Case Study Based on Human Activity Recognition Using Smartphones Dataset

卢伟灏 2015301000086

Abstract.
Human-centered computing is an emerging field of research, and one of the most recent, challenging and appealing applications in this framework consists in sensing human body motion using smartphones to gather context information about people actions. In this context, we classify the Human Activity Recognition Using Smartphones Dataset. The classification methods include ordinary least squares estimation, cluster analysis, discriminant analysis and principal component regression analysis. By comparing the accuracy of the above categories, we will give a recommended classification method; the same time, through a combination of several methods, we will try to get a higher accuracy. Results, obtained on the dataset by exploiting other possible ways, are also acknowledged.

1 Introduction

基于 n 个观测值，对于一组名义型变量的预测，有很多判别的方法。其中，我们可以将名义型变量化为一组数字，进行最小二乘法回归分析；也可以直接进行聚类分析将样本分为指定的类数。当存在 train data 和 test data 时，聚类分析无法利用 train data 来分类 test data，因而正确率可能不高，可以采用判别分析进行分类。而现实生活中，各 variable 往往存在相关性，甚至相关性很高，因而普通最小二乘法估计系数方差会大得 intolerable，因而此时更适合改用主成分回归分析。

下面简要介绍下各方法的原理：

1) OLS 思想为，假设方程有形式：

$$Y = X\beta + \varepsilon$$

其中ε 满足 Gauss-Markov 条件，求β的线性无偏估计值$\hat{\beta}$，使得离差和

$$Q(\hat{\beta}) = min Q(\beta) = (Y - X\beta)'(Y - X\beta)$$

达到最小。OLS 的优点是求解简单，并且有显式解；但是当模型自变量间相关系数很高时，OLS 往往预测效果极不理想。

2) 聚类分析 根据某种定义的距离来计算样本间的距离，将距离最小的样本聚成一类；或者先将样本粗略分类，不断迭代重新分类来得到最终分类结果。聚类分析计算量主要集中在距离的运算，因而运行速度较快，但只看

分类结果并不能知道某一类代表的含义是什么，比如将一些发达或落后的城市通过聚类分析分类，则并不能从某一城市被分到某一类的结果得知这一城市是发达还是落后。同时，聚类分析不具有学习功能，即训练样本的聚类不能改变测试样本聚类的结果。

3) 判别分析 基于训练样本，构建合适的判别准则，并用此来判别测试样本。用于训练的样本量越多，则正确率越高，但同时耗时更长。

4) 主成分回归分析的方程假设同OLS，但实际中X中各自变量的相关系数极高时，可以作变换将 $X_1, ..., X_p$ 转化为互不相关的单位向量 $F_1, ..., F_p$，即p个主成分，然后使用前k个主成分对Y进行OLS估计，可以消除模型的多重共线性。由于PCA是非参数方法，因而适用范围广，但是PCA会使数据来源不可知。另外，在一些应用领域，PCA主要用于将变量还原为真实世界可以解释的变量，如将观测到的人的一些指标还原为人的社交水平、心理能力、知识水平等，如果真实的变量之间不是正交的（如社交水平和心理能力有关），PCA将会失效[1]。为此，Pierre Comon提出了一种新的方法，ICA (independent component analysis)[2]，被这些领域广泛应用并获得了一定的成功。

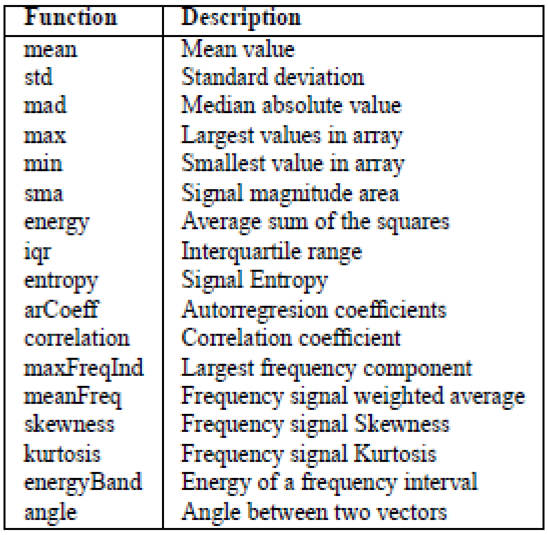
2 Data description

Human Activity Recognition（HAR）旨在通过他/她自己和周围环境的一系列观察确定一个人的行为。但是这些传感器通常令人不舒服，不利于长期的数据记录。而手机带有嵌入式内置传感器，如麦克风，双摄像头，加速度计，陀螺仪等惯性传感器，因而可作为 HAR 的另一种解决方案。这些大众市场的设备提供灵活，经济实惠的独立解决方案，以自动和不显眼的方式监控日常生活活动。因此，在过去的几年中，一些专家已经基于手机设备记录的数据设计出了推测人的行为的算法，例如 Jennifer R 在 Activity Recognition using Cell Phone Accelerometers 中就手机感应三个坐标轴的加速度以 77%至 96%的正确率区分了包括坐立和直立等 5 种行为[3]。
本文中，我们引用来自 UCI 的数据 Human Activity Recognition Using Smartphones Data Set[4]来进行实例的分析。该数据共有 10299 个样本，563 个变量。其中，第 562 个变量 subject 表示测试者的编号，一共有 30 个测试者，70%被选做训练，剩余的 30%则用于测试模型的预测正确性。第 563 个变量 activity labels 以 1 到 6 代表了 6 中不同的状态，具体如下图：

1	WALKING
2	WALKING_UPSTAIRS
3	WALKING_DOWNSTAIRS
4	SITTING
5	STANDING
6	LAYING

Table 1: Different numbers and their representative status

其中 1 到 3 显然为运动的状态，4 到 6 则为相对静止的状态。
而前面的 561 个变量分为三类（具体可参考 dataset 中的 features_info.txt）:
第一类为时域信号（time domain signals）（表示时间的前缀“t”）。来自加速度计和陀螺仪三轴原始信号（tAcc-XYZ 和 tGyro-XYZ），和加速度信号分离成的车身和重力加速度信号（tBodyAcc-XYZ 和 tGravityAcc-XYZ）。
第二类为 Jerk 信号（Jerk signals），由人体线性加速度和角速度导出（tBodyAccJerk-XYZ 和 tBodyGyroJerk-XYZ）。
第三类为频域信号（frequency domain signals）（注意'f'表示频域信号），是一些信号应用快速傅立叶变换（FFT）所产生的（fBodyAcc-XYZ，fBodyAccJerk-XYZ）。
变量后缀则为记录的数据的某一运算值，如均值，标准误，最值等，意义如下表：



Function	Description
mean	Mean value
std	Standard deviation
mad	Median absolute value
max	Largest values in array
min	Smallest value in array
sma	Signal magnitude area
energy	Average sum of the squares
iqr	Interquartile range
entropy	Signal Entropy
arCoeff	Autorregresion coefficients
correlation	Correlation coefficient
maxFreqInd	Largest frequency component
meanFreq	Frequency signal weighted average
skewness	Frequency signal Skewness
kurtosis	Frequency signal Kurtosis
energyBand	Energy of a frequency interval
angle	Angle between two vectors

Table 2: Measures for computing feature vectors

为了消除量纲的影响，数据皆已标准化。
首先，我们查看下前面 12 个变量名：

[1]	"tBodyAcc-mean()-X"	"tBodyAcc-mean()-Y"	"tBodyAcc-mean()-Z"
[4]	"tBodyAcc-std()-X"	"tBodyAcc-std()-Y"	"tBodyAcc-std()-Z"
[7]	"tBodyAcc-mad()-X"	"tBodyAcc-mad()-Y"	"tBodyAcc-mad()-Z"
[10]	"tBodyAcc-max()-X"	"tBodyAcc-max()-Y"	"tBodyAcc-max()-Z"

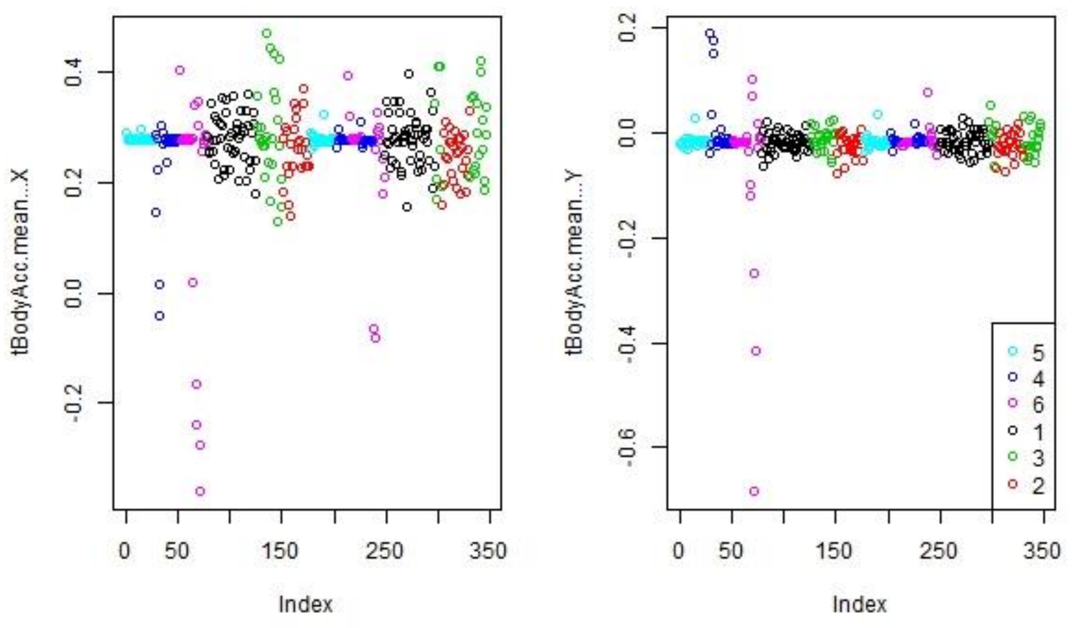
Table 3: The first 12 variables’ names

可以看出，变量包括身体加速度里面的均值，标准差等特征，并且分为 x,y,z 三个方向。尝试列出 train data 中各活动标签的数量：

1	2	3	4	5	6
1226	1073	986	1286	1374	1407

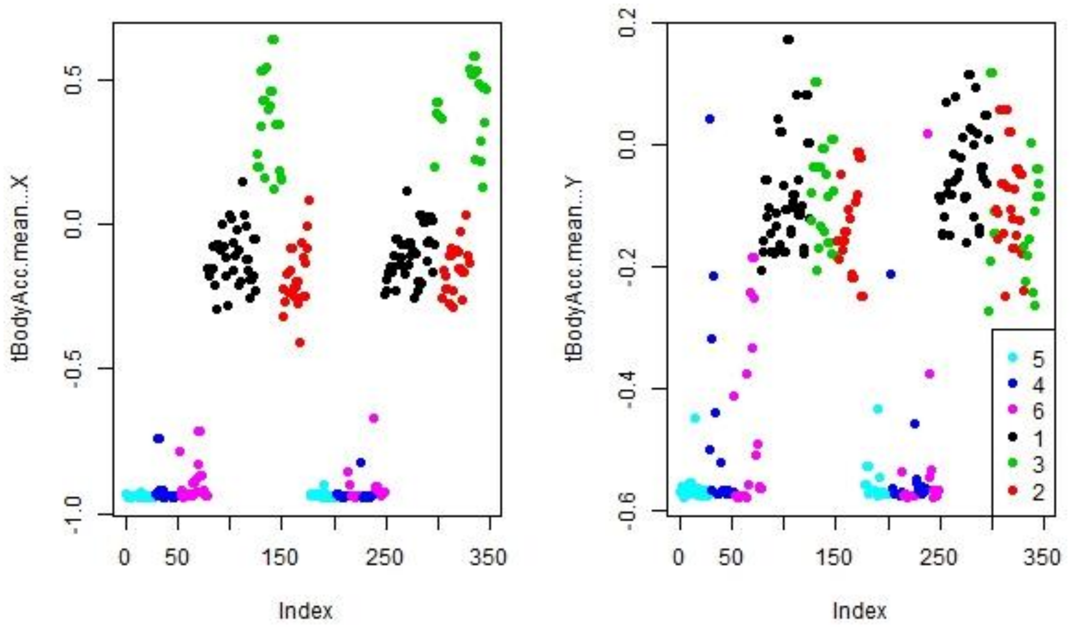
Table 4: Each activity labels’ amount

我们发现，6 种状态有不同的数量，可以在后续研究中通过加速器和陀螺仪收集的各种特征变量来区别这 6 种类别
下面，我们就 train data 中 subject=1 的子集进行描述性统计，大概了解一下这个 dataset 的性质。
首先看看前两个变量的分类情况，即 x 和 y 方向上的平均身体加速度



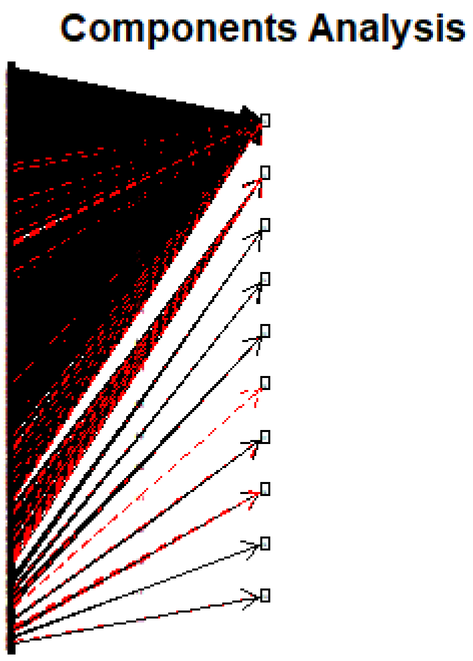
Graph 5: Classification situation for the first two variables

可见 6 种状态之间的区别不明显，基本都是 x 方向有正的加速度，而 y 方向上无加速度(即匀速或者静止)。
然后再看看第 11 和 12 个变量的分类情况，即 x 和 y 方向上的最大身体加速度



Graph 6: Classification situation for the 11th and 12th variables

这两个变量明显区分开了动态组(1,2,3)和静态组(4,5,6)。但是，一旦你进入到两大组的某一组里面，就很难进行进一步的细分组，比如，你测得一个数据 x 方向加速度均值为 0，y 方向加速度均值为-0.1，那么你可以知道这个数据中测试者处于运动状态，但无法知道他是在走路还是在上下楼梯。鉴于此，我们不能单就某一个或某几个变量就能准确地说出携带手机的测试者出于何种状态；相反，我们要对所有变量综合考虑，才有可能得出更精确的结果。
最后，我们尝试对数据进行主成分分析。psych package 中的 fa.diagram()可以画出主成分和原始自变量之间的关系，下面只画出前十个主成分与变量之间的关系图：



Graph 7: Relationship between the former 10 PCs and origin variables

可以看到，前面 2/3 的自变量基本上与 F1 有关，注意到前面的变量属于时域信号（time domain signals）和 Jerk 信号（Jerk signals），都是未经处理的，因而可称 F1 为原始因子。而后面的变量则多与 F2 有关，而后面的变量属于经 FFT 变换处理过的频域信号（frequency domain signals），因而可称 F2 为变换因子。

3 Methodology & experimental results

3.1 OLS analysis

由前面的分析，我们知道，要想预测测试者的状态，必须综合考虑所有变量。因而我们首先想到的是建立自变量与因变量(6 种状态)的线性方程组。为此，我们计算 train data 的 OLS 估计系数，并将得到的模型用于估计 test data，得到的结果如下：

	G_predict							
		1	2	3	4	5	6	Recall
G_origin	1	**436**	60	0	0	0	0	87.9%
	2	39	**416**	16	0	0	0	88.3%
	3	0	105	**305**	10	0	0	72.6%
	4	0	0	5	**408**	77	1	83.1%
	5	0	0	1	43	**475**	13	89.3%
	6	0	0	0	1	16	**520**	96.8%
	Precision	91.8%	71.6%	93.3%	88.3%	83.6%	97.4%	**86.9%**

Table 8: The result of linear model

模型以 87%的准确率估计了 test data 的行为，但是计算模型时提示设计阵存在秩缺乏的情况，这也造成了模型的预测准确率不算太高。进一步计算模型的条件数 ＝9.232255e+64，说明数据存在严重的多重共线性，需要消除多重共线性以获得更好的预测结果。

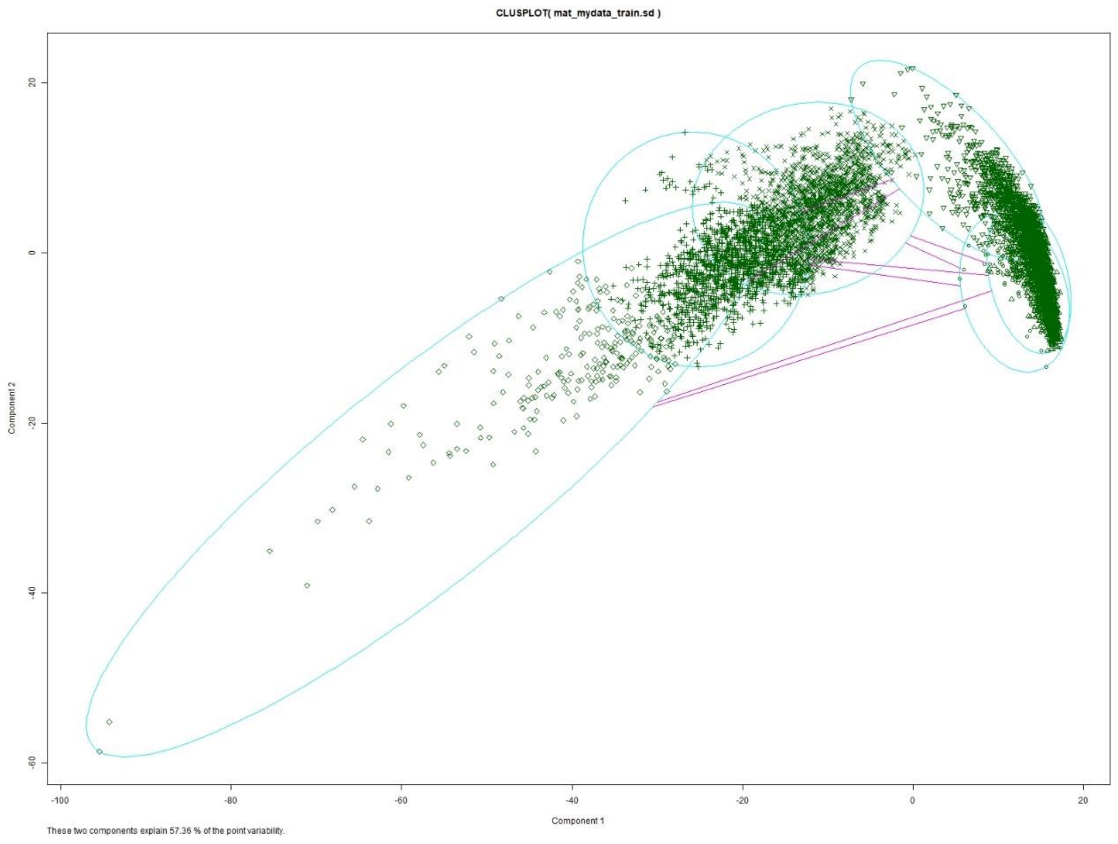
3.2 Clustering analysis

由于动态聚类运行的分类结果一般与原先的分类所用的编号不同，因而我们只能猜测分类结果与原先数字的关系，如下图，分类结果中大部分的 4 属于原来的第 6 类，说明分类结果的 5 对应 laying 这一状态。总体来看，聚类分析的预测结果并不好，可能是不同类别的相似度太高所导致的。

	G_predict						
		1	2	3	4	5	6
G_origin	1	514	68	0	0	0	644
	2	211	5	0	0	2	855
	3	620	166	0	0	0	200
	4	0	0	889	86	310	1
	5	0	0	1003	0	371	0
	6	0	0	26	1107	270	4

Table 9: The result of clustering analysis

我们也可以用 cluster package 中的 clusplot() 来可视化聚类结果，如下所示：



Graph 10: Visualize clustering results with clusplot

可以发现，类与类的重合相当严重，最右的两个聚类圈甚至是互相包含的关系，说明我们不能在此案例中使用聚类分析。同时，聚类分析不能通过 train data 来提高对 test data 的预测准确性，这也进一步使我们怀疑聚类分析在此案例中的效果。

3.3 Principal component regression analysis(PCRA)

在 3.1 的讨论中，我们在建立线性模型时遇到了自变量相关系数极高的问题，并且 OLS 的低预测正确率也印证了模型的严重多重共线性。我们因此转而尝试使用主成分回归分析，构建一系列不相关的正交变量，重新进行线性回归。但是，在本案例中，因变量为 1 到 6 的名义变量，因而主成分回归分析可能精确度不高，同时，PCs 的个数 k 的选取也是一个见仁见智的问题。运行 psych package 的 fa.parallel()，其建议我们选择 44 个主成分进行分析。为此，我们先选择 k=44 进行主成分回归分析，得到结果如下：

	G_predict							
G_origin		1	2	3	4	5	6	Recall
	1	**356**	136	4	0	0	0	71.8%
	2	41	**393**	37	0	0	0	83.4%
	3	9	210	**195**	6	0	0	46.4%
	4	0	2	11	**330**	148	0	67.2%
	5	0	0	7	186	**337**	2	63.3%
	6	0	0	0	2	97	**438**	81.6%
	Precision	87.7%	53.0%	76.8%	63.0%	57.9%	99.5%	**69.5%**

Table 11: The result of PCA, k=44

类别间混淆程度过高，而且某些类别的正确率甚至只有 50%左右。因此尝试加大 k 重新进行分析：

	G_predict							
G_origin		1	2	3	4	5	6	Recall
	1	**430**	66	0	0	0	0	86.7%
	2	45	**406**	20	0	0	0	86.2%
	3	0	112	**292**	16	0	0	69.5%
	4	0	0	6	**422**	59	4	85.9%
	5	0	0	1	59	**464**	8	87.2%
	6	0	0	0	0	19	**518**	96.5%
	Precision	90.5%	69.5%	91.5%	84.9%	85.6%	97.7%	**85.9%**

Table 12: The result of PCA, k=250

混淆程度大大降低，但正确率仍然不太令人满意。分析其原因，有可能是由于因变量非连续值导致线性模型预测不准确；有可能是并没有选对实际中区分这几种状态的正确 k 个主成分；也有可能是真实的变量并非正交，导致主成分这一方法本身就是无效的。

3.4 Discriminant analysis

本节中，我们尝试使用判别分析的 distance method，bayes method 和 fisher method 这 3 种方法分别来预测测试者的状态。由于设计矩阵相关性过高，使得协方差阵奇异，因而 distance method 和 bayes method 皆失败；如果尝试将变量缩减为前 50 个并运行程序，则会由于使用的变量过少，使运行结果的正确率偏低。

	G_predict							
G_origin		1	2	3	4	5	6	Recall
	1	**323**	21	152	0	0	0	65.1%
	2	14	**422**	35	0	0	0	89.6%
	3	0	4	**416**	0	0	0	99.0%
	4	0	2	0	**426**	1	62	86.8%
	5	0	0	1	449	**45**	37	8.5%
	6	0	0	0	0	0	**537**	100.0%
	Precision	95.8%	94.0%	68.9%	48.7%	97.8%	84.4%	**73.6%**

Table 13: Result of discriminant analysis with 50 variables

进一步，我们转而到 fisher method 并运用近似值进行判别分析，结果如下图：

	G_predict							
G_origin		1	2	3	4	5	6	Recall
	1	**490**	6	0	0	0	0	98.8%
	2	11	**460**	0	0	0	0	97.7%
	3	1	14	**405**	0	0	0	96.4%
	4	0	**1**	0	**434**	56	0	88.4%
	5	0	0	0	22	**510**	0	95.9%
	6	0	0	0	0	0	**537**	100.0%
	Precision	97.6%	95.6%	100.0%	95.2%	90.1%	100.0%	**96.2%**

Table 14: Result of Fisher discriminant analysis

正确率达到了 96%，基本上可以说能很好地在实际中预测模型了，特别是在区分动态组和静态组中只有一个错误值，预测的第三组和第六组甚至全部正确。

3.5 The combination of discriminant analysis and PCRA

当样本量极大时，越是正确率高的估计往往需要提取更多样本的内在信息，也就是耗时更长，因而在实现中要注意尽量缩减不必要的和重复的计算量。另一方面，我们也可以先用一种方法粗略进行分类，再在此基础上运用另一种方法进行进一步的细分。
在之前的讨论中，我们注意到判别分析成功地将动态组和静态组区分开来（错误率为 1/2947），因此我们尝试先用判别分析将 test data 分为两组，再运用其他方法分别处理两组样本。下面，我们使用 PCRA 处理分组后的样本，并先尝试两组皆取前 250 个主成分，结果如下：

		G1_predict			G2_predict			
G_origin		1	2	3	4	5	6	Recall
	1	**462**	34	0	0	0	0	93.1%
	2	59	**412**	0	0	0	0	87.5%
	3	0	81	**339**	0	0	0	80.7%
	4	0	1	0	**424**	62	4	86.4%
	5	0	0	0	18	**499**	15	93.8%
	6	0	0	0	0	8	**529**	98.5%
	Precision	88.7%	78.0%	100.0%	95.9%	87.7%	96.5%	**90.4%**
Group precision		**88.9%**			**93.4%**			

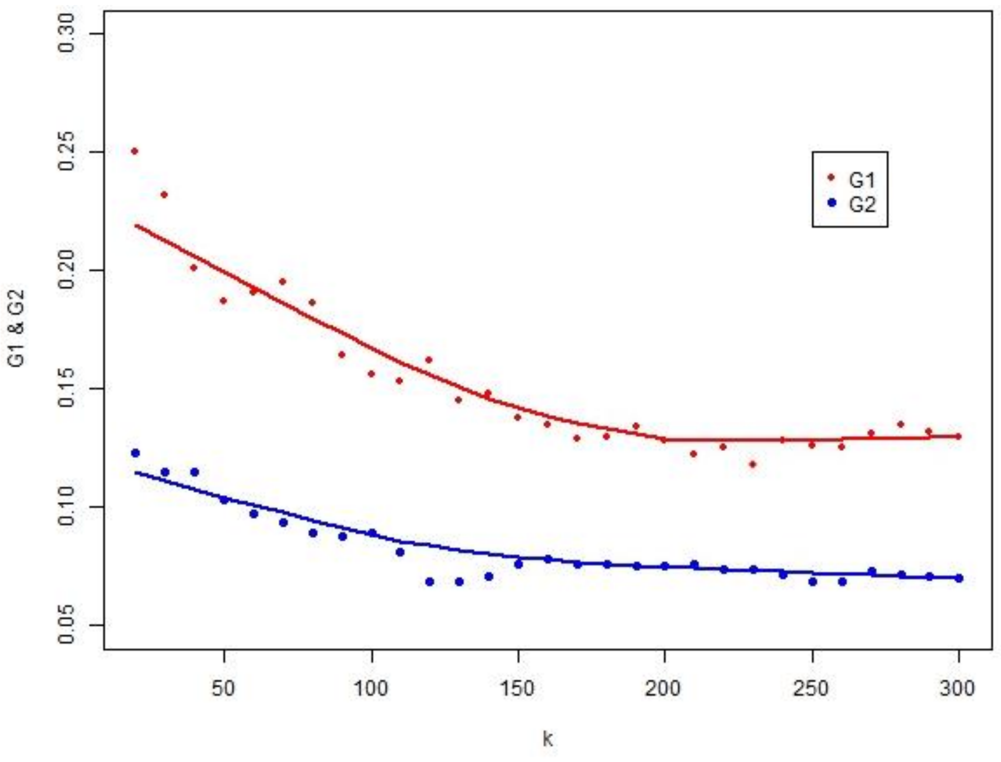
Table 15: Result of PCA analysis: by group, $k_1=k_2=250$

两组的正确率都较单独使用 PCA 有了很大的上升，特别是 G2 组（静态组）的 PCA 回归正确率更是达到了 93%，而这仅仅是初步使用了判别分析的优秀结果和随机选择一组 k 值而已。下面，我们编写循环程序，尝试一组 k 值，尝试寻找下最优的 k 值大概的位置。

k	G1	G2
20	0.250	0.123
30	0.232	0.115
40	0.201	0.115
50	0.187	0.103
60	0.191	0.097
70	0.195	0.094
80	0.186	0.089
90	0.164	0.088
100	0.156	0.089
110	0.153	0.081
120	0.162	0.069
130	0.145	0.069
140	0.148	0.071
150	0.138	0.076
160	0.135	0.078

k	G1	G2
170	0.129	0.076
180	0.13	0.076
190	0.134	0.075
200	0.128	0.075
210	0.122	0.076
220	0.125	0.074
230	0.118	0.074
240	0.128	0.072
250	0.126	0.069
260	0.125	0.069
270	0.131	0.073
280	0.135	0.072
290	0.132	0.071
300	0.13	0.07

Table 16: Measures for computing feature vectors



Graph 17: The relationship between wrong rate and k

可见，随着 k 值(主成分选取个数)的增加，两组的 k 值均逐渐下降并最终稳定，当 k_1 取 210，k_2 取 260 时，正确率基本上趋于稳定，不再随 k 值上升而上升。此时两组的正确率为 88%和 93%。
综上，联立方法无论是理论上还是实际上都取得了较为显著的效果，说明预处理数据不仅使得主成分回归计算量减小，而且还使精度获得了较大的提高。但在实际中其准确率仍不如判别分析，因而仍值得进一步改进模型和联立的方法。

4 Further discussion

4.1 Variable choosing in principle component regression analysis

在主成分回归分析中，某些主成分对因变量不显著，但是我们是把所有主成分考虑进去了，理论上会使估计的方差加大，不利于模型的预测。因此，我们可以在选出 k 个主成分后进行变量的选择，比如说用 step()函数来减少主成分的个数。下以 data G1 为例，取 k=240，并对主成分进行变量选择。

	G1_predict				
		1	2	3	Recall
G_origin	1	**433**	63	0	87.3%
	2	95	**371**	5	78.8%
	3	0	179	**241**	57.4%
	4	0	1	0	
	Precision	82.0%	60.4%	98.0%	**75.3%**

Table 18: The result of PCA with variables selecting, k=240

令人讶异的是，正确率陡然下降至 75%，实际与理论严重不合。观察 step()删除的自变量，发现其删除的变量最前的到了第 46 个，而最后的一些变量反而很多都保留了。实际分析，主成分是按照方差大小排序的，因而较后的主成分即使显著，其对因变量影响也是微乎其微的；而较前的变量即使对整体方程不显著，但可能对某些类的影响是显著的，而基于 AIC 最小来选择变量的 step()函数显然没有考虑到这些因素。
因此，我们加入限制条件，使 step()的搜索范围固定在后面的变量中。由于前面估计有 k=210 时预测值最优，因而我们取 260 个变量，并在主成分选择时固定前 210 个变量，只对后面的变量进行变量选择。

	G1_predict				
		1	2	3	Recall
G_origin	1	**444**	52	0	89.5%
	2	53	**417**	1	88.5%
	3	0	77	**343**	81.7%
	4	0	1	0	
	Precision	89.3%	76.2%	99.7%	**86.7%**

Table 19: The result of PCA with variables selecting and former variables fixed, k=260

step 函数留下了 240 个自变量，预测的正确率为 87%，虽然较无限制下的变量选择效率有所提升，但还是比不上直接进行主成分回归分析。因而认为：至少在本例中，在进行主成分的求取后不必再进行主成分的选择，保留那些不显著的主成分也会对模型的预测有所帮助。

4.2 The selecting of k PCs

在之前的讨论中，我们对 k 值的选择是基于计算各 k 值下 test data 的正确率的，然而，在实际生活中，我们不仅要考虑到估计的正确率，还要考虑到时效性。如果计算时间过长，那么结果出来时用户可能已经改变了状态，那么此次计算将变得无效，因而只由 train data 来确定一个近似最优的或局部最优的 k 值就显得极其重要。
其一，我们可以根据历史经验所确定的最优 k 值所对应的累计贡献率来选择当前的 k 值。在当前案例中，k_1=210 所对应的累计贡献率为 99.24%，k_2=260 所对应的累计贡献率为 99.99%。因此，我们在增大或改变 train data 的样本量时，可以根据这两个贡献率分别选取相应的新 k 值。
其二，psych package 的 fa.parallel()函数可以给出建议的 k 值，但是由于模型是

奇异的，因而推荐值可能不准确；而且其推荐的 k 值普遍较小，不利于模型的预测。

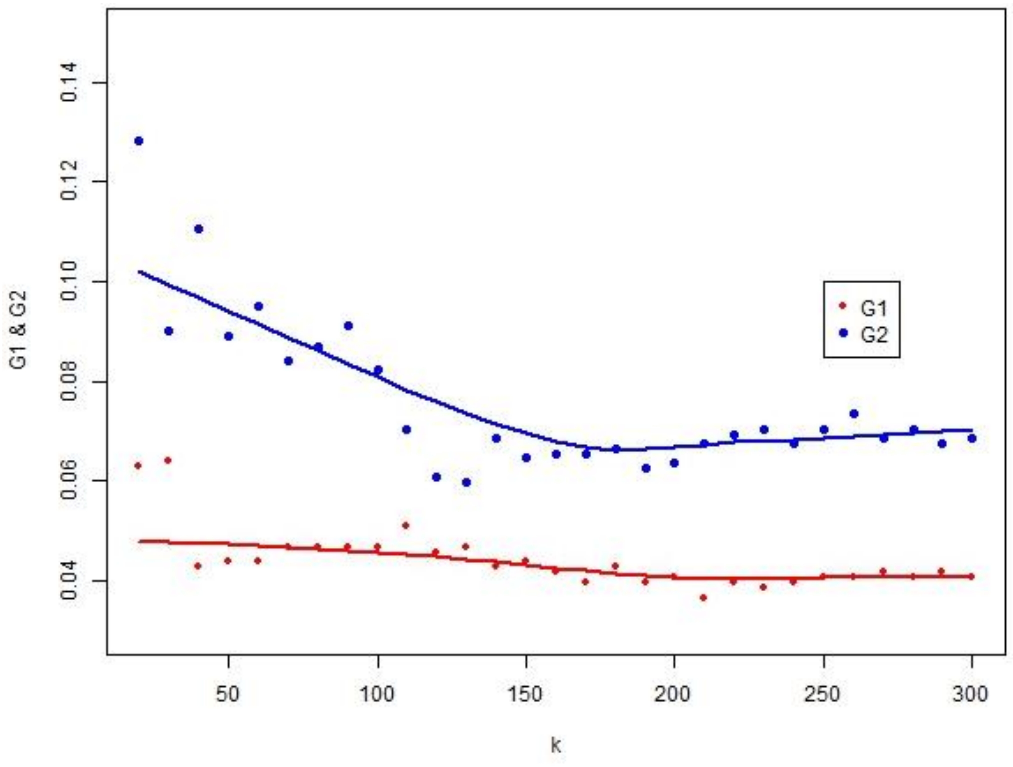
4.3 Modified Combination Method

在之前的分析中，我们知道判别分析对模型有 96. 2%的预测正确性，且对于第三组和第六组的分类是全部正确的。我们在此基础上，改进联立分类，具体如下：

Activity labels (origin)	Modified group labels
1	**1**
2	**1**
3	3
4	**2**
5	**2**
6	6

Table 20: Measures for computing feature vectors

在此基础上，只对 G1 和 G2 进行 PCRA，得到结果如下：



Graph 21: The relationship between wrong rate and k

可见当 k_1 取 210，k_2 取 130 时，正确率达到最小值，分别为 96.4%和 94.4%。而只使用判别分析对两组（1,2 和 4,5 两组）的正确率分别为 96.6%和 92.3%。也就是说，PCRA 对两组的使用不仅能达到判别分析的正确率，甚至在 G2 组中

的性能表现优于判别分析。我们就 k_1=210，k_2=130 的选择下再次运行程序，得到如下结果：

		G1_predict			G2_predict			
G_origin		1	2	3	4	5	6	Recall
	1	**496**	0	0	0	0	0	100.0%
	2	20	**451**	0	0	0	0	95.8%
	3	1	14	**405**	0	0	0	96.4%
	4	0	1	0	**440**	50	0	89.6%
	5	0	0	0	11	**521**	0	97.9%
	6	0	0	0	0	0	**537**	100.0%
	Precision	95.9%	96.8%	100.0%	97.6%	91.2%	100.0%	**96.7%**
Group precision		**96.4%**			**94.4%**			

Table 22: The result of PCRA with modified combination method, k_1=210, k_2=130

可以发现，即使是综合预测率，该改进方法的正确率也略微高于判别分析96.2%的正确率。

4.4 More complex forms of regression

在上一小节的讨论中，我们虽然得到了更好的结果，然而这只是使用 lm()进行线性回归模型上建立的。事实上，像 G2 组中，因变量只取两个值，因此我们还可以采用 Logistic regression，对模型进行更精确的预测。也可以尝试其他更复杂的高阶回归模型来测试预测的正确率。

4.5 Support vector machines(SVM)

Support vector machines(SVM)的思想与聚类分析类似，但是其在 n 维空间进行划分的不是 n-1 维超平面，而是根据样本点局部弯曲的曲面，并且其分割的思想是使分割面与样本点间的距离最近，而不是聚类分析的使样本点间距离最远。
本文使用数据的发布者尝试了使用 CORINNA CORTES 提出的 support vector networks method[4] binary classifiers，并取得了 96%的正确率。其通过 One-Vs-All（OVA）方法处理案例：选择 SVM 超参数，10 倍交叉验证程序(即 jackknife)和高斯内核。
而在 R 中，e2017 package 中的 svm()可以很方便地进行 SVM 分析，在对模型的 summary 中，SVM-Type 项目说明本模型的类别为 C 分类器模型；SVM-Kernel 项目说明本模型所使用的核函数为高斯内积函数且核函数中参数 gamma 的取值为 0.001782531；cost 项目说明本模型确定的约束违反成本为 1。而且我们还可以看到，模型找到了 2330 个支持向量：第一类包含有 518 个支持向量，第二类包含有 571 个支持向量，第三类包含 234 个支持向量，第四类包含有 300 个支持向量，第五类包含有 358 个支持向量，第六类包含 349 个支持向量。最后一行说明模型中的六个类别分别为 1、2、3、4、5、6。
模型的预测结果如下：

	G_predict							
G_origin		1	2	3	4	5	6	Recall
	1	**482**	6	8	0	0	0	97.2%
	2	14	**456**	1	0	0	0	96.8%
	3	6	28	**386**	0	0	0	91.9%
	4	0	1	0	**441**	47	2	89.8%
	5	0	0	0	29	**503**	0	94.5%
	6	0	0	0	0	0	**537**	100.0%
	Precision	96.0%	92.9%	97.7%	93.8%	91.5%	99.6%	**95.2%**

Table 23: The result of SVM

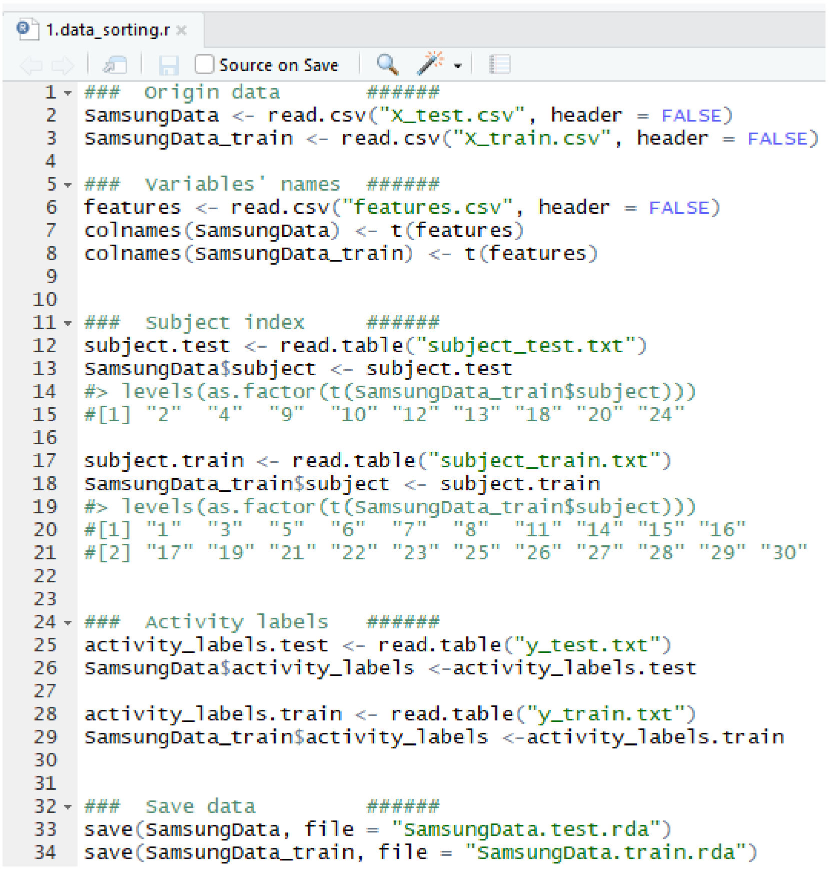
我们使用默认设定，结果得到了 95%的正确率，发现 svm 对第二、五类的分类正确率不高，但对第六类的分类结果较好。若想得到更高的正确率，可以尝试修改函数默认的参数。

5 Conclusions

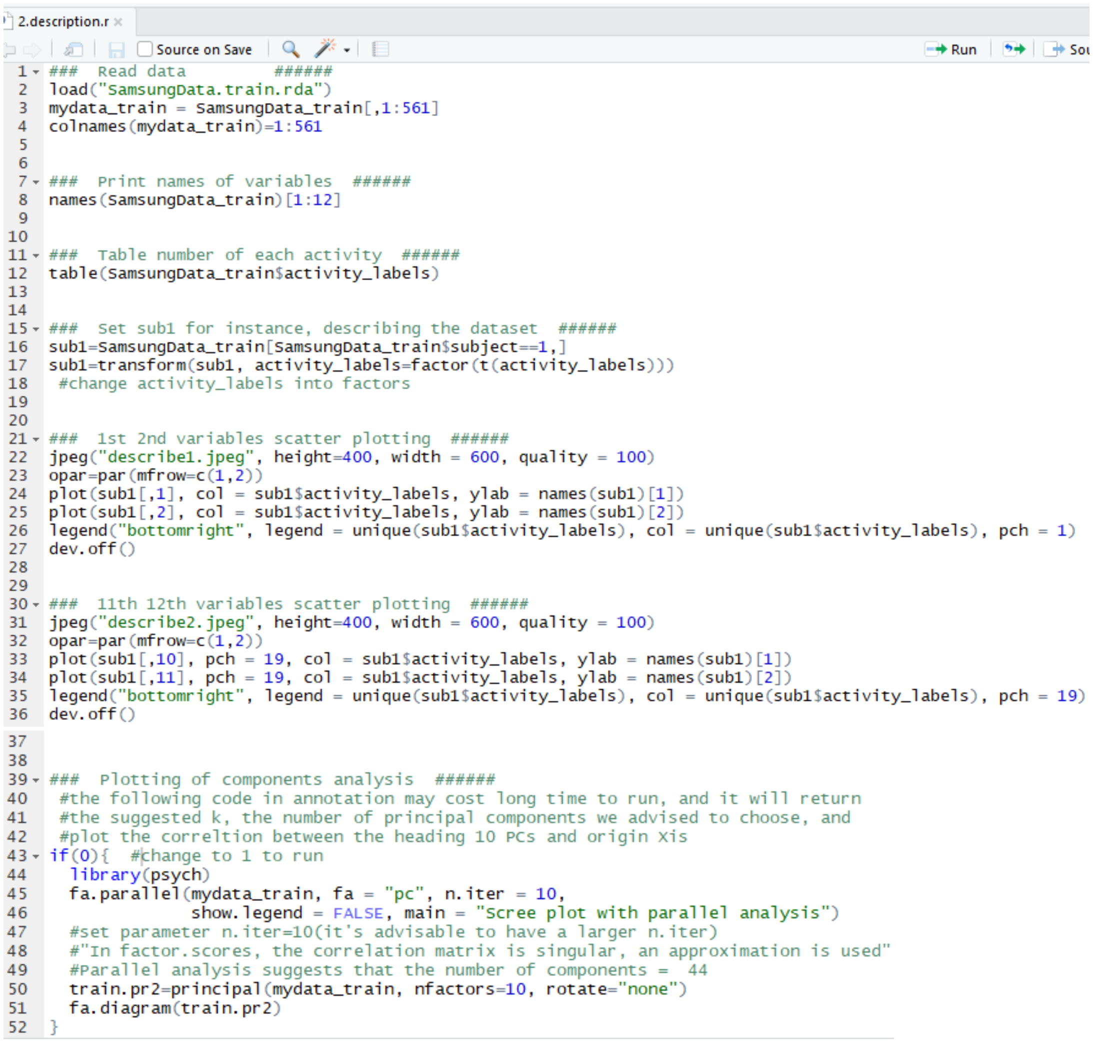
In this paper we introduce the HAR public dataset using smartphones, and try to predict the classification of test samples in several ways. The method based on discriminant classification has achieved great success. At the same time, the results obtained by the modified method of discriminant classification and PCRA can not be neglected.
However, rooms for improvements exist: there are still some mistakes in the classification of various methods, which deserve further study and analysis, modification and improvement of the classification process and try more fitting models. For example, some of the sample points in either 4 or 5 are always confused in many taxonomies, and we can extract these sample points and find out the reason behind the anomaly. Finally, we may conduct further discussions based on computational complexity and model extensibility.

Appendix: Code

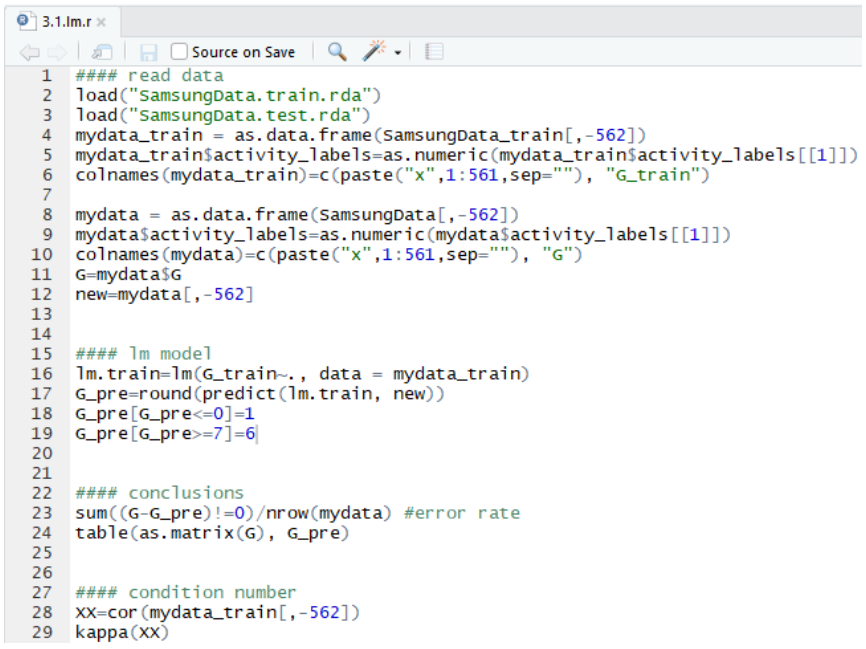
详细 Code 见网址：https://github.com/luweihao/bio-stat.git



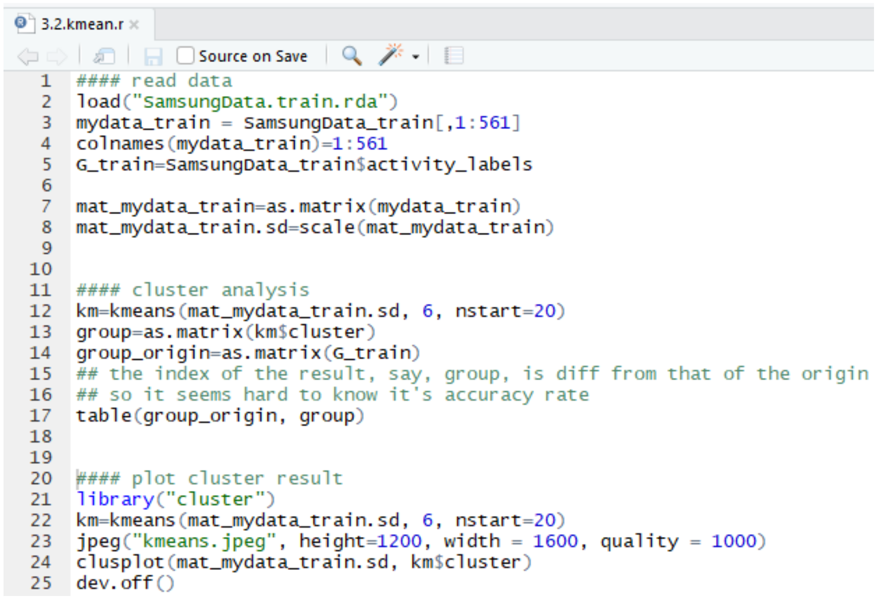
```r
###  Origin data         ######
SamsungData <- read.csv("X_test.csv", header = FALSE)
SamsungData_train <- read.csv("X_train.csv", header = FALSE)

###  Variables' names   ######
features <- read.csv("features.csv", header = FALSE)
colnames(SamsungData) <- t(features)
colnames(SamsungData_train) <- t(features)


###  Subject index      ######
subject.test <- read.table("subject_test.txt")
SamsungData$subject <- subject.test
#> levels(as.factor(t(SamsungData_train$subject)))
#[1] "2"  "4"  "9"  "10" "12" "13" "18" "20" "24"

subject.train <- read.table("subject_train.txt")
SamsungData_train$subject <- subject.train
#> levels(as.factor(t(SamsungData_train$subject)))
#[1] "1"  "3"  "5"  "6"  "7"  "8"  "11" "14" "15" "16"
#[2] "17" "19" "21" "22" "23" "25" "26" "27" "28" "29" "30"


###  Activity labels    ######
activity_labels.test <- read.table("y_test.txt")
SamsungData$activity_labels <-activity_labels.test

activity_labels.train <- read.table("y_train.txt")
SamsungData_train$activity_labels <-activity_labels.train


###  Save data          ######
save(SamsungData, file = "SamsungData.test.rda")
save(SamsungData_train, file = "SamsungData.train.rda")
```



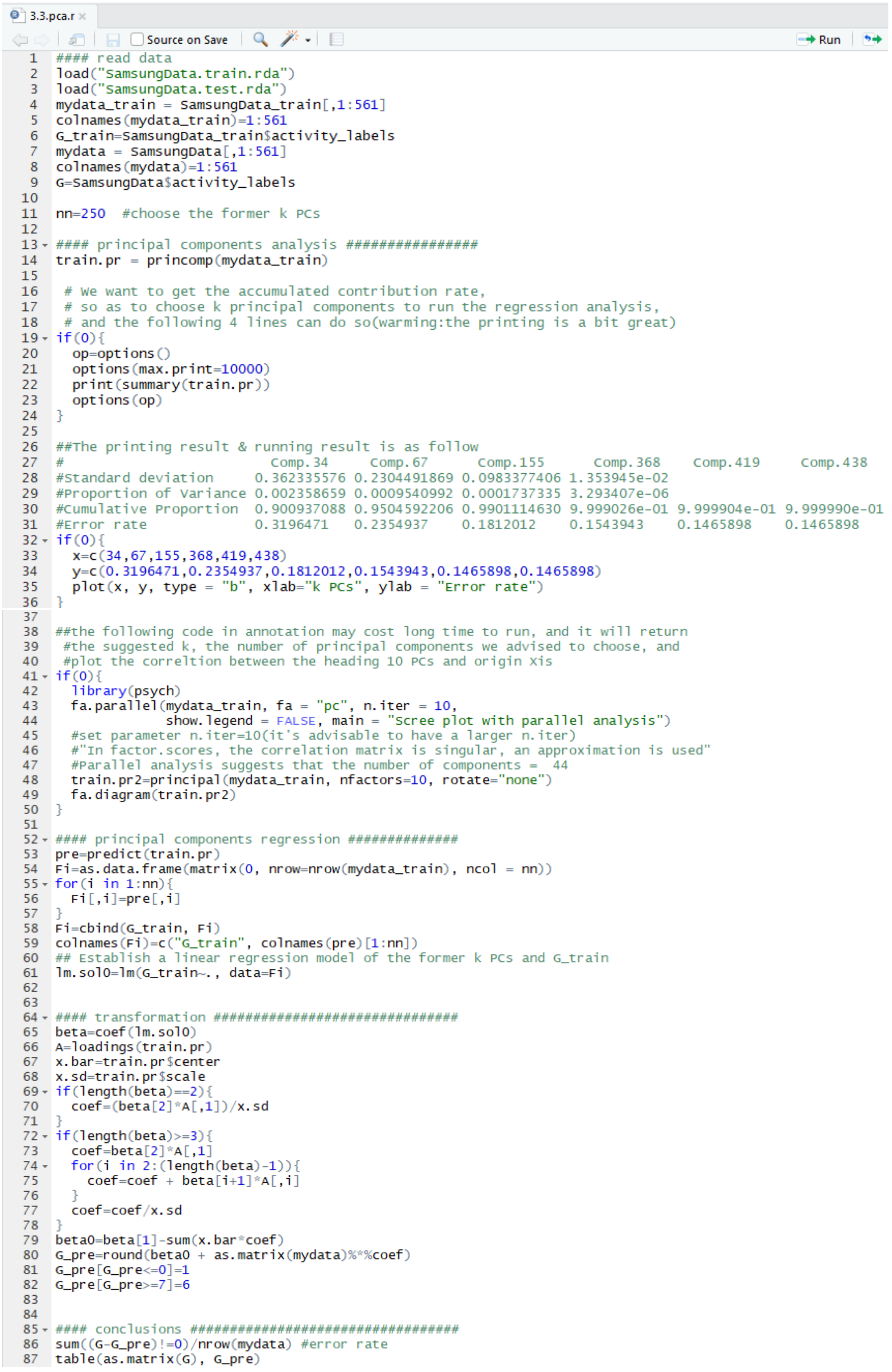
```r
###  Read data          ######
load("SamsungData.train.rda")
mydata_train = SamsungData_train[,1:561]
colnames(mydata_train)=1:561


###  Print names of variables  ######
names(SamsungData_train)[1:12]


###  Table number of each activity  ######
table(SamsungData_train$activity_labels)


###  Set sub1 for instance, describing the dataset  ######
sub1=SamsungData_train[SamsungData_train$subject==1,]
sub1=transform(sub1, activity_labels=factor(t(activity_labels)))
 #change activity_labels into factors


###  1st 2nd variables scatter plotting  ######
jpeg("describe1.jpeg", height=400, width = 600, quality = 100)
opar=par(mfrow=c(1,2))
plot(sub1[,1], col = sub1$activity_labels, ylab = names(sub1)[1])
plot(sub1[,2], col = sub1$activity_labels, ylab = names(sub1)[2])
legend("bottomright", legend = unique(sub1$activity_labels), col = unique(sub1$activity_labels), pch = 1)
dev.off()


###  11th 12th variables scatter plotting  ######
jpeg("describe2.jpeg", height=400, width = 600, quality = 100)
opar=par(mfrow=c(1,2))
plot(sub1[,10], pch = 19, col = sub1$activity_labels, ylab = names(sub1)[1])
plot(sub1[,11], pch = 19, col = sub1$activity_labels, ylab = names(sub1)[2])
legend("bottomright", legend = unique(sub1$activity_labels), col = unique(sub1$activity_labels), pch = 19)
dev.off()


###  Plotting of components analysis  ######
 #the following code in annotation may cost long time to run, and it will return
 #the suggested k, the number of principal components we advised to choose, and
 #plot the correltion between the heading 10 PCs and origin Xis
if(0){  #change to 1 to run
  library(psych)
  fa.parallel(mydata_train, fa = "pc", n.iter = 10,
              show.legend = FALSE, main = "Scree plot with parallel analysis")
  #set parameter n.iter=10(it's advisable to have a larger n.iter)
  #"In factor.scores, the correlation matrix is singular, an approximation is used"
  #Parallel analysis suggests that the number of components =  44
  train.pr2=principal(mydata_train, nfactors=10, rotate="none")
  fa.diagram(train.pr2)
}
```



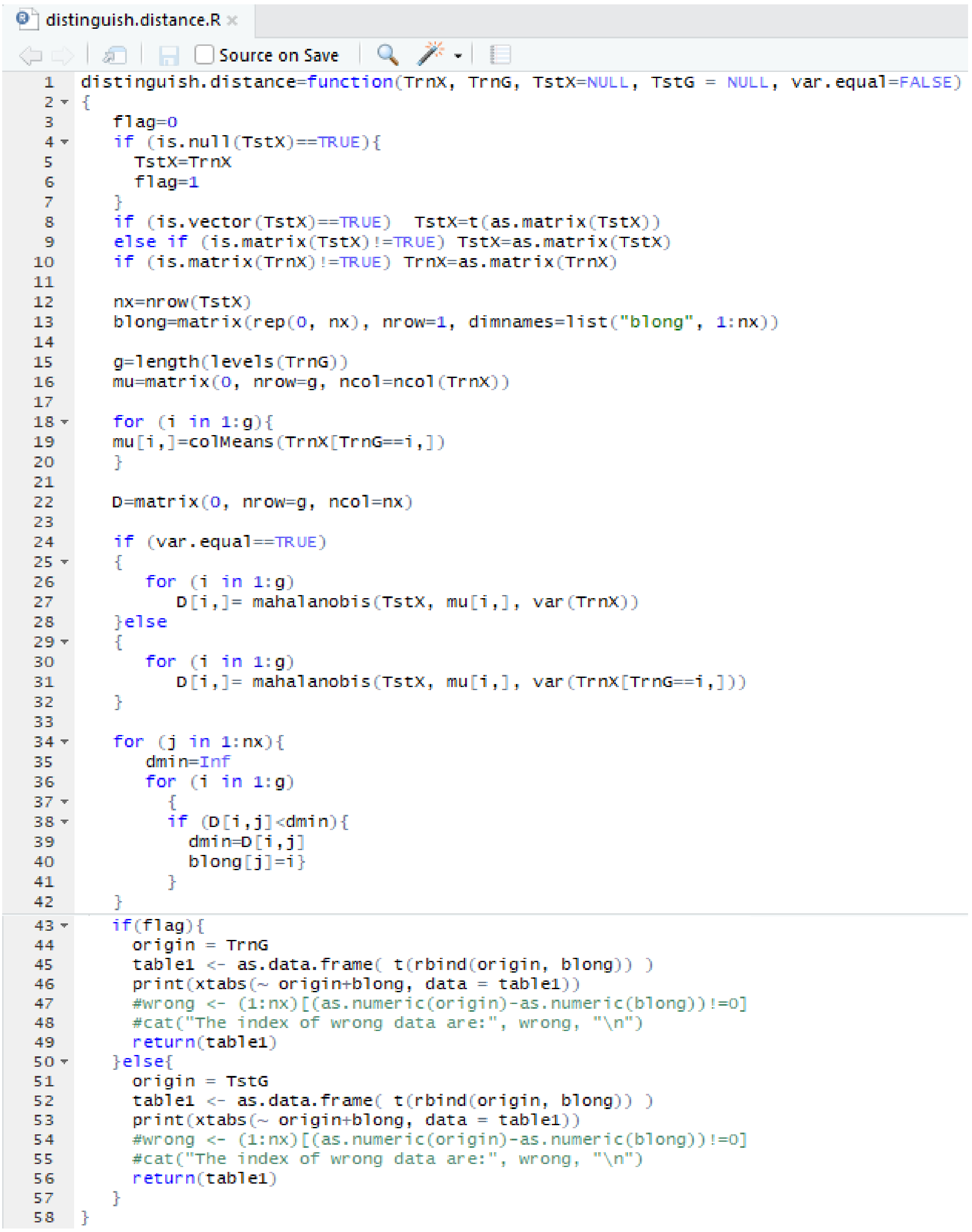
```r
#### read data
load("SamsungData.train.rda")
load("SamsungData.test.rda")
mydata_train = as.data.frame(SamsungData_train[,-562])
mydata_train$activity_labels=as.numeric(mydata_train$activity_labels[[1]])
colnames(mydata_train)=c(paste("x",1:561,sep=""), "G_train")

mydata = as.data.frame(SamsungData[,-562])
mydata$activity_labels=as.numeric(mydata$activity_labels[[1]])
colnames(mydata)=c(paste("x",1:561,sep=""), "G")
G=mydata$G
new=mydata[,-562]


#### lm model
lm.train=lm(G_train~., data = mydata_train)
G_pre=round(predict(lm.train, new))
G_pre[G_pre<=0]=1
G_pre[G_pre>=7]=6


#### conclusions
sum((G-G_pre)!=0)/nrow(mydata) #error rate
table(as.matrix(G), G_pre)


#### condition number
XX=cor(mydata_train[,-562])
kappa(XX)
```



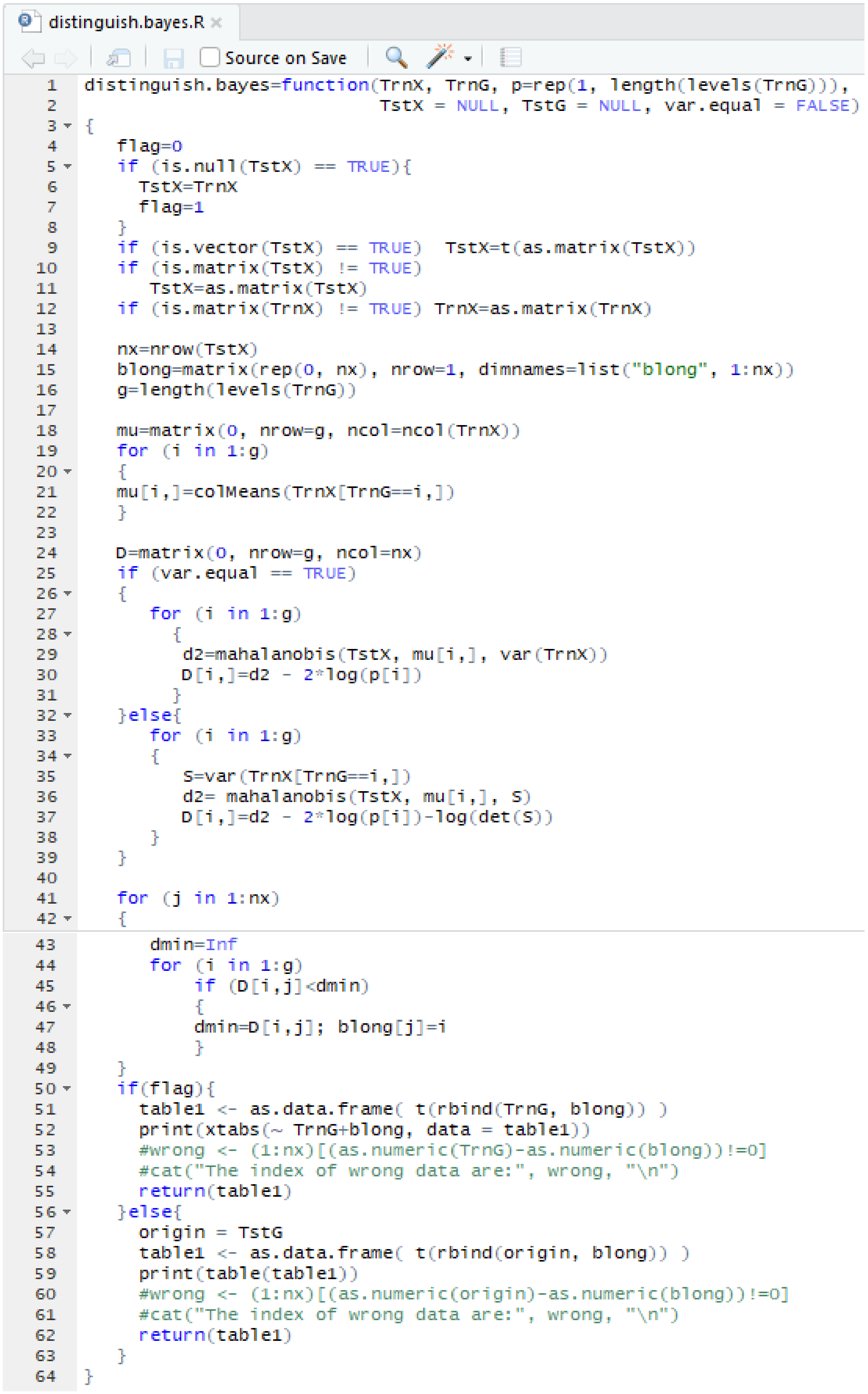
```r
#### read data
load("SamsungData.train.rda")
mydata_train = SamsungData_train[,1:561]
colnames(mydata_train)=1:561
G_train=SamsungData_train$activity_labels

mat_mydata_train=as.matrix(mydata_train)
mat_mydata_train.sd=scale(mat_mydata_train)


#### cluster analysis
km=kmeans(mat_mydata_train.sd, 6, nstart=20)
group=as.matrix(km$cluster)
group_origin=as.matrix(G_train)
## the index of the result, say, group, is diff from that of the origin
## so it seems hard to know it's accuracy rate
table(group_origin, group)


#### plot cluster result
library("cluster")
km=kmeans(mat_mydata_train.sd, 6, nstart=20)
jpeg("kmeans.jpeg", height=1200, width = 1600, quality = 1000)
clusplot(mat_mydata_train.sd, km$cluster)
dev.off()
```



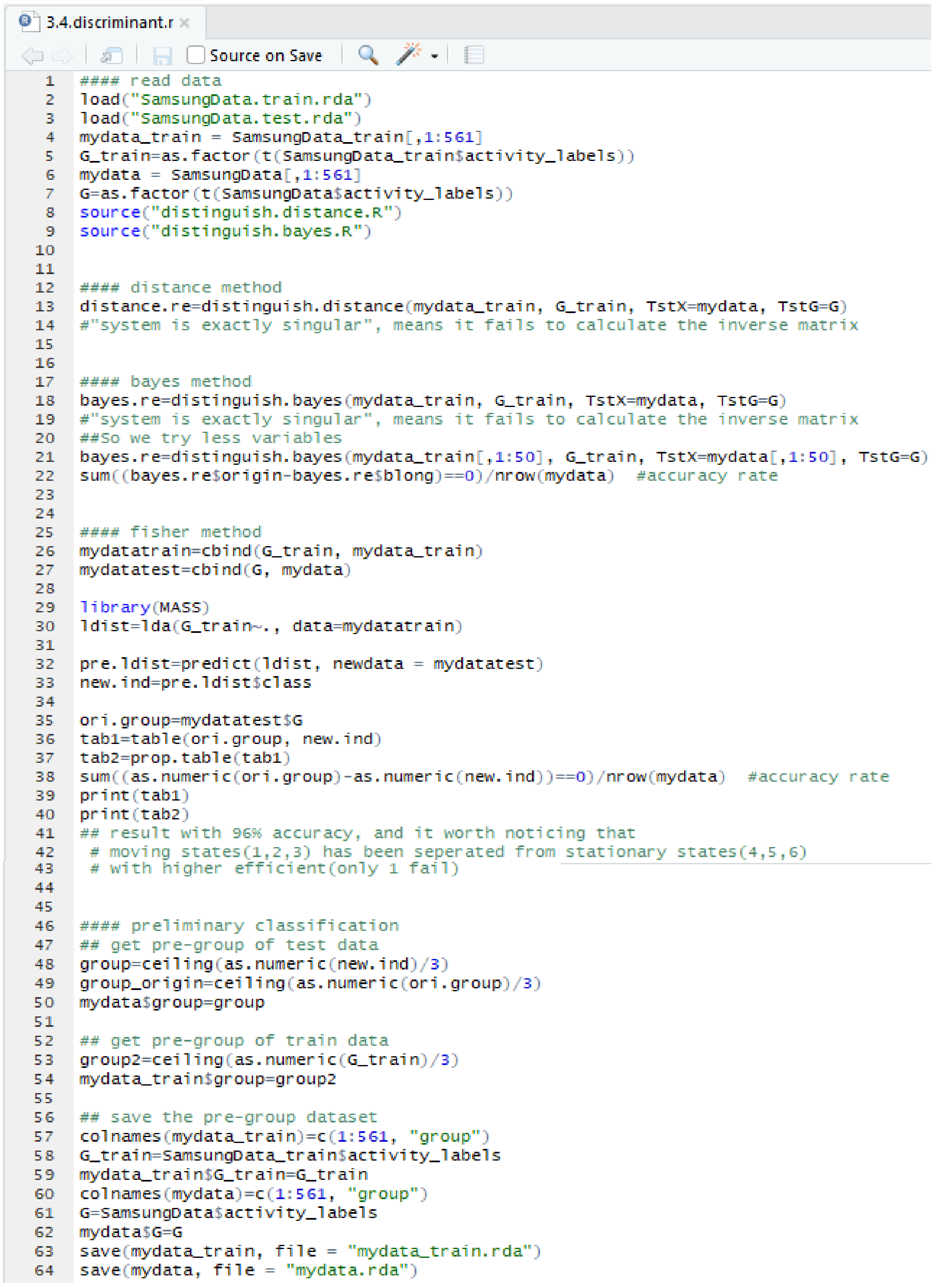
```r
#### read data
load("SamsungData.train.rda")
load("SamsungData.test.rda")
mydata_train = SamsungData_train[,1:561]
colnames(mydata_train)=1:561
G_train=SamsungData_train$activity_labels
mydata = SamsungData[,1:561]
colnames(mydata)=1:561
G=SamsungData$activity_labels

nn=250  #choose the former k PCs

#### principal components analysis ################
train.pr = princomp(mydata_train)

 # We want to get the accumulated contribution rate,
 # so as to choose k principal components to run the regression analysis,
 # and the following 4 lines can do so(warming:the printing is a bit great)
if(0){
  op=options()
  options(max.print=10000)
  print(summary(train.pr))
  options(op)
}

##The printing result & running result is as follow
#                         Comp.34     Comp.67      Comp.155      Comp.368     Comp.419     Comp.438
#Standard deviation     0.362335576 0.2304491869 0.0983377406 1.353945e-02
#Proportion of Variance 0.002358659 0.0009540992 0.0001737335 3.293407e-06
#Cumulative Proportion  0.900937088 0.9504592206 0.9901114630 9.999026e-01 9.999904e-01 9.999990e-01
#Error rate             0.3196471   0.2354937    0.1812012    0.1543943    0.1465898    0.1465898
if(0){
  x=c(34,67,155,368,419,438)
  y=c(0.3196471,0.2354937,0.1812012,0.1543943,0.1465898,0.1465898)
  plot(x, y, type = "b", xlab="k PCs", ylab = "Error rate")
}

##the following code in annotation may cost long time to run, and it will return
 #the suggested k, the number of principal components we advised to choose, and
 #plot the correltion between the heading 10 PCs and origin Xis
if(0){
  library(psych)
  fa.parallel(mydata_train, fa = "pc", n.iter = 10,
              show.legend = FALSE, main = "Scree plot with parallel analysis")
  #set parameter n.iter=10(it's advisable to have a larger n.iter)
  #"In factor.scores, the correlation matrix is singular, an approximation is used"
  #Parallel analysis suggests that the number of components =  44
  train.pr2=principal(mydata_train, nfactors=10, rotate="none")
  fa.diagram(train.pr2)
}

#### principal components regression ##############
pre=predict(train.pr)
Fi=as.data.frame(matrix(0, nrow=nrow(mydata_train), ncol = nn))
for(i in 1:nn){
  Fi[,i]=pre[,i]
}
Fi=cbind(G_train, Fi)
colnames(Fi)=c("G_train", colnames(pre)[1:nn])
## Establish a linear regression model of the former k PCs and G_train
lm.sol0=lm(G_train~., data=Fi)


#### transformation ################################
beta=coef(lm.sol0)
A=loadings(train.pr)
x.bar=train.pr$center
x.sd=train.pr$scale
if(length(beta)==2){
  coef=(beta[2]*A[,1])/x.sd
}
if(length(beta)>=3){
  coef=beta[2]*A[,1]
  for(i in 2:(length(beta)-1)){
    coef=coef + beta[i+1]*A[,i]
  }
  coef=coef/x.sd
}
beta0=beta[1]-sum(x.bar*coef)
G_pre=round(beta0 + as.matrix(mydata)%*%coef)
G_pre[G_pre<=0]=1
G_pre[G_pre>=7]=6


#### conclusions ##################################
sum((G-G_pre)!=0)/nrow(mydata) #error rate
table(as.matrix(G), G_pre)
```



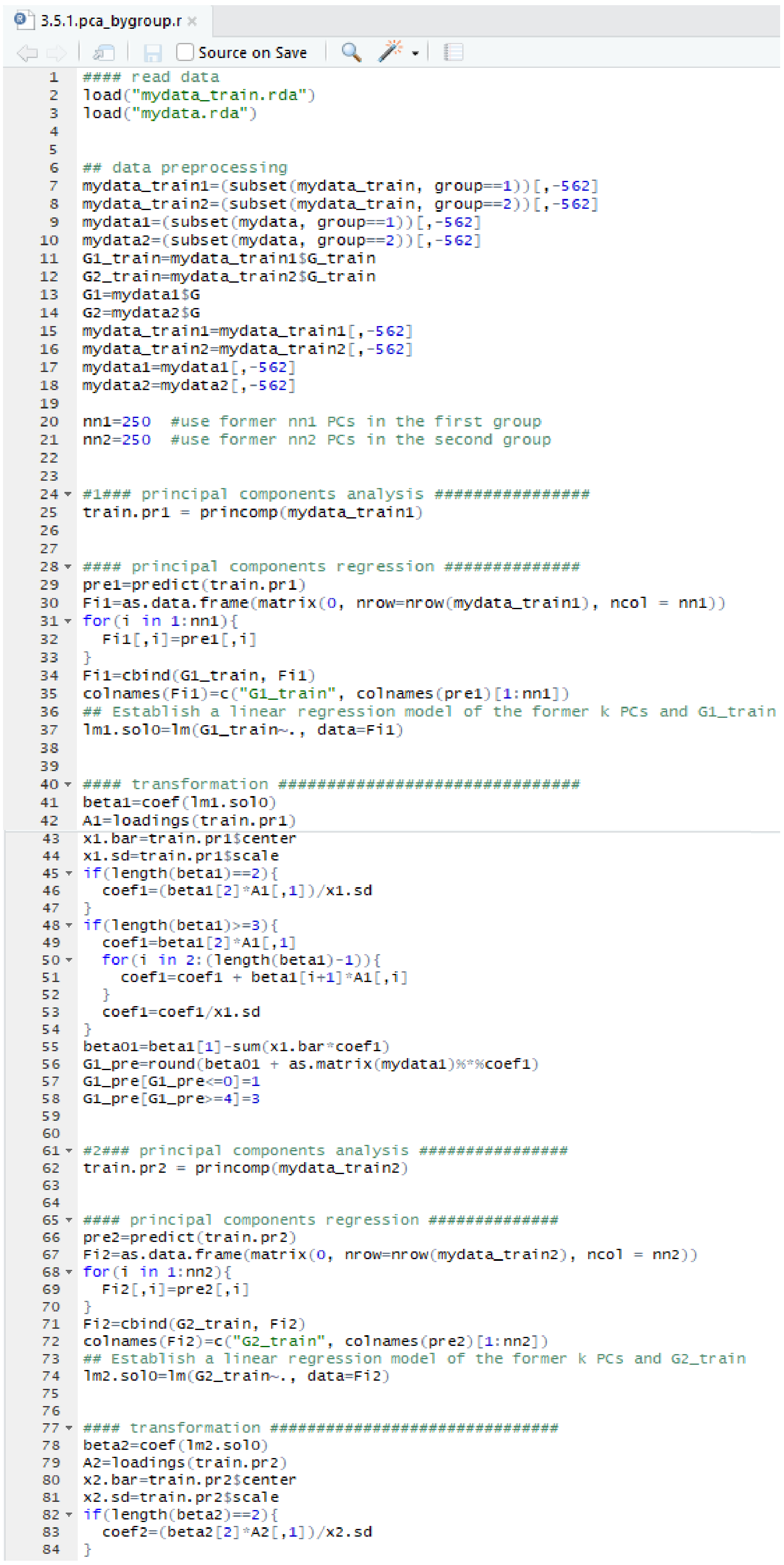
```r
distinguish.distance=function(TrnX, TrnG, TstX=NULL, TstG = NULL, var.equal=FALSE)
{
   flag=0
   if (is.null(TstX)==TRUE){
     TstX=TrnX
     flag=1
   }
   if (is.vector(TstX)==TRUE)  TstX=t(as.matrix(TstX))
   else if (is.matrix(TstX)!=TRUE) TstX=as.matrix(TstX)
   if (is.matrix(TrnX)!=TRUE) TrnX=as.matrix(TrnX)

   nx=nrow(TstX)
   blong=matrix(rep(0, nx), nrow=1, dimnames=list("blong", 1:nx))

   g=length(levels(TrnG))
   mu=matrix(0, nrow=g, ncol=ncol(TrnX))

   for (i in 1:g){
   mu[i,]=colMeans(TrnX[TrnG==i,])
   }

   D=matrix(0, nrow=g, ncol=nx)

   if (var.equal==TRUE)
   {
      for (i in 1:g)
         D[i,]= mahalanobis(TstX, mu[i,], var(TrnX))
   }else
   {
      for (i in 1:g)
         D[i,]= mahalanobis(TstX, mu[i,], var(TrnX[TrnG==i,]))
   }

   for (j in 1:nx){
      dmin=Inf
      for (i in 1:g)
        {
        if (D[i,j]<dmin){
          dmin=D[i,j]
          blong[j]=i}
        }
   }
   if(flag){
     origin = TrnG
     table1 <- as.data.frame( t(rbind(origin, blong)) )
     print(xtabs(~ origin+blong, data = table1))
     #wrong <- (1:nx)[(as.numeric(origin)-as.numeric(blong))!=0]
     #cat("The index of wrong data are:", wrong, "\n")
     return(table1)
   }else{
     origin = TstG
     table1 <- as.data.frame( t(rbind(origin, blong)) )
     print(xtabs(~ origin+blong, data = table1))
     #wrong <- (1:nx)[(as.numeric(origin)-as.numeric(blong))!=0]
     #cat("The index of wrong data are:", wrong, "\n")
     return(table1)
   }
}
```



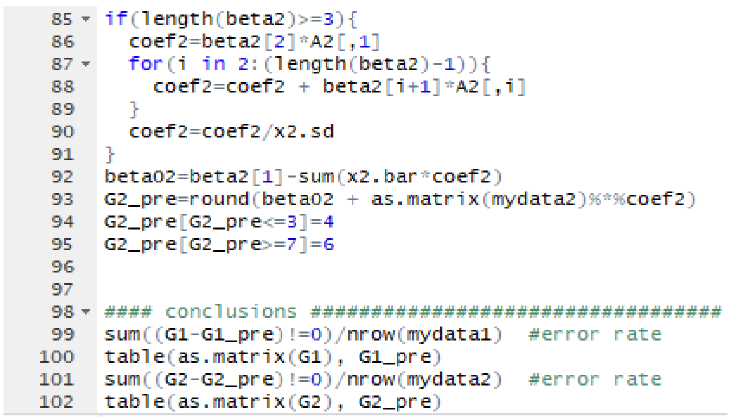
```r
distinguish.bayes=function(TrnX, TrnG, p=rep(1, length(levels(TrnG))),
                           TstX = NULL, TstG = NULL, var.equal = FALSE)
{
   flag=0
   if (is.null(TstX) == TRUE){
     TstX=TrnX
     flag=1
   }
   if (is.vector(TstX) == TRUE)  TstX=t(as.matrix(TstX))
   if (is.matrix(TstX) != TRUE)
      TstX=as.matrix(TstX)
   if (is.matrix(TrnX) != TRUE) TrnX=as.matrix(TrnX)

   nx=nrow(TstX)
   blong=matrix(rep(0, nx), nrow=1, dimnames=list("blong", 1:nx))
   g=length(levels(TrnG))

   mu=matrix(0, nrow=g, ncol=ncol(TrnX))
   for (i in 1:g)
   {
   mu[i,]=colMeans(TrnX[TrnG==i,])
   }

   D=matrix(0, nrow=g, ncol=nx)
   if (var.equal == TRUE)
   {
      for (i in 1:g)
        {
         d2=mahalanobis(TstX, mu[i,], var(TrnX))
         D[i,]=d2 - 2*log(p[i])
        }
   }else{
      for (i in 1:g)
      {
         S=var(TrnX[TrnG==i,])
         d2= mahalanobis(TstX, mu[i,], S)
         D[i,]=d2 - 2*log(p[i])-log(det(S))
      }
   }

   for (j in 1:nx)
   {
      dmin=Inf
      for (i in 1:g)
          if (D[i,j]<dmin)
          {
          dmin=D[i,j]; blong[j]=i
          }
   }
   if(flag){
     table1 <- as.data.frame( t(rbind(TrnG, blong)) )
     print(xtabs(~ TrnG+blong, data = table1))
     #wrong <- (1:nx)[(as.numeric(TrnG)-as.numeric(blong))!=0]
     #cat("The index of wrong data are:", wrong, "\n")
     return(table1)
   }else{
     origin = TstG
     table1 <- as.data.frame( t(rbind(origin, blong)) )
     print(table(table1))
     #wrong <- (1:nx)[(as.numeric(origin)-as.numeric(blong))!=0]
     #cat("The index of wrong data are:", wrong, "\n")
     return(table1)
   }
}
```



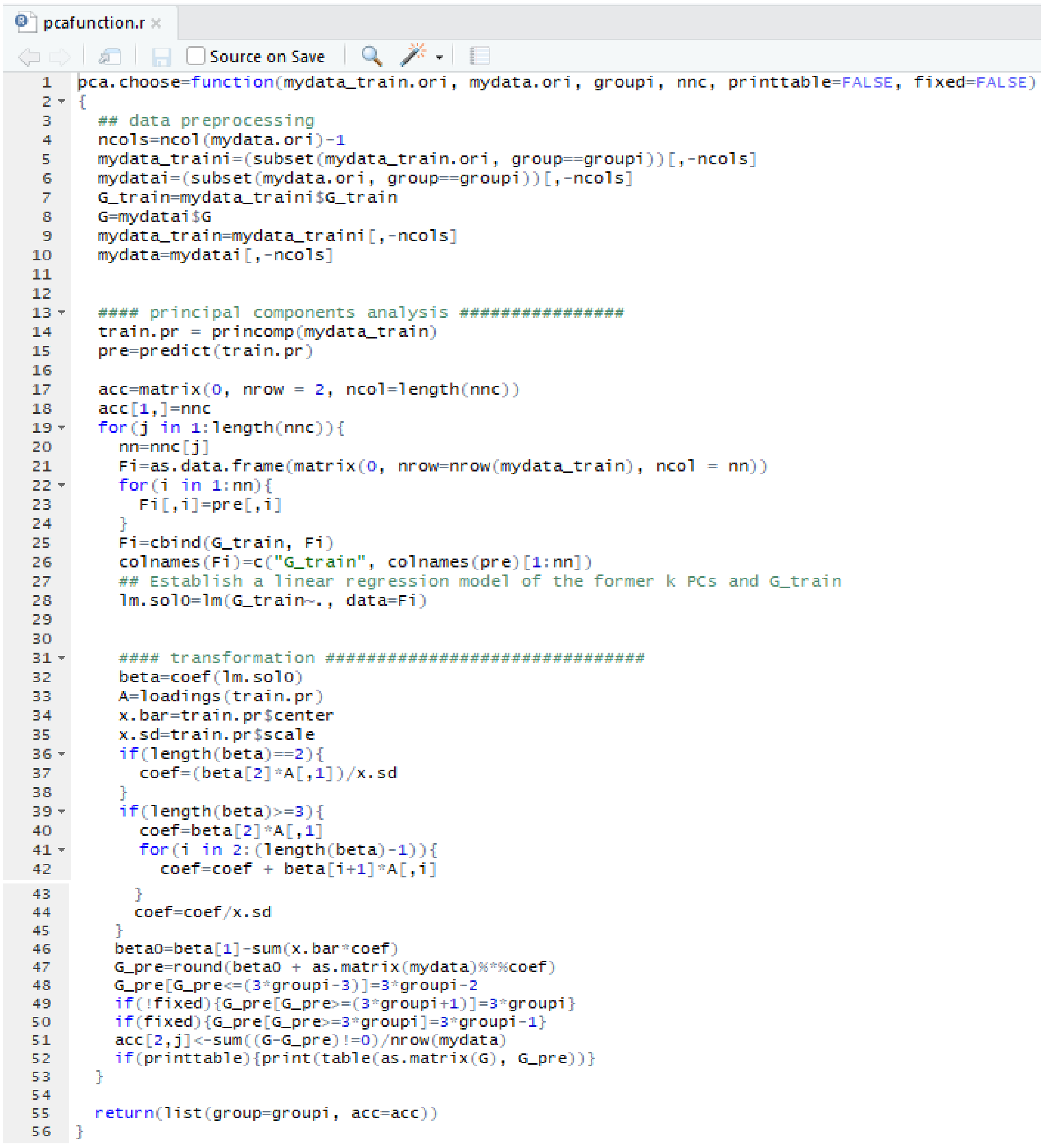
```r
#### read data
load("SamsungData.train.rda")
load("SamsungData.test.rda")
mydata_train = SamsungData_train[,1:561]
G_train=as.factor(t(SamsungData_train$activity_labels))
mydata = SamsungData[,1:561]
G=as.factor(t(SamsungData$activity_labels))
source("distinguish.distance.R")
source("distinguish.bayes.R")


#### distance method
distance.re=distinguish.distance(mydata_train, G_train, TstX=mydata, TstG=G)
#"system is exactly singular", means it fails to calculate the inverse matrix


#### bayes method
bayes.re=distinguish.bayes(mydata_train, G_train, TstX=mydata, TstG=G)
#"system is exactly singular", means it fails to calculate the inverse matrix
##So we try less variables
bayes.re=distinguish.bayes(mydata_train[,1:50], G_train, TstX=mydata[,1:50], TstG=G)
sum((bayes.re$origin-bayes.re$blong)==0)/nrow(mydata)  #accuracy rate


#### fisher method
mydatatrain=cbind(G_train, mydata_train)
mydatatest=cbind(G, mydata)

library(MASS)
ldist=lda(G_train~., data=mydatatrain)

pre.ldist=predict(ldist, newdata = mydatatest)
new.ind=pre.ldist$class

ori.group=mydatatest$G
tab1=table(ori.group, new.ind)
tab2=prop.table(tab1)
sum((as.numeric(ori.group)-as.numeric(new.ind))==0)/nrow(mydata)  #accuracy rate
print(tab1)
print(tab2)
## result with 96% accuracy, and it worth noticing that
 # moving states(1,2,3) has been seperated from stationary states(4,5,6)
 # with higher efficient(only 1 fail)


#### preliminary classification
## get pre-group of test data
group=ceiling(as.numeric(new.ind)/3)
group_origin=ceiling(as.numeric(ori.group)/3)
mydata$group=group

## get pre-group of train data
group2=ceiling(as.numeric(G_train)/3)
mydata_train$group=group2

## save the pre-group dataset
colnames(mydata_train)=c(1:561, "group")
G_train=SamsungData_train$activity_labels
mydata_train$G_train=G_train
colnames(mydata)=c(1:561, "group")
G=SamsungData$activity_labels
mydata$G=G
save(mydata_train, file = "mydata_train.rda")
save(mydata, file = "mydata.rda")
```



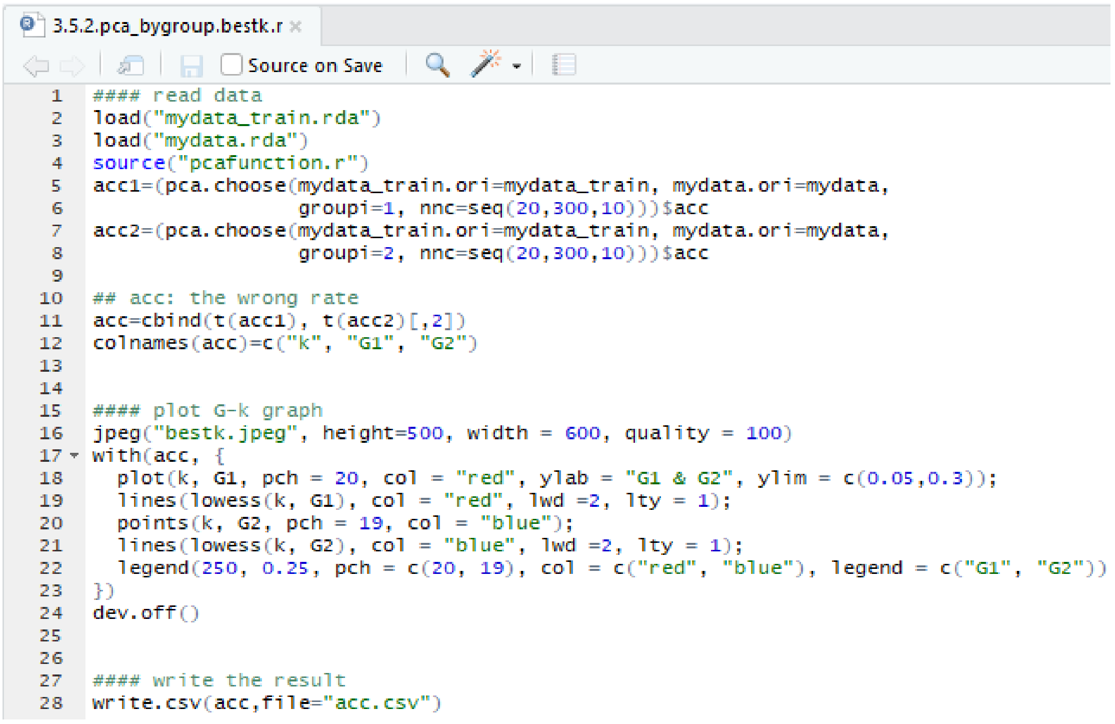
```r
#### read data
load("mydata_train.rda")
load("mydata.rda")


## data preprocessing
mydata_train1=(subset(mydata_train, group==1))[,-562]
mydata_train2=(subset(mydata_train, group==2))[,-562]
mydata1=(subset(mydata, group==1))[,-562]
mydata2=(subset(mydata, group==2))[,-562]
G1_train=mydata_train1$G_train
G2_train=mydata_train2$G_train
G1=mydata1$G
G2=mydata2$G
mydata_train1=mydata_train1[,-562]
mydata_train2=mydata_train2[,-562]
mydata1=mydata1[,-562]
mydata2=mydata2[,-562]

nn1=250  #use former nn1 PCs in the first group
nn2=250  #use former nn2 PCs in the second group


#1### principal components analysis ################
train.pr1 = princomp(mydata_train1)


#### principal components regression ##############
pre1=predict(train.pr1)
Fi1=as.data.frame(matrix(0, nrow=nrow(mydata_train1), ncol = nn1))
for(i in 1:nn1){
  Fi1[,i]=pre1[,i]
}
Fi1=cbind(G1_train, Fi1)
colnames(Fi1)=c("G1_train", colnames(pre1)[1:nn1])
## Establish a linear regression model of the former k PCs and G1_train
lm1.sol0=lm(G1_train~., data=Fi1)


#### transformation ################################
beta1=coef(lm1.sol0)
A1=loadings(train.pr1)
x1.bar=train.pr1$center
x1.sd=train.pr1$scale
if(length(beta1)==2){
  coef1=(beta1[2]*A1[,1])/x1.sd
}
if(length(beta1)>=3){
  coef1=beta1[2]*A1[,1]
  for(i in 2:(length(beta1)-1)){
    coef1=coef1 + beta1[i+1]*A1[,i]
  }
  coef1=coef1/x1.sd
}
beta01=beta1[1]-sum(x1.bar*coef1)
G1_pre=round(beta01 + as.matrix(mydata1)%*%coef1)
G1_pre[G1_pre<=0]=1
G1_pre[G1_pre>=4]=3


#2### principal components analysis ################
train.pr2 = princomp(mydata_train2)


#### principal components regression ##############
pre2=predict(train.pr2)
Fi2=as.data.frame(matrix(0, nrow=nrow(mydata_train2), ncol = nn2))
for(i in 1:nn2){
  Fi2[,i]=pre2[,i]
}
Fi2=cbind(G2_train, Fi2)
colnames(Fi2)=c("G2_train", colnames(pre2)[1:nn2])
## Establish a linear regression model of the former k PCs and G2_train
lm2.sol0=lm(G2_train~., data=Fi2)


#### transformation ################################
beta2=coef(lm2.sol0)
A2=loadings(train.pr2)
x2.bar=train.pr2$center
x2.sd=train.pr2$scale
if(length(beta2)==2){
  coef2=(beta2[2]*A2[,1])/x2.sd
}
```



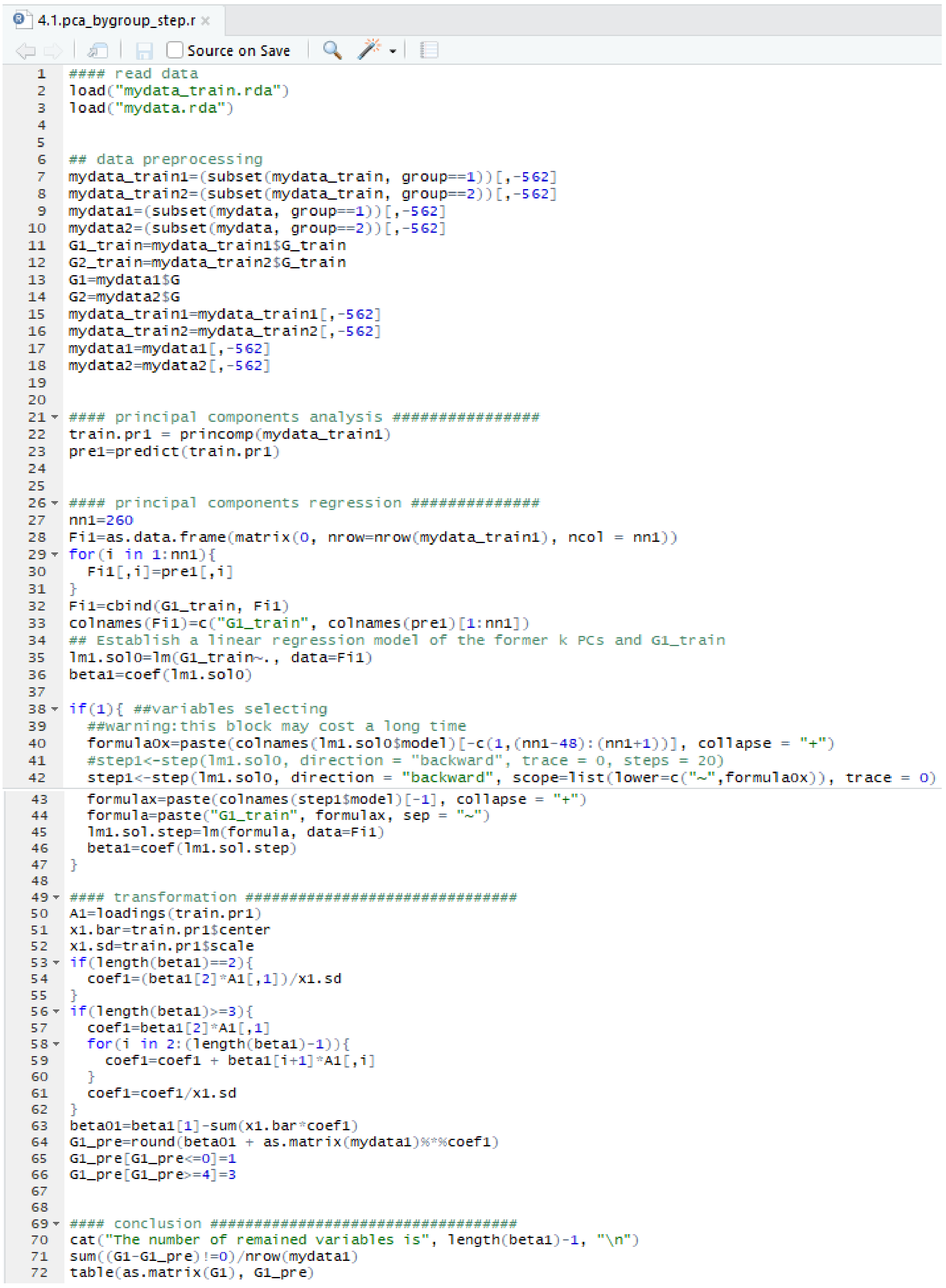
```r
if(length(beta2)>=3){
  coef2=beta2[2]*A2[,1]
  for(i in 2:(length(beta2)-1)){
    coef2=coef2 + beta2[i+1]*A2[,i]
  }
  coef2=coef2/x2.sd
}
beta02=beta2[1]-sum(x2.bar*coef2)
G2_pre=round(beta02 + as.matrix(mydata2)%*%coef2)
G2_pre[G2_pre<=3]=4
G2_pre[G2_pre>=7]=6


#### conclusions ##################################
sum((G1-G1_pre)!=0)/nrow(mydata1)  #error rate
table(as.matrix(G1), G1_pre)
sum((G2-G2_pre)!=0)/nrow(mydata2)  #error rate
table(as.matrix(G2), G2_pre)
```



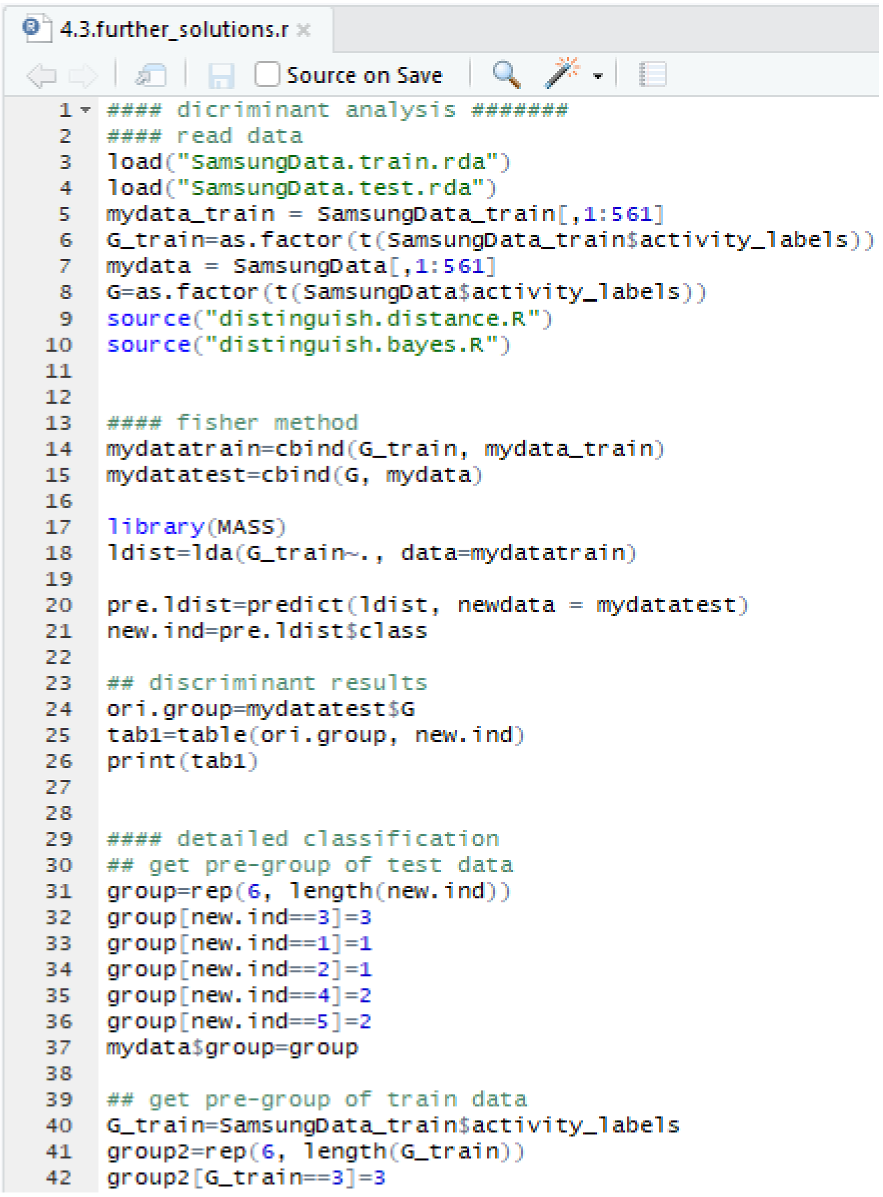
```r
pca.choose=function(mydata_train.ori, mydata.ori, groupi, nnc, printtable=FALSE, fixed=FALSE)
{
  ## data preprocessing
  ncols=ncol(mydata.ori)-1
  mydata_traini=(subset(mydata_train.ori, group==groupi))[,-ncols]
  mydatai=(subset(mydata.ori, group==groupi))[,-ncols]
  G_train=mydata_traini$G_train
  G=mydatai$G
  mydata_train=mydata_traini[,-ncols]
  mydata=mydatai[,-ncols]


  #### principal components analysis ################
  train.pr = princomp(mydata_train)
  pre=predict(train.pr)

  acc=matrix(0, nrow = 2, ncol=length(nnc))
  acc[1,]=nnc
  for(j in 1:length(nnc)){
    nn=nnc[j]
    Fi=as.data.frame(matrix(0, nrow=nrow(mydata_train), ncol = nn))
    for(i in 1:nn){
      Fi[,i]=pre[,i]
    }
    Fi=cbind(G_train, Fi)
    colnames(Fi)=c("G_train", colnames(pre)[1:nn])
    ## Establish a linear regression model of the former k PCs and G_train
    lm.sol0=lm(G_train~., data=Fi)


    #### transformation ###############################
    beta=coef(lm.sol0)
    A=loadings(train.pr)
    x.bar=train.pr$center
    x.sd=train.pr$scale
    if(length(beta)==2){
      coef=(beta[2]*A[,1])/x.sd
    }
    if(length(beta)>=3){
      coef=beta[2]*A[,1]
      for(i in 2:(length(beta)-1)){
        coef=coef + beta[i+1]*A[,i]
      }
      coef=coef/x.sd
    }
    beta0=beta[1]-sum(x.bar*coef)
    G_pre=round(beta0 + as.matrix(mydata)%*%coef)
    G_pre[G_pre<=(3*groupi-3)]=3*groupi-2
    if(!fixed){G_pre[G_pre>=(3*groupi+1)]=3*groupi}
    if(fixed){G_pre[G_pre>=3*groupi]=3*groupi-1}
    acc[2,j]<-sum((G-G_pre)!=0)/nrow(mydata)
    if(printtable){print(table(as.matrix(G), G_pre))}
  }

  return(list(group=groupi, acc=acc))
}
```



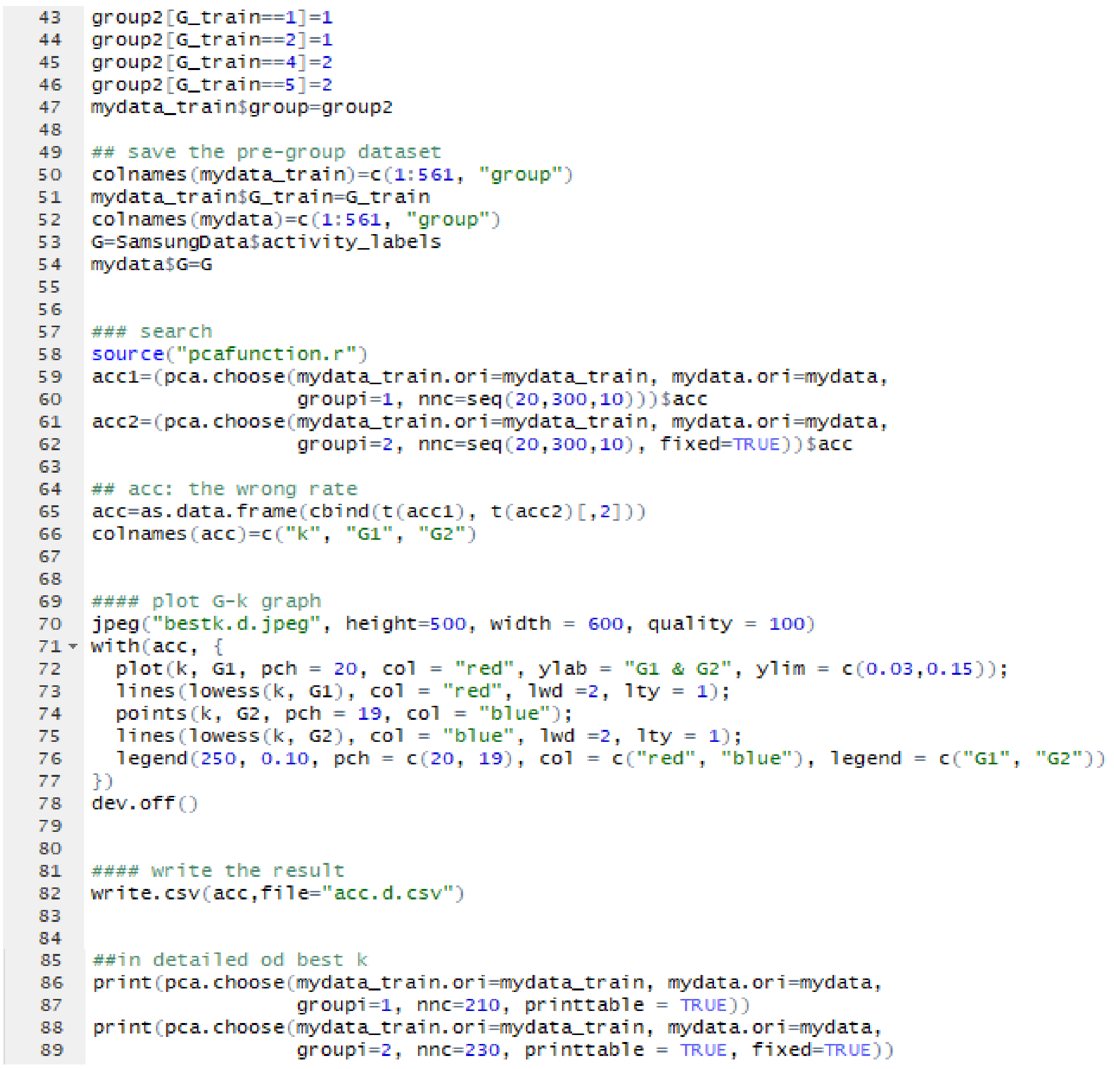
```r
#### read data
load("mydata_train.rda")
load("mydata.rda")
source("pcafunction.r")
acc1=(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=1, nnc=seq(20,300,10)))$acc
acc2=(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=2, nnc=seq(20,300,10)))$acc

## acc: the wrong rate
acc=cbind(t(acc1), t(acc2)[,2])
colnames(acc)=c("k", "G1", "G2")


#### plot G-k graph
jpeg("bestk.jpeg", height=500, width = 600, quality = 100)
with(acc, {
  plot(k, G1, pch = 20, col = "red", ylab = "G1 & G2", ylim = c(0.05,0.3));
  lines(lowess(k, G1), col = "red", lwd =2, lty = 1);
  points(k, G2, pch = 19, col = "blue");
  lines(lowess(k, G2), col = "blue", lwd =2, lty = 1);
  legend(250, 0.25, pch = c(20, 19), col = c("red", "blue"), legend = c("G1", "G2"))
})
dev.off()


#### write the result
write.csv(acc,file="acc.csv")
```



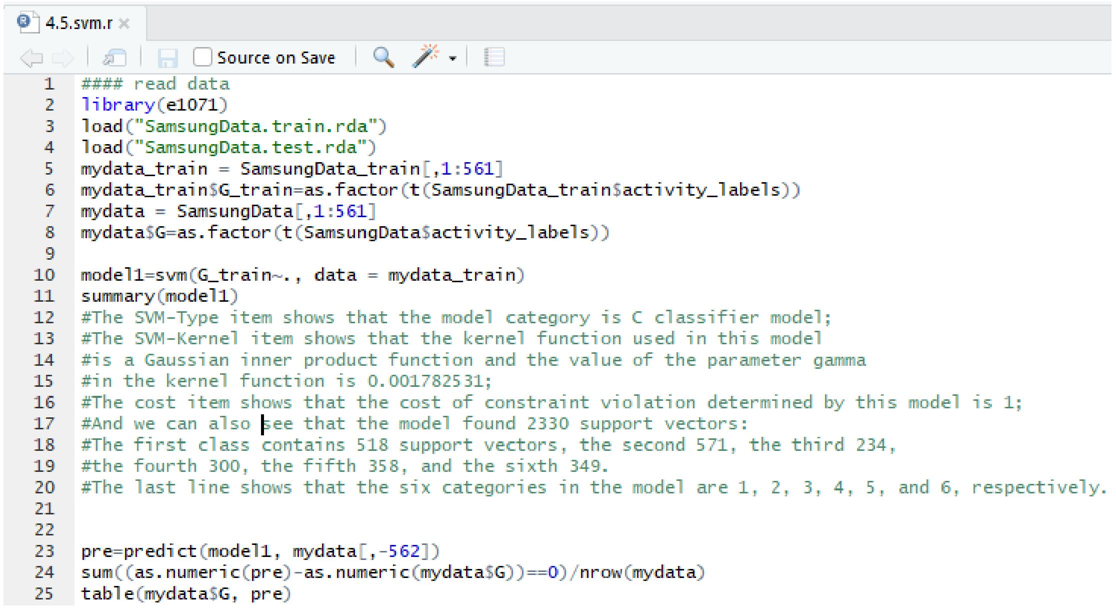
```r
#### read data
load("mydata_train.rda")
load("mydata.rda")


## data preprocessing
mydata_train1=(subset(mydata_train, group==1))[,-562]
mydata_train2=(subset(mydata_train, group==2))[,-562]
mydata1=(subset(mydata, group==1))[,-562]
mydata2=(subset(mydata, group==2))[,-562]
G1_train=mydata_train1$G_train
G2_train=mydata_train2$G_train
G1=mydata1$G
G2=mydata2$G
mydata_train1=mydata_train1[,-562]
mydata_train2=mydata_train2[,-562]
mydata1=mydata1[,-562]
mydata2=mydata2[,-562]


#### principal components analysis ################
train.pr1 = princomp(mydata_train1)
pre1=predict(train.pr1)


#### principal components regression ##############
nn1=260
Fi1=as.data.frame(matrix(0, nrow=nrow(mydata_train1), ncol = nn1))
for(i in 1:nn1){
  Fi1[,i]=pre1[,i]
}
Fi1=cbind(G1_train, Fi1)
colnames(Fi1)=c("G1_train", colnames(pre1)[1:nn1])
## Establish a linear regression model of the former k PCs and G1_train
lm1.sol0=lm(G1_train~., data=Fi1)
beta1=coef(lm1.sol0)

if(1){ ##variables selecting
  ##warning:this block may cost a long time
  formula0x=paste(colnames(lm1.sol0$model)[-c(1,(nn1-48):(nn1+1))], collapse = "+")
  #step1<-step(lm1.sol0, direction = "backward", trace = 0, steps = 20)
  step1<-step(lm1.sol0, direction = "backward", scope=list(lower=c("~",formula0x)), trace = 0)
  formulax=paste(colnames(step1$model)[-1], collapse = "+")
  formula=paste("G1_train", formulax, sep = "~")
  lm1.sol.step=lm(formula, data=Fi1)
  beta1=coef(lm1.sol.step)
}

#### transformation ################################
A1=loadings(train.pr1)
x1.bar=train.pr1$center
x1.sd=train.pr1$scale
if(length(beta1)==2){
  coef1=(beta1[2]*A1[,1])/x1.sd
}
if(length(beta1)>=3){
  coef1=beta1[2]*A1[,1]
  for(i in 2:(length(beta1)-1)){
    coef1=coef1 + beta1[i+1]*A1[,i]
  }
  coef1=coef1/x1.sd
}
beta01=beta1[1]-sum(x1.bar*coef1)
G1_pre=round(beta01 + as.matrix(mydata1)%*%coef1)
G1_pre[G1_pre<=0]=1
G1_pre[G1_pre>=4]=3


#### conclusion ####################################
cat("The number of remained variables is", length(beta1)-1, "\n")
sum((G1-G1_pre)!=0)/nrow(mydata1)
table(as.matrix(G1), G1_pre)
```



```r
#### dicriminant analysis #######
#### read data
load("SamsungData.train.rda")
load("SamsungData.test.rda")
mydata_train = SamsungData_train[,1:561]
G_train=as.factor(t(SamsungData_train$activity_labels))
mydata = SamsungData[,1:561]
G=as.factor(t(SamsungData$activity_labels))
source("distinguish.distance.R")
source("distinguish.bayes.R")


#### fisher method
mydatatrain=cbind(G_train, mydata_train)
mydatatest=cbind(G, mydata)

library(MASS)
ldist=lda(G_train~., data=mydatatrain)

pre.ldist=predict(ldist, newdata = mydatatest)
new.ind=pre.ldist$class

## discriminant results
ori.group=mydatatest$G
tab1=table(ori.group, new.ind)
print(tab1)


#### detailed classification
## get pre-group of test data
group=rep(6, length(new.ind))
group[new.ind==3]=3
group[new.ind==1]=1
group[new.ind==2]=1
group[new.ind==4]=2
group[new.ind==5]=2
mydata$group=group

## get pre-group of train data
G_train=SamsungData_train$activity_labels
group2=rep(6, length(G_train))
group2[G_train==3]=3
```



```r
group2[G_train==1]=1
group2[G_train==2]=1
group2[G_train==4]=2
group2[G_train==5]=2
mydata_train$group=group2

## save the pre-group dataset
colnames(mydata_train)=c(1:561, "group")
mydata_train$G_train=G_train
colnames(mydata)=c(1:561, "group")
G=SamsungData$activity_labels
mydata$G=G


### search
source("pcafunction.r")
acc1=(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=1, nnc=seq(20,300,10)))$acc
acc2=(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=2, nnc=seq(20,300,10), fixed=TRUE))$acc

## acc: the wrong rate
acc=as.data.frame(cbind(t(acc1), t(acc2)[,2]))
colnames(acc)=c("k", "G1", "G2")


#### plot G-k graph
jpeg("bestk.d.jpeg", height=500, width = 600, quality = 100)
with(acc, {
  plot(k, G1, pch = 20, col = "red", ylab = "G1 & G2", ylim = c(0.03,0.15));
  lines(lowess(k, G1), col = "red", lwd =2, lty = 1);
  points(k, G2, pch = 19, col = "blue");
  lines(lowess(k, G2), col = "blue", lwd =2, lty = 1);
  legend(250, 0.10, pch = c(20, 19), col = c("red", "blue"), legend = c("G1", "G2"))
})
dev.off()


#### write the result
write.csv(acc,file="acc.d.csv")


##in detailed od best k
print(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=1, nnc=210, printtable = TRUE))
print(pca.choose(mydata_train.ori=mydata_train, mydata.ori=mydata,
                 groupi=2, nnc=230, printtable = TRUE, fixed=TRUE))
```



```r
#### read data
library(e1071)
load("SamsungData.train.rda")
load("SamsungData.test.rda")
mydata_train = SamsungData_train[,1:561]
mydata_train$G_train=as.factor(t(SamsungData_train$activity_labels))
mydata = SamsungData[,1:561]
mydata$G=as.factor(t(SamsungData$activity_labels))

model1=svm(G_train~., data = mydata_train)
summary(model1)
#The SVM-Type item shows that the model category is C classifier model;
#The SVM-Kernel item shows that the kernel function used in this model
#is a Gaussian inner product function and the value of the parameter gamma
#in the kernel function is 0.001782531;
#The cost item shows that the cost of constraint violation determined by this model is 1;
#And we can also see that the model found 2330 support vectors:
#The first class contains 518 support vectors, the second 571, the third 234,
#the fourth 300, the fifth 358, and the sixth 349.
#The last line shows that the six categories in the model are 1, 2, 3, 4, 5, and 6, respectively.


pre=predict(model1, mydata[,-562])
sum((as.numeric(pre)-as.numeric(mydata$G))==0)/nrow(mydata)
table(mydata$G, pre)
```

References

[1] Jonathon Shlens, A Tutorial on Principal Component Analysis Mountain View, CA 94043, 2014

[2] Pierre Comon, Independent component analysis, A new concept? Signal Processing 36 (1994) 287-314, 1992

[3] J.R. Kwapisz, G.M. Weiss, and S.A. Moore. Activity recognition using cell phone accelerometers. SIGKDD Explorations Newsletter, 12(2):74–82, 2011.

[4] Davide Anguita, Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine, 2012 http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[5] C. Cortes and V. Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995