-related code:

```r
#each time, these will be programmed with a 20% disease prevalence
disease.index <- sample(1:nrow(clean4), nrow(clean4)/5, replace=FALSE)
disease.status <- rep(0,nrow(clean4))

for (i in disease.index)
  disease.status[i] = 1

mock_disease_data <- cbind(disease.status, clean4)
```

Each non-FA simulation was run on 2000 SNPs instead of 1000 or even fewer.

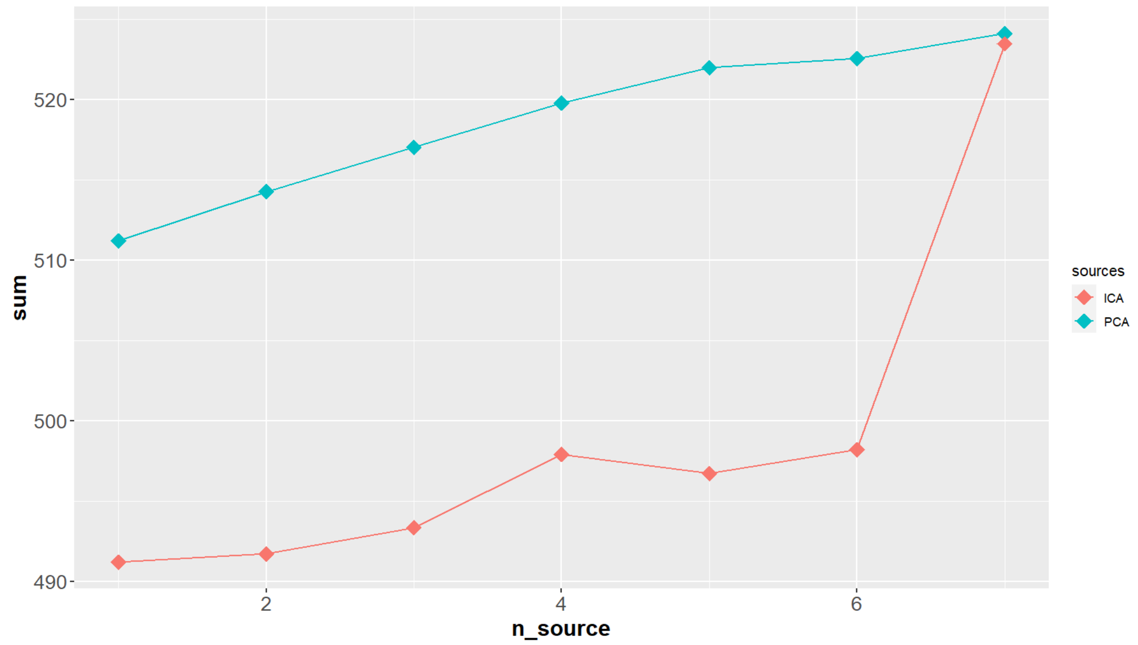
```r
pca7<-principal(mock_disease_data[,1:2000],nfactors=7,scores=TRUE)
ica<-fastICA(mock_disease_data[,1:2000],7)
dat <- as.data.frame(mock_disease_data[,1:2000])
snp_names <- as.character(1:2000)
colnames(dat) <- snp_names

dis <- mock_disease_data[,1]


data_full <- cbind(dis,dat,pca_sources,#fa_sources,
                   ica_sources)
data_full <- as.data.frame(data_full)
```

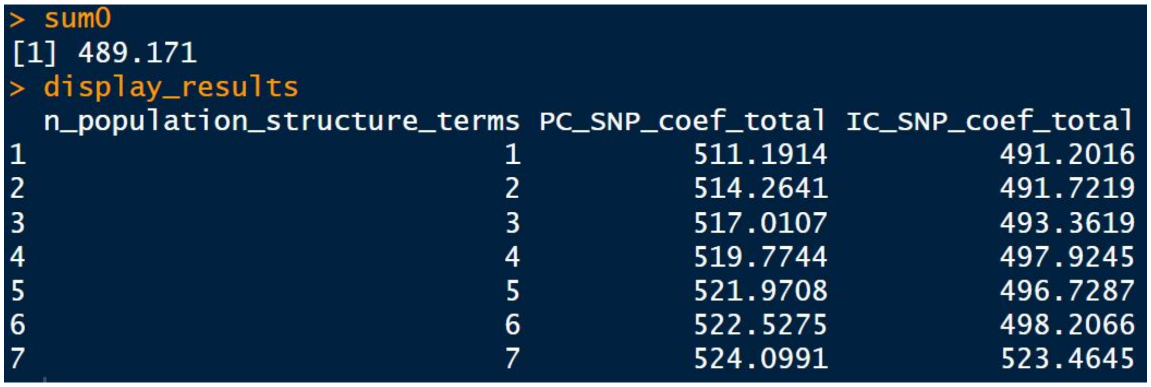
Numerical and graphical results for the simulation of 2000 SNPs without factor analysis are shown in figure 15 for the clean4 SNP subset, 16 for clean3, 17 for clean2, and 18 for clean1.

7 Early Exploration and Troubleshooting

7.1 Factor Analysis Hypothesis Tests on smaller subset: an exploration in failure

See figure 19 for strong evidence against *every* factor model, with the ‘least bad’ being the 55-factor and the 65-factor models; simply put, this led to the nonsensical comparison of a 55-factor model structure from FA against a 2-component model structure for both PCA and ICA. However, concern over selecting the best factor model among all possible sizes of FA is beside the point; the main aim of this project to compare PCA, FA, and ICA with *each other* in the context of genomic data.





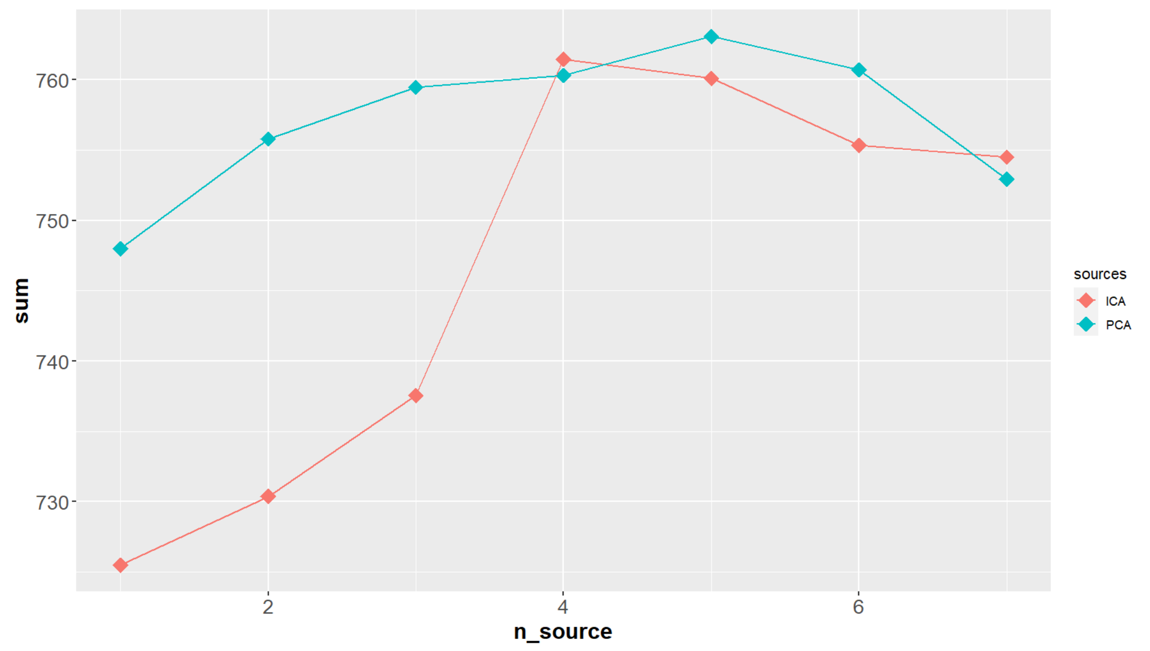
```
> sum0
[1] 489.171
> display_results
  n_population_structure_terms PC_SNP_coef_total IC_SNP_coef_total
1                            1          511.1914          491.2016
2                            2          514.2641          491.7219
3                            3          517.0107          493.3619
4                            4          519.7744          497.9245
5                            5          521.9708          496.7287
6                            6          522.5275          498.2066
7                            7          524.0991          523.4645
```

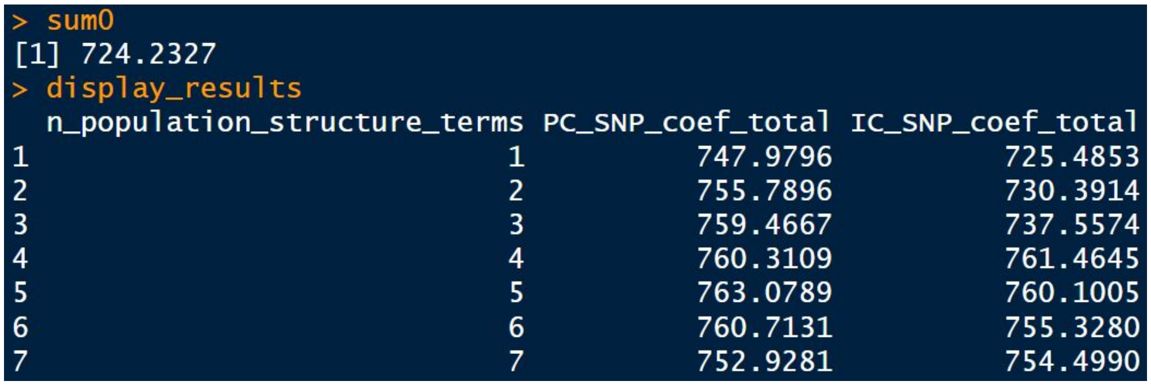
Figure 15: In this larger simulation on the clean4 subset described in section 6.4, ICA continues to outperform PCA, and the sources still don't seem to correct for coefficient inflation. Back to section 6.4.

7.2 Thought experiment: Another way to simulate disease status for comparative simulations

Initially I considered two approaches to artificially generate the affection status variable with respect to SNP data:

- Randomly assign disease status.
- Use one to five SNPs in an equation to generate an output variable, treat that response as the log-odds of disease status, and use it to produce a vector of most likely disease status.





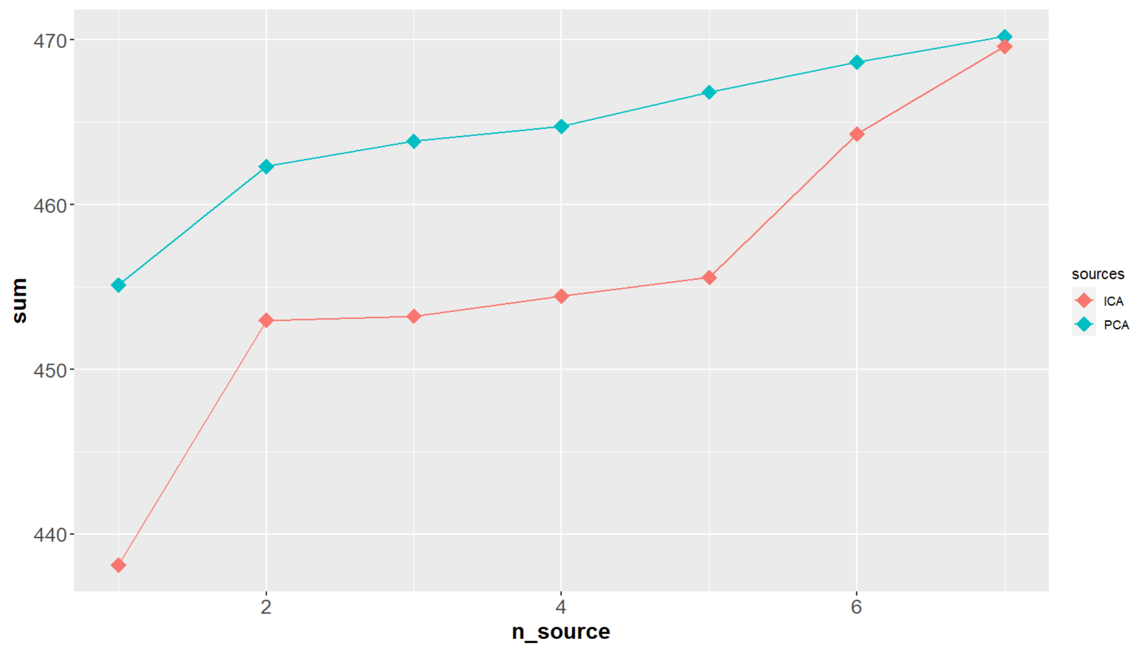
```
> sum0
[1] 724.2327
> display_results
  n_population_structure_terms PC_SNP_coef_total IC_SNP_coef_total
1                            1          747.9796          725.4853
2                            2          755.7896          730.3914
3                            3          759.4667          737.5574
4                            4          760.3109          761.4645
5                            5          763.0789          760.1005
6                            6          760.7131          755.3280
7                            7          752.9281          754.4990
```

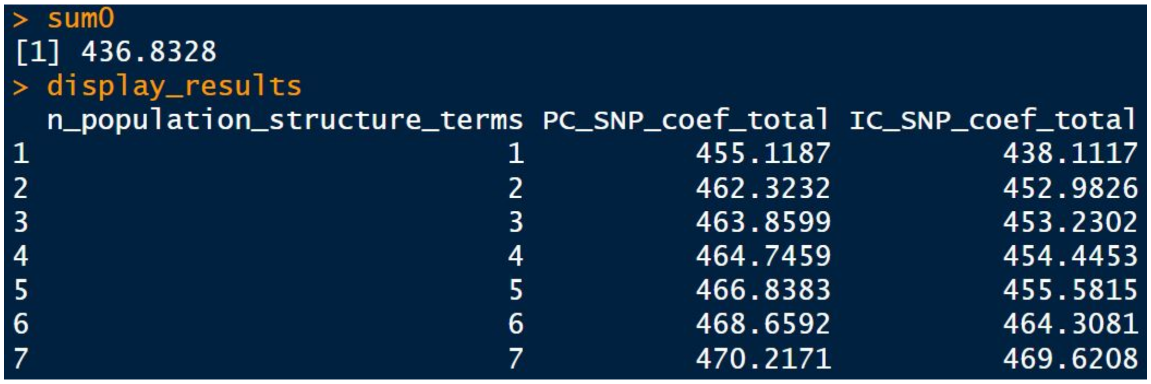
Figure 16: Results of no FA simulation for clean3 subset. Back to section 6.4.

After developing GLM's with SNPs and varying population structure scores, the two approaches would offer different interpretations:

- **If affection status variable is not associated with any SNPs:** Because there is no association between the disease and any of the SNPs, we would expect the SNP coefficients to all approach 0. The best model will be the one with the smallest sum of SNP coefficients.

- **If affection status variable is associated with some SNPs:** The best model should similarly diminish all SNPs, apart from the handful truly associated with the disease. As with all GLMs, the best model should have the highest prediction accuracy (e.g. in a confusion table).





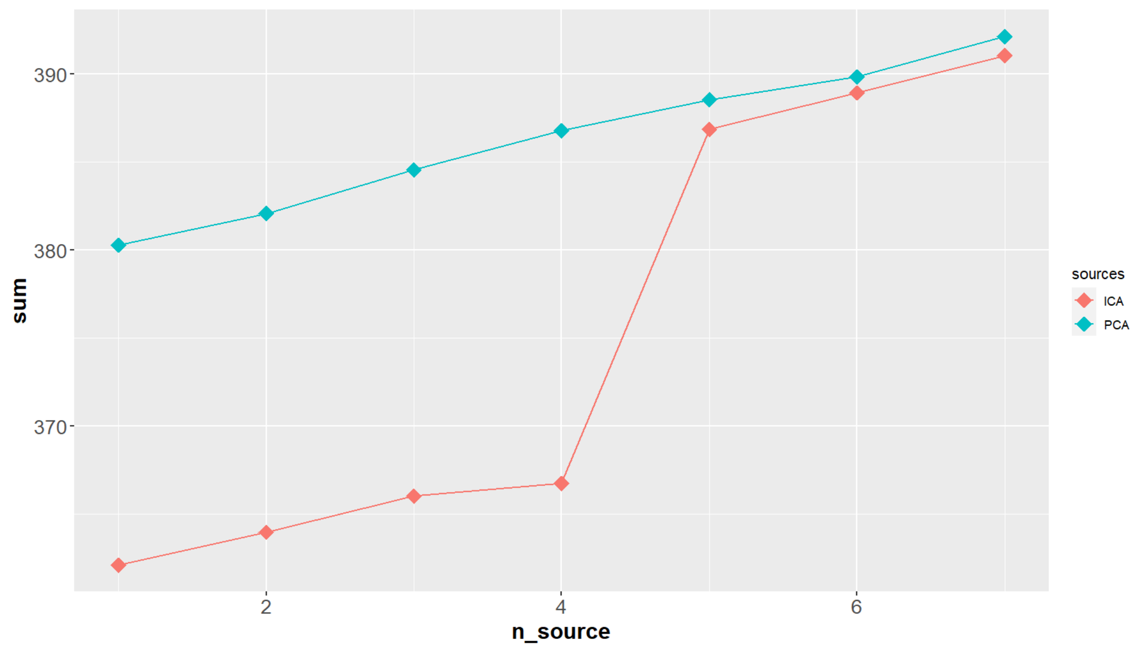
```
> sum0
[1] 436.8328
> display_results
  n_population_structure_terms PC_SNP_coef_total IC_SNP_coef_total
1                            1          455.1187          438.1117
2                            2          462.3232          452.9826
3                            3          463.8599          453.2302
4                            4          464.7459          454.4453
5                            5          466.8383          455.5815
6                            6          468.6592          464.3081
7                            7          470.2171          469.6208
```

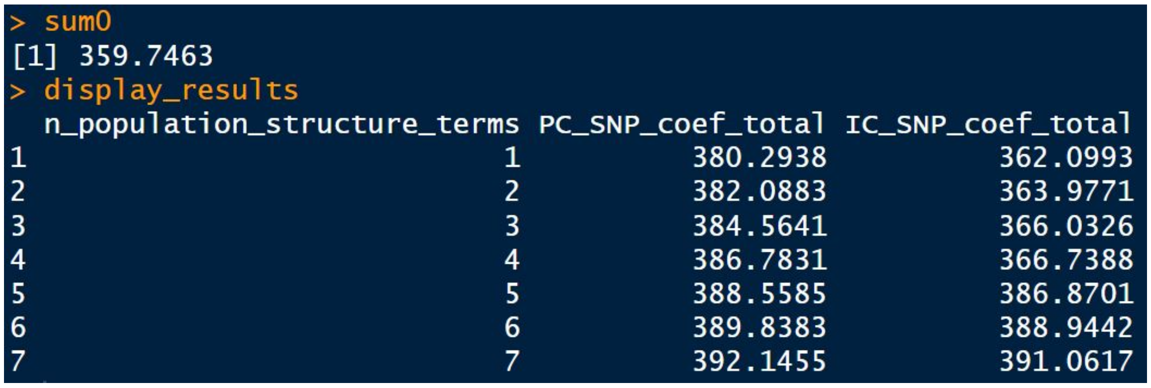
Figure 17: Results of no FA simulation for clean2 subset. Back to section 6.4.

7.3 Experimental simulation: All SNPs in one GLM and confusion table success metric

The following is a sample of code in which I attempted to run glm across all SNPs at once (only appropriate on a small dataset), and then, out of curiosity, set up testing and training data to examine the prediction error of the glm.

```

#Save Screen
save.image("small_GWAS.R")

setwd("C:/Users/user/Desktop/PROJECT/PCA_FA_ICA_code")

load("small_GWAS.R")

```



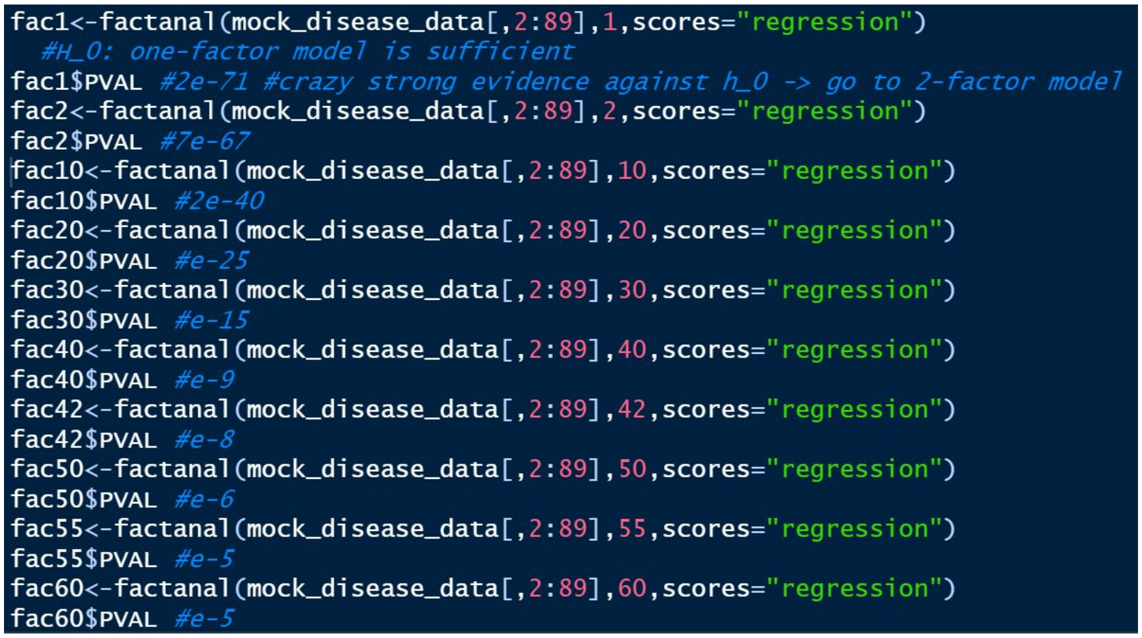


```
> sum0
[1] 359.7463
> display_results
  n_population_structure_terms PC_SNP_coef_total IC_SNP_coef_total
1                            1          380.2938          362.0993
2                            2          382.0883          363.9771
3                            3          384.5641          366.0326
4                            4          386.7831          366.7388
5                            5          388.5585          386.8701
6                            6          389.8383          388.9442
7                            7          392.1455          391.0617
```

Figure 18: Results of no FA simulation for clean1 subset. Back to section 6.4.

```
#Packages to Load

#install.packages("humarray") #version 1.2
#library(humarray)
library(LEA)
library(caret)
library(psych)
library(stats)
library(fastICA)

##Data Setup
#Load and Clean

#test<-ped2geno("newhap1map.ped")
```

Figure 19: With only 89 data points versus 88 SNPs, there was evidence against the sufficiency of *every* q-factor model, with the 55-factor and 65-factor models having the *least* strong evidence against them, with a still minuscule p-value of 10^{-5}.



```
fac1<-factanal(mock_disease_data[,2:89],1,scores="regression")
  #H_0: one-factor model is sufficient
fac1$PVAL #2e-71 #crazy strong evidence against h_0 -> go to 2-factor model
fac2<-factanal(mock_disease_data[,2:89],2,scores="regression")
fac2$PVAL #7e-67
fac10<-factanal(mock_disease_data[,2:89],10,scores="regression")
fac10$PVAL #2e-40
fac20<-factanal(mock_disease_data[,2:89],20,scores="regression")
fac20$PVAL #e-25
fac30<-factanal(mock_disease_data[,2:89],30,scores="regression")
fac30$PVAL #e-15
fac40<-factanal(mock_disease_data[,2:89],40,scores="regression")
fac40$PVAL #e-9
fac42<-factanal(mock_disease_data[,2:89],42,scores="regression")
fac42$PVAL #e-8
fac50<-factanal(mock_disease_data[,2:89],50,scores="regression")
fac50$PVAL #e-6
fac55<-factanal(mock_disease_data[,2:89],55,scores="regression")
fac55$PVAL #e-5
fac60<-factanal(mock_disease_data[,2:89],60,scores="regression")
fac60$PVAL #e-5
```

```
read1<-read.geno("newhap1map.geno")

max.col <-NA
for(i in 1:ncol(read1)) max.col[i]<-max(read1[,i])
clean1<-read1[,max.col==2]

rm(read1, max.col)

var.col <- NA
for (i in 1:ncol(clean1))
  var.col[i] = var(clean1[,i])
clean2 <- clean1[,var.col!=0]

rm(clean1, var.col)

corr.col <- findCorrelation(cor(clean2[,c(1:12000)]), cutoff=0.9)

clean3 <- clean2[,c(1:12000)]
clean3 <- clean3[,-corr.col]

rm(clean2, corr.col,i)

```

```r
#Randomly Assigning Disease Status

disease.index <- sample(1:nrow(clean3), nrow(clean3)/2, replace=FALSE)

disease.status <- rep(0,nrow(clean3))

for (i in disease.index)
  disease.status[i] = 1

mock_disease_data <- cbind(disease.status, clean3)

mock_disease_data[1:10,1:5] #just to check it looks right.
sum(mock_disease_data[,1]==1) #44 out of 89 is roughly half.
dim(mock_disease_data)

rm(clean3, disease.index, i)

rm(test)

##### Global Environment now contains
#####  disease.status (length = 89) and mock_disease_data (dim = 89 by
    10127)

save.image("small_GWAS.R")


#Train
test.index <- sample(1:nrow(mock_disease_data),
                     nrow(mock_disease_data)/4, replace=FALSE)

train_data <- mock_disease_data[-test.index,]

#Check: This is the true disease status of the test data.

true_disease <- mock_disease_data[test.index,1]

#Test

test_data <- mock_disease_data[test.index,-1]

rm(test.index, disease.status)
##### now Environment contains mock_disease_data, test_data,
##### train_data, and true_disease
```

```r
save.image("small_GWAS.R")

##GLM

glm_train <- glm(train_data[,1]  ~ train_data[,-1],
                        family = binomial())

#Now is a good time to derive all-numerical coefficients.

coef_glm <- replace(glm_train$coefficients,
                    is.na(glm_train$coefficients), 0)

save.image("small_GWAS.R")

##Predictions
#Preliminary Functions
dot_coeff <- function(any_vector){
  any_vector %*% coef_glm
}

add_intercept <- function(any_vector){
  cbind(rep(1, dim(test_data)[1]) , any_vector)
}
```

The following function was created to work around an error; as a result, this code resembles lasso regression [25], in which unimportant predictors are assigned coefficients of zero, according to some tuning parameter (also called λ, coincidentally).

```r
infs_100 <- function(any_vector){
  for (i in 1:length(any_vector)){
    if (exp(any_vector[i]) == Inf){
      any_vector[i] <- 100}
  }
  return(any_vector)}

logit_response <- function(any_vector){
  exp(any_vector)/(exp(any_vector)+1)
}

disease_predict <- function(any_vector){
  round(any_vector)
}


```

```
save.image("small_GWAS.R")

#Prediction

pred_test <- apply(add_intercept(test_data), MARGIN = 1, dot_coeff)

pred_disease <- disease_predict(logit_response(infs_100(pred_test)))
head(pred_disease)
range(pred_disease)
table(pred_disease)

save.image("small_GWAS.R")

#Confusion Table (for fun)

table(true_disease) #in truth there were 12
#assigned with the disease, not 7.

table(true_disease, pred_disease)
```

The results of that final table are as follows:

	Predicted Negative	Predicted Positive
True Negative	6	5
True Positive	4	7

At the very least, this table demonstrates that without correcting for population structure, the prediction rate is awful, with a test error rate of $\frac{4+5}{22} \approx 45\%$.

7.4 Trial and Error with principal() and an examination of poor cumulative variance capture

This section of the paper is dominated by a small narrative of the sort of experimentation and troubleshooting that was inextricable from this project. To skip to the essential code for reproduction, see subsection 4.2.1.

Attempting to generate principal components using the principal() function didn't work on the cleaned hapmap3_r2 dataset in the following code, since the matrix was too memory intensive compared to our initial experimental code.

```
>pca2<-principal(mock_disease_data[,-1],nfactors=2,scores=TRUE)
```

The above code does not include the first column of data, which is the affection status variable, and response in an association model; it is not a SNP and therefore isn't part of the population structure.

In an effort to improve functionality, I examined advice from [37]; this tangent reminded me of prior experiments ([20]) which demonstrated it was necessary to reduce the number of SNPs so that it would not exceed the number of samples, and even this returned warnings about matrix insufficiency[13]:

```
> pca2<-principal(mock_disease_data[,2:1184],nfactors=4,scores=TRUE)
'
The determinant of the smoothed correlation was zero.
This means the objective function is not defined.
Chi square is based upon observed residuals.
The determinant of the smoothed correlation was zero.
This means the objective function is not defined for the null model either.
The Chi square is thus based upon observed correlations.
Warning messages:
1: In cor.smooth(r) : Matrix was not positive definite, smoothing was done
2: In principal(mock_disease_data[, 2:1184], nfactors = 4, scores = TRUE) :
  The matrix is not positive semi-definite, scores found from Structure
    loadings
'
```

The metric for successful dimensionality reduction in this experiment will be the minimal sum of SNP coefficients. However, it is interesting to observe that in such high-dimensional data, even as many as four principal components only account for 15% of the total variance among the SNPs; this observation prompted more thorough experiments.

```
> pca2$Vaccounted
                              RC1         RC2
SS loadings           99.13893172 53.28465595
Proportion Var         0.08380299  0.04504197
Cumulative Var         0.08380299  0.12884496
Proportion Explained   0.55460135  0.29808413
Cumulative Proportion  0.55460135  0.85268547
                              RC4         RC3
SS loadings           13.48491806 12.84860007
Proportion Var         0.01139892  0.01086103
```

[13]Why is it important to principal() to estimate a positive definite or positive semi-definite matrix? Recall from section 3.1.1 that PCA ranks components based on the size of their eigenvectors. According to [50], if A is a positive definite matrix, then its eigenvectors are all greater than 0; failing this stipulation, there is ambiguity in how the components should be ranked.

```
Cumulative Var          0.14024388   0.15110491
Proportion Explained    0.07543710   0.07187742
Cumulative Proportion   0.92812258   1.00000000
```

Regardless, for the purposes of our study, from this PCA we produced four vectors of scores:

```
> pc1 <- pca2$scores[,1]
> pc2 <- pca2$scores[,2]
> pc3 <- pca2$scores[,3]
> pc4 <- pca2$scores[,4]
> length(pc3)
[1] 1184
#Confirms that there is a score for each sample.
```

7.5 Attempts to improve principal() functionality without dropping SNPs

Why was this worth attempting? According to [37], the historical limit on vector size for R data sets is $2^31 - 1$. As you can see in the code below, even our large data, once cleaned, and already reduced to under 10,000 SNPs, is well below that limit.

```
> dim(mock_disease_data)
[1] 1184 9171
> 1184*9171 < 2^31 -1
[1] TRUE
```

Looking at Task Manager, it certainly appeared as though the primary cause of R freezing was insufficient working memory, rather than insufficient static memory. Therefore, parallel computing and the *bigstatsr* package, both suggested applications of CRAN Task View ([48]), looked like potential solutions; however, the former would take a lot more time to craft working code, and the latter would be a fix only for PCA, but I anticipated similar problems would arise for FA and ICA.

Next was the solution proposed by statquant on Feb 26, 2013 on [37]; however, I am working from a Windows 10 laptop rather than a multicore computer. The solution offered by aL3xa on March 8, 2011, was too specific to the read.table() function, as there is no colClasses argument in principal().

One potential approach was inspired by the R documentation on principal() [48], which indicated that an argument of r, the correlation matrix, could be separately indicated. Could prior access to the correlation matrix save computing memory and time for principal()?

```
> ?cor
> sum(is.na(mock_disease_data))
[1] 0
> corr_matrix <- cor(mock_disease_data[,-1])
> dim(corr_matrix)
[1] 9170 9170
```

This reveals that a correlation matrix over the samples was done over the SNPs; since this was generated from a smaller matrix, it must not be full rank, which will cause problems for certain forms of dimensionality reduction. It was at this point that I remembered a necessary missing step: reduce the number of SNPs so that it did not exceed the number of samples; this is why the prior examples of 89 samples were limited to 88 SNPs. Back to section 4.2

Ultimately, this project is testing the efficacy of different dimensionality reduction methods against *one another*, so the number of SNPs to be summarised in population structure in our artificial model need not be specifically large or unwieldy, given that it is consistent across PCA, FA, and ICA. Because the project model measures dimensionality reduction by the remaining sum of the coefficients of the SNPs, it is also not essential to identify the cumulative proportion of the variance captured by the components in each case; after all, using "variance captured" as a proxy measure for efficient population structure would simply maintain the arbitrary hegemony of PCA within genomics, when in fact, this project aims to establish whether FA or ICA might do a better job of reducing spurious correlations in a disease association model.

8 Acknowledgements

I am grateful for the opportunity to develop my skills and knowledge of data analysis, R, and genomics. Thanks to the many developers of R and Plink, for creating free, thoroughly designed and documented, open-source software. Thanks to those who freely collected and saved genomic data for further study, specifically [5], [35], and [13]. Thanks to my friend, Rob, and my husband, Chris, for their attempts at technical support. Thanks to my aunt, Erin, for letting me talk through key ideas from the paper. Thanks to inspiring scientists and science communicators for fostering curiosity, including Dr. Sarah [23], Kiran Krishnan, Max Luggavere, and Andrew Huberman. And thanks especially to my project supervisor, James Bentham, for your thoughtful guidance, and for allowing me to take part in your research; best of luck in 2024!

References

[1] B.F. Abrantes and K.G. Ostergaard. “Digital footprint wrangling: are analytics used for better or worse? A concurrent mixed methods research on the commercial (ab)use of dataveillance”. In: *Journal of Marketing Analytics* (2021). URL: https://link.springer.com/article/10.1057/s41270-021-00144-5.

[2] Statistical tools for high-throughput data analysis. *Best Practices in Preparing Data Files for Importing into R*. URL: http://www.sthda.com/english/wiki/best-practices-in-preparing-data-files-for-importing-into-r.

[3] P.I. de Bakker et al. “Practical aspects of imputation-driven meta-analysis of genome-wide association studies”. In: *Hum Mol Genet* (2008). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2782358/.

[4] Pritha Bhandari. *Statistical Power and Why It Matters | A Simple Introduction*. 2021. URL: https://www.scribbr.com/statistics/statistical-power/.

[5] Brigham and Harvard University Women’s Hospital. *A PLINK tutorial*. 2017. URL: https://zzz.bwh.harvard.edu/plink/tutorial.shtml.

[6] D. Carey. *Market Volatility And Your Retirement Plan*. URL: https://www.mywealthtrace.com/blog/market-volatility-and-your-retirement-plan.

[7] J. Chiang et al. “High-grade glioma in infants and young children is histologically, molecularly, and clinically diverse: Results from the SJYC07 trial and institutional experience”. In: *Neuro-Oncology* (2023). URL: https://doi.org/10.1093/neuonc/noad130.

[8] D.G. Clayton et al. “Population structure, differential bias and genomic control in a large-scale, case-control association study.” In: *Nat Genet* (2005). URL: https://pubmed.ncbi.nlm.nih.gov/16228001/.

[9] B. Devlin and K. Roeder. “Genomic control for association studies”. In: *Biometrics* (1999). URL: https://pubmed.ncbi.nlm.nih.gov/11315092/.

[10] Merriam-Webster Dictionary. “aglet”. In: (). URL: https://www.merriam-webster.com/agdictionary/aglet.

[11] E. Emmenegger et al. “Statistical Learning of Large-Scale Genetic Data: How to Run a Genome-Wide Association Study of Gene-Expression Data using the 1000 Genomes Project Data”. In: *Statistics in Biosciences* (2023). URL: https://link.springer.com/article/10.1007/s12561-023-09375-9.

[12] S. Fairley et al. "The International Genome Sample Resource (IGSR) collection of open human genomic variation resources". In: *Nucleic Acids Research, Volume 48, Issue D1, pages D941-D947* (2020). URL: https://doi.org/10.1093/nar/gkz836.

[13] P. Flicek. *Accessing the 1000 Genomes Data*. URL: https://www.genome.gov/Pages/Research/DER/ICHG-1000GenomesTutorial/How_to_Access_the_Data.pdf.

[14] GitHub > Docs > Plink » File Format. *File format reference*. URL: https://plink.readthedocs.io/en/latest/plink_fmt/.

[15] M.L. Freedman et al. "Assessing the impact of population stratification on genetic association studies". In: *Nat Genet* (2004).

[16] X. Gao et al. "A multiple testing correction method for genetic association studies using correlated single nucleotide polymorphisms." In: *Genet Epidemiol.* (2008). URL: https://pubmed.ncbi.nlm.nih.gov/18271029/.

[17] GitHub. *read.pedfile: Reading a Ped File*. URL: https://rdrr.io/bioc/trio/man/read.pedfile.html.

[18] GP2 Complex Disease Data Analysis Working Group. "Understanding GWAS: Part 2 – Additional Insights and Tips". In: *ASAP: Aligning Science Across Parkinson's* (2021).

[19] T. Hammond-Antrobus. *Playing with HapMap3 and Psudeocode*. 2023. URL: https://wordpress.com/post/sistertree321.wordpress.com/485.

[20] T. Hammond-Antrobus. *Population Structure*. 2023. URL: https://sistertree321.wordpress.com/2023/08/15/15-08-23-population-structure/.

[21] T. Hammond-Antrobus. *Project Idea: Contraception Selection*. 2023. URL: https://sistertree321.wordpress.com/2023/07/22/project-idea-contraception-selection/.

[22] T. Hammond-Antrobus. *The invisible little darlings inside you*. 2023. URL: https://sistertree321.wordpress.com/2023/07/25/the-invisible-little-darlings-inside-you/.

[23] S.E. Hill. *This Is Your Brain on Birth Control*. Avery: An Imprint of Penguin Random House LLC, 2019.

[24] A. Hyvarinen et al. *Independent Component Analysis*. John Wiley & Sons, Inc, 2001.

[25] G. James et al. In: Springer Texts in Statistics, 2017. Chap. 6.

[26] John. *Calculate inflation observed and expected p-values from uniform distribution in QQ plot*. 2014. URL: https://stats.stackexchange.com/questions/110755/calculate-inflation-observed-and-expected-p-values-from-uniform-distribution-in.

[27] B.D. Jovanovic and P.S. Levy. “A Look at the Rule of Three”. In: *The American Statistician* (1997).

[28] David Reich Lab. *Ancient DNA, Biology, and Disease : Input File Formats and Conversion Program*. URL: https://reich.hms.harvard.edu/software/InputFileFormats.

[29] J. Lee et al. “Biochemistry, Telomere And Telomerase”. In: *Treasure Island (FL): StatPearls Publishing* (2022). URL: https://www.ncbi.nlm.nih.gov/books/NBK576429/.

[30] D. Li and M. Kuhn. “findCorrelation: Determine highly correlated variables”. In: *caret documentation* (2023). URL: https://rdrr.io/cran/caret/man/findCorrelation.html.

[31] H. Li et al. “An efficient unified model for genome-wide association studies and genomic selection.” In: *Genet Sel Evol.* (2017). URL: https://pubmed.ncbi.nlm.nih.gov/28836943/.

[32] C. Liu et al. “Statistical Populatation Genomics”. In: Methods in Molecular Biology, vol 2090. Humana, New York, NY., 2020. Chap. Exploring Population Structure with Admixture Models and Principal Component Analysis.

[33] National Library of Medicine. *Download large genome data packages*. 2023. URL: https://www.ncbi.nlm.nih.gov/datasets/docs/v2/how-tos/genomes/large-download/.

[34] Genomics Boot Camp Mésáros G. *Genome wide association studies | Introduction to genomics theory | Genomics101 (beginner-friendly)*. 2022. URL: https://www.youtube.com/watch?v=7iarFfxYxQc.

[35] University of Michigan School of Public Health. *HapMap3 r2 Phased Data Download*. URL: http://csg.sph.umich.edu/abecasis/MACH/download/HapMap3.r2.b36.html.

[36] J. Novembre and M. Stephens. “Interpreting principal component analyses of spatial population genetic variation.” In: *Nature Genetics* (2008). URL: https://doi.org/10.1038/ng.139.

[37] *Practical limits of R data frame*. 2011. URL: https://stackoverflow.com/questions/5233769/practical-limits-of-r-data-frame.

[38] J.K. Pritchard et al. “Association Mapping in Structured Populations”. In: *Am J Hum Genet* (2000).

[39] S. Purcell et al. “PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses”. In: *The American Journal of Human Genetics* 81.3 (2007), pp. 559–575. DOI: https://doi.org/10.1086/519795. URL: https://www.sciencedirect.com/science/article/pii/S0002929707613524.

[40] D. Reich et al. “Principal component analysis of genetic data”. In: *Nature Genetics* (2008). URL: https://www.nature.com/articles/ng0508-491.

[41] M Reyes. *No Time To Die: The Evil Plot Of Rami Malek’s Safin Explained*. 2021. URL: https://www.cinemablend.com/movies/no-time-to-die-the-evil-plot-of-rami-maleks-safin-explained.

[42] N.M. Roslin et al. “Quality control analysis of the 1000 Genomes Project”. In: *bioRxiv 078600* (2016). URL: https://doi.org/10.1101/078600.

[43] Paterson A.D. Roslin N.M. Weili L. and Strug L.J. “.5 genotypes”. In: *bioRxiv* (2016). URL: https://www.biorxiv.org/%20content/early/2016/09/30/078600.

[44] M. Slatkin. “Linkage disequilibrium — understanding the evolutionary past and mapping the medical future”. In: *Nat Rev Genet.* (2008). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5124487/.

[45] O. Teymur. “MAST7053: Statistical Learning for Data Scientists”. In: 2023.

[46] D. Cole. “MAST7012: Statistical Consultancy and Data Presentation”. In: 2023.

[47] B. Santos. “MAST8820: Advanced Regression Modelling”. In: 2023.

[48] Open Source. *R Documentation*. URL: https://www.rdocumentation.org/.

[49] BBC author ulisted. “Henrietta Lacks: ’Mother’ of modern medicine honoured”. In: *BBC* (2021). URL: https://www.bbc.co.uk/news/world-us-canada-58903934.

[50] Purdue University. “Lecture 4.9: Positive definite and semidefinite forms”. In: 2020. URL: https://www.math.purdue.edu/~eremenko/dvi/lect4.9.pdf.

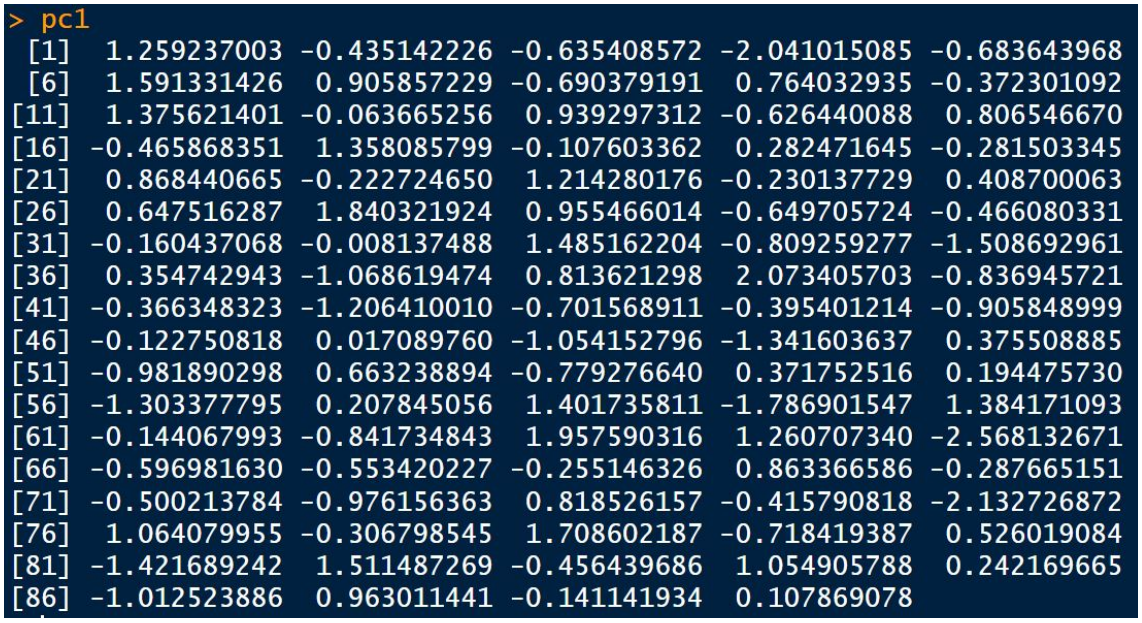
[51] Cross Validated. *What is the advantage of FastICA over other ICA algorithms?* 2017. URL: https://stats.stackexchange.com/questions/260399/what-is-the-advantage-of-fastica-over-other-ica-algorithms.

[52] *Various types of variant: what is genomic variation?* 2016. URL: https://www.genomicseducation.hee.nhs.uk/blog/various-types-of-variant-what-is-genomic-variation/.

[53] J. Wang et al. “Torkamaneh, D., Belzile, F. (eds) Genome-Wide Association Studies. Methods in Molecular Biology, vol 2481.” In: Humana, New York, NY, 2022. Chap. Interpretation of Manhattan Plots and Other Outputs of Genome-Wide Association Studies. URL: https://doi.org/10.1007/978-1-0716-2237-7_5.

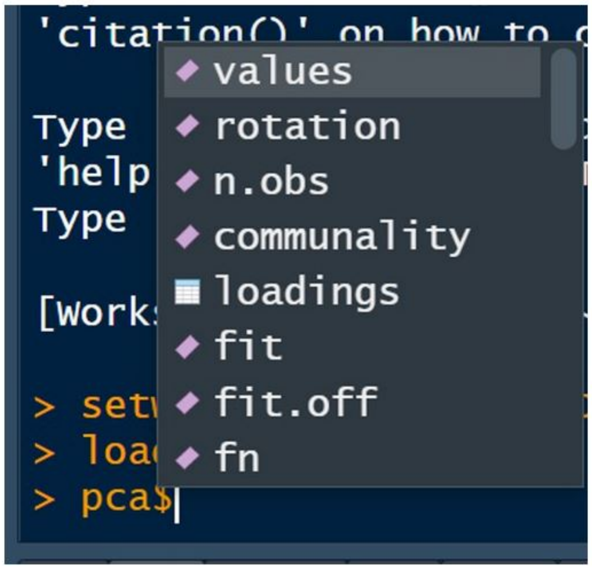
[54] J. Ye. “Genomic inflation factor calculation”. In: *Bioinformatics-Genomics-Genetics (Wordpress)* (2016). URL: https://bioinformaticsngs.wordpress.com/2016/03/08/genomic-inflation-factor-calculation/.

Figure 20: You can see that calling the 1st set of scores produces a vector of length 89, one element for each sample that has been projected onto the first axis.



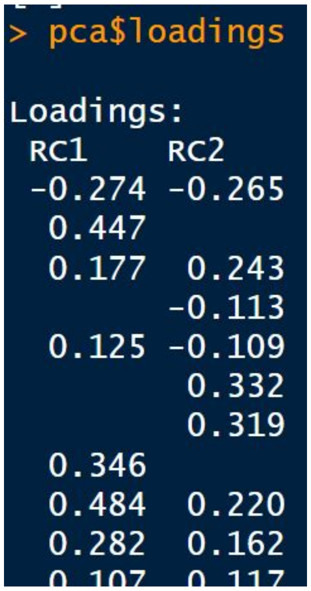
```
> pc1
 [1]  1.259237003 -0.435142226 -0.635408572 -2.041015085 -0.683643968
 [6]  1.591331426  0.905857229 -0.690379191  0.764032935 -0.372301092
[11]  1.375621401 -0.063665256  0.939297312 -0.626440088  0.806546670
[16] -0.465868351  1.358085799 -0.107603362  0.282471645 -0.281503345
[21]  0.868440665 -0.222724650  1.214280176 -0.230137729  0.408700063
[26]  0.647516287  1.840321924  0.955466014 -0.649705724 -0.466080331
[31] -0.160437068 -0.008137488  1.485162204 -0.809259277 -1.508692961
[36]  0.354742943 -1.068619474  0.813621298  2.073405703 -0.836945721
[41] -0.366348323 -1.206410010 -0.701568911 -0.395401214 -0.905848999
[46] -0.122750818  0.017089760 -1.054152796 -1.341603637  0.375508885
[51] -0.981890298  0.663238894 -0.779276640  0.371752516  0.194475730
[56] -1.303377795  0.207845056  1.401735811 -1.786901547  1.384171093
[61] -0.144067993 -0.841734843  1.957590316  1.260707340 -2.568132671
[66] -0.596981630 -0.553420227 -0.255146326  0.863366586 -0.287665151
[71] -0.500213784 -0.976156363  0.818526157 -0.415790818 -2.132726872
[76]  1.064079955 -0.306798545  1.708602187 -0.718419387  0.526019084
[81] -1.421689242  1.511487269 -0.456439686  1.054905788  0.242169665
[86] -1.012523886  0.963011441 -0.141141934  0.107869078
```

Figure 21: There are many more values. In section 3.1.1 are summaries of the most relevant to this project.



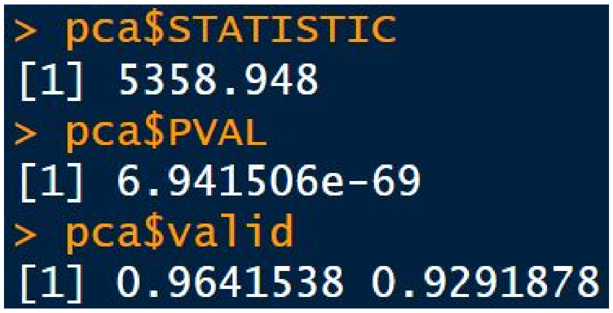
```
'citation()' on how to c

Type
'help
Type

[Work

> set
> loa
> pca$
```

Figure 22: Negative and positive loadings for each element indicate that variables may contribute negatively or positively to the new axis of variation.



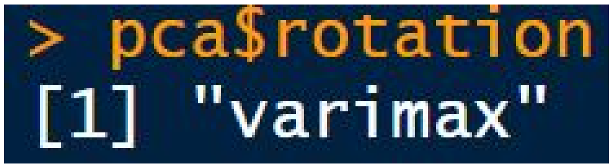
```
> pca$loadings

Loadings:
 RC1    RC2
 -0.274 -0.265
  0.447
  0.177  0.243
        -0.113
  0.125 -0.109
         0.332
         0.319
  0.346
  0.484  0.220
  0.282  0.162
```

Figure 23: The $\$STATISTIC$ value gives the χ^2 test statistic based on the specific number of observations, if known. The $\$PVAL$ returns the probability that the observed χ^2 is larger than this one, if the number of observations is greater than 0. And the value $\$valid$ returns estimates of how well each principal component and its estimated scores are correlated.



```
> pca$STATISTIC
[1] 5358.948
> pca$PVAL
[1] 6.941506e-69
> pca$valid
[1] 0.9641538 0.9291878
```

Figure 24: Indicates how the problem was solved. To see other possible approaches, use the value $\$fa$.



```
> pca$rotation
[1] "varimax"
```