Basilico Eleonora, Caporali Francesco, Malgieri Luigi Maria
16 May 2023

UNIVERSITÀ DI TORINO

Multivariate Statistical Analysis
Problem Set 2

Exercise 1

Consider the data set `psych`, which contains 24 psychological tests ($t_i, \forall i \in \{1, \dots, 24\}$) administered to 301 students, with ages ranging from 11 to 16, in a suburb of Chicago:

- the 1st group is made of 156 students (74 boys, 82 girls) from the *Pasteur School*;
- the 2nd group is made of 145 students (72 boys, 73 girls) from the *Grant-White School*.

```
psych_0 = read.table("data/psych.txt", header = T)
dim_p = dim(psych_0)
colnames(psych_0) = c(c("case", "sex", "age"), paste0("t_", 1:(dim_p[2] - 4)), "group")
psych_0[2] = tolower(unlist(psych_0[2]))
psych_0[28] = tolower(unlist(psych_0[28]))
```

case	sex	age	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_10	t_11	t_12	t_13
1	m	13.1	20	31	12	3	40	7	23	22	9	78	74	115	229
2	f	13.6	32	21	12	17	34	5	12	22	9	87	84	125	285
3	f	13.1	27	21	12	15	20	3	7	12	3	75	49	78	159
4	m	13.2	32	31	16	24	42	8	18	21	17	69	65	106	175
5	f	12.2	29	19	12	7	37	8	16	25	18	85	63	126	213
6	f	14.1	32	20	11	18	31	3	12	25	6	100	92	133	270

t_14	t_15	t_16	t_17	t_18	t_19	t_20	t_21	t_22	t_23	t_24	group
170	86	96	6	9	16	3	14	34	5	24	pasteur
184	85	100	12	12	10	-3	13	21	1	12	pasteur
170	85	95	1	5	6	-3	9	18	7	20	pasteur
181	80	91	5	3	10	-2	10	22	6	19	pasteur
187	99	104	15	14	14	29	15	19	4	20	pasteur
164	84	104	6	6	14	9	2	16	10	22	pasteur

The 24 tests correspond to the following subjects:

	test
t_1	visual perception
t_2	cubes
t_3	paper form board
t_4	flags
t_5	general information
t_6	paragraph comprehension
t_7	sentence completion
t_8	word classification
t_9	word meaning

	test
t_10	addition
t_11	code
t_12	counting dots
t_13	straight-curved capitals
t_14	word recognition
t_15	number recognition
t_16	figure recognition
t_17	object-number
t_18	number-figure
t_19	figure-word
t_20	deduction
t_21	numerical puzzles
t_22	problem reasoning
t_23	series completion
t_24	arithmetic problems

Note that the variable `case` does not provide any important information as it only corresponds to an enumeration of the students, who were tested in sequential order (containing some gaps probably due to the absence of data for some of the students).

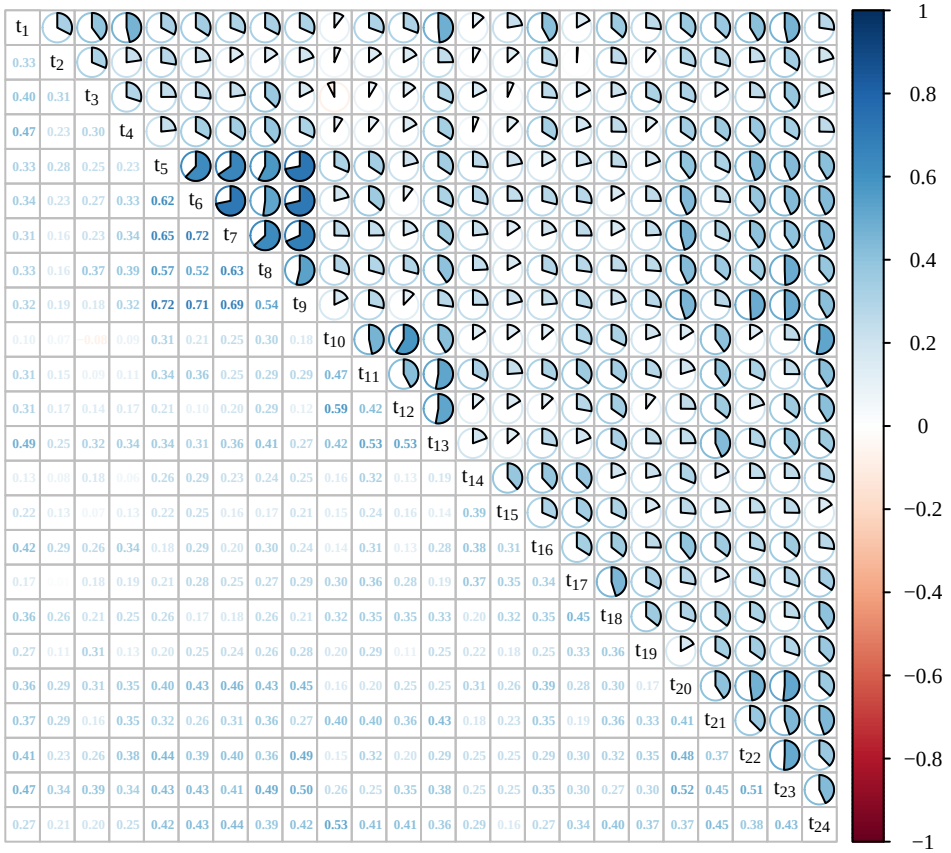
1.1

In performing the factor analysis we are only interested in the 24 variables corresponding to the psychological tests, hence we remove the variables `case`, `age` and `sex` from our dataset. Moreover, we are asked to use only the Grant-White students data, so we subset the remaining data frame according to the request.

```
psych_1 = psych_0[, 4:28]
gw = subset(psych_1, group == "grant", select = -group)
```

Before fitting the model, we scale our data and examine the correlation matrix. Indeed correlations between variables play a central role in Factor Analysis. Since we have a very large number of variables, we choose not to display the values of the matrix directly, but we rather visualize them with a plot.

```
gws = scale(gw)
cor_gws = cor(gws)
dim_gws = dim(gws)
colnames(cor_gws) = paste0("$t[", 1:(dim_p[2] - 4), "]")
rownames(cor_gws) = colnames(cor_gws)
par(family = "serif")
corrplot.mixed(cor_gws, upper = "pie",
    number.cex = 0.4, tl.col = "black", tl.cex = 0.7, cl.cex = 0.7)
```




```
neg_cor_gws = ((24^2 - sum(sign(cor_gws))) / 2) / 2
```

From the correlation matrix we can immediatly note that:

- all the correlations except for `neg_cor_gws` $= 1$ are positive (`neg_cor_gws` computes the number of total negative correlations in the matrix, without multiplicity. See the computation above). This is reasonable since all the variables represent psychological tests measuring different kinds of cognitive abilities, which of course are more or less related to one another;
- by just looking at the correlation matrix, it is difficult to guess whether 5 or 6 common factors are an appropriate choice or not.

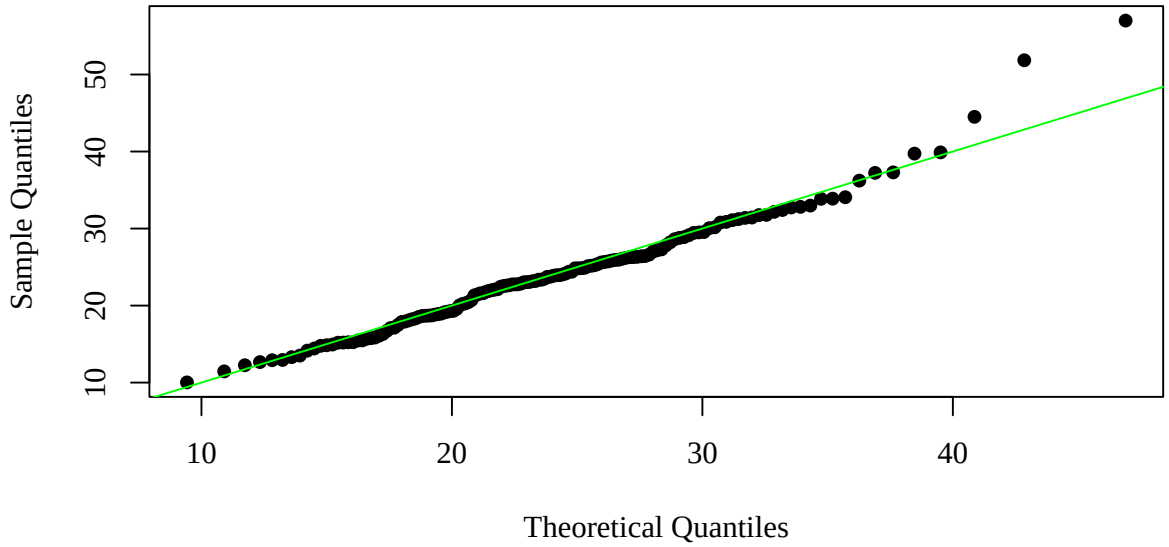
In order to obtain the maximum likelihood solution for $m = 5$ and $m = 6$ factors in `R` we can use the built-in function `factanal()`.
Before proceeding with the computation, we would like to recall that the *maximum likelihood* method, unlike the *principal component method*, relies on the necessary assumption of normality of the *common factors* ($\boldsymbol{F}$) and of the *specific error terms* ($\boldsymbol{\varepsilon}$). In particular, if $\boldsymbol{F} = (F_1, \ldots, F_m)$ and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_p)$ are normally distributed, then

$$\boldsymbol{X} = \boldsymbol{L}\boldsymbol{F} + \boldsymbol{\varepsilon} \sim \mathcal{N}(\mu, \Sigma), \text{ with } \boldsymbol{L} \in \mathbb{R}^{p \times m}.$$

We can check the normality by observing that our input data $\boldsymbol{x} \in \mathbb{R}^{24}$, which was previously rescaled, actually comes from a $\boldsymbol{X} \sim \mathcal{N}(0, I)$.

For this purpose we look at the Q-Q plot of the squared Mahalanobis distances *vs* a χ^2_{24}.

```
par(family = "serif", mar = c(4, 4, 1, 1))
d = mahalanobis(gws, center = colMeans(gws), cov = cov(gws))
plot(qchisq(ppoints(d), df = ncol(gws)), sort(d), pch = 16,
    xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")
abline(0, 1, col = "green")
```



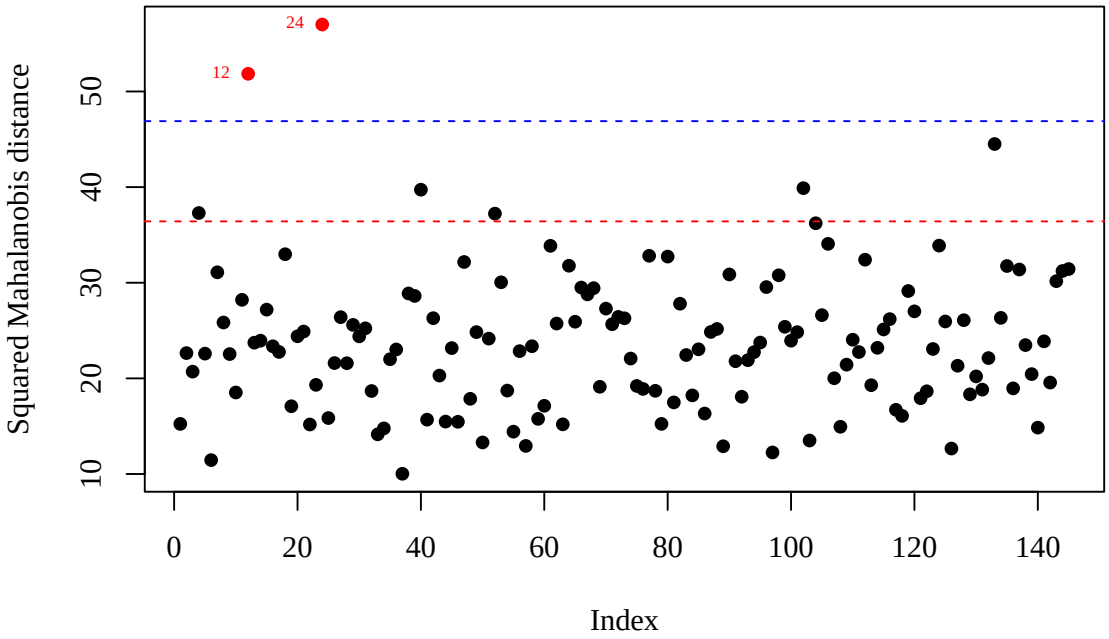
The plot shows that the variables jointly seem to follow a gaussian behaviour: except for the last 3 points, which create a heavy right tail the other points lie on the Q-Q line.

Moreover, we can test the multivariate normality through statistical tests in order to be able to state with greater accuracy if our hypotheses are satisfied. In particular, we use the Mardia test implemented in `R` inside the library `MVN`: `mvn(_, mvnTest = "mardia")`.

```
test_gws = mvn(data = gws, mvnTest = "mardia")
test_gws$multivariateNormality$Test = c("mardia_skewness", "mardia_kurtosis", "mvn")
names(test_gws$multivariateNormality) = c("test", "stat", "p-value", "result")
test_gws$multivariateNormality$result = tolower(test_gws$multivariateNormality$result)
test_gws$multivariateNormality
```

test	stat	p-value	result
mardia_skewness	2867.01367856929	0.000163876133958305	no
mardia_kurtosis	1.24485915986261	0.213183525332604	yes
mvn	NA	NA	no

The results show that our data appear to fail the multivariate gaussianity test. This could be due to the presence of some possible outliers, hence we try to remove them and do the test again. In order to identify the outliers of the multivariate distribution, we plot the vector of the squared Mahalanobis distances and then add two threshold lines corresponding to different levels of the theoretical quantiles of a χ^2_{24}. In particular, we use $\alpha_1 = 0.95$ (in red (■)) and $\alpha_2 = \frac{(n-0.5)}{n}$ with $n =$ `nrow(gws)` (in blue (■)).



After removing the two most extreme outliers (corresponding to the 12$^{\text{th}}$ an the 24$^{\text{th}}$ students), we obtain the following results:

```
out = c(12, 24)
test_gws_out = mvn(data = gws[-out, ], mvnTest = "mardia")
```

test	stat	p-value	result
mardia_skewness	2663.88953618774	0.187254232636786	yes
mardia_kurtosis	-0.639916415536184	0.522226941149228	yes
mvn	NA	NA	yes

We can therefore conclude that it is reasonable to assume that our variables are normally distributed since the presence of 2 outliers on 145 is amply justifiable considering the source of our data (i.e. some tests).

We can now proceed with the computation of the maximum likelihood solution, first with $m = 5$ factors, then with $m = 6$ factors (without any rotation):

```
faml_5 = factanal(gws, factors = 5, rotation = "none")
load_5 = faml_5$loadings[, ]
```

	Factor1	Factor2	Factor3	Factor4	Factor5
t_1	0.5549	-0.0032	0.4659	-0.1495	0.0015
t_2	0.3444	-0.0287	0.2917	-0.0563	0.1250
t_3	0.3734	-0.1422	0.4267	-0.1045	0.0418
t_4	0.4634	-0.1044	0.3032	-0.1128	0.1482
t_5	0.7226	-0.2536	-0.2249	-0.0756	-0.0044

	Factor1	Factor2	Factor3	Factor4	Factor5
t_6	0.7208	-0.3742	-0.1685	-0.0139	-0.1453
t_7	0.7278	-0.3355	-0.2323	-0.1317	0.0131
t_8	0.6917	-0.1442	-0.0421	-0.1066	0.0801
t_9	0.7232	-0.4245	-0.1967	0.0169	-0.0214
t_10	0.5182	0.6034	-0.3795	0.0411	0.1158
t_11	0.5701	0.3495	-0.0240	0.0649	-0.3670
t_12	0.4872	0.5444	0.0052	-0.1179	0.1277
t_13	0.6305	0.3467	0.2011	-0.3833	-0.2058
t_14	0.3929	-0.0013	0.0648	0.3688	-0.2378
t_15	0.3456	0.0268	0.1282	0.3678	-0.1281
t_16	0.4559	0.0247	0.3781	0.2755	-0.0855
t_17	0.4530	0.1283	0.0333	0.4382	-0.1130
t_18	0.4749	0.2521	0.2182	0.2588	0.0177
t_19	0.4179	0.0511	0.1376	0.1964	-0.0669
t_20	0.5961	-0.1672	0.1806	0.1546	0.2271
t_21	0.5741	0.2267	0.1539	0.0252	0.1590
t_22	0.5946	-0.1395	0.1803	0.1287	0.0982
t_23	0.6650	-0.0636	0.2131	0.0332	0.2445
t_24	0.6571	0.1864	-0.1262	0.1451	0.1292

```
faml_6 = factanal(gws, factors = 6, rotation = "none")
load_6 = faml_6$loadings[, ]
```

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
t_1	0.5486	0.0039	0.4562	-0.1968	-0.0599	0.0333
t_2	0.3388	-0.0273	0.3009	-0.1585	0.0715	0.2322
t_3	0.3725	-0.1392	0.4443	-0.1107	0.0336	-0.2323
t_4	0.4600	-0.1066	0.3043	-0.1332	0.1257	-0.0871
t_5	0.7243	-0.2605	-0.2170	-0.0734	-0.0261	0.0920
t_6	0.7240	-0.3674	-0.1557	0.0278	-0.1458	0.0009
t_7	0.7329	-0.3539	-0.2340	-0.0919	0.0229	-0.1481
t_8	0.6953	-0.1550	-0.0401	-0.1049	0.0908	-0.2071
t_9	0.7277	-0.4211	-0.1804	0.0539	-0.0332	0.0922
t_10	0.5131	0.5871	-0.3853	-0.0239	0.1601	0.0291
t_11	0.5786	0.3898	-0.0434	0.0797	-0.4217	0.1269
t_12	0.4816	0.5361	-0.0146	-0.1655	0.1279	-0.1027
t_13	0.6175	0.3280	0.1533	-0.3573	-0.2278	-0.1395
t_14	0.3978	0.0305	0.0803	0.3532	-0.1307	0.0058
t_15	0.3494	0.0578	0.1457	0.3323	-0.0393	0.0969
t_16	0.4568	0.0562	0.3879	0.2097	-0.0402	0.0753
t_17	0.4744	0.1802	0.0697	0.5696	0.0082	-0.2565
t_18	0.4783	0.2777	0.2330	0.2208	0.0730	0.0072
t_19	0.4218	0.0713	0.1544	0.1842	-0.0259	-0.0171
t_20	0.5961	-0.1556	0.2009	0.0750	0.2310	0.0915
t_21	0.5706	0.2318	0.1513	-0.0958	0.1371	0.2158
t_22	0.5970	-0.1208	0.1977	0.0858	0.0702	0.1688
t_23	0.6616	-0.0583	0.2287	-0.0376	0.2257	0.0688
t_24	0.6561	0.1904	-0.1127	0.0757	0.1584	0.0672

It is remarkable that in the case $m = 5$ all but two variables load on the first factor higher than on any other.

This makes any factor interpretation very difficult, at least without applying any rotation to the loadings. We will discuss it in more detail in the next point.

Then we proceed with the computation of the proportion of total sample variance due to each factor. We recall that the proportion of total sample variance due to the k^{th} factor is defined as

$$\text{prop\_var}(k) = \frac{\sum_{j=1}^{p} \hat{l}_{j,k}^2}{\text{trace}\,(\boldsymbol{S})},$$

with $\hat{\boldsymbol{L}} = \left(\hat{l}_{j,k}\right)_{\substack{j=1,\dots p \\ k=1,\dots,m}}$ factor loadings and $\boldsymbol{S}$ sample covariance matrix.
Due to the scaling performed at the beginning of the computation in our case $\text{trace}\,(\boldsymbol{S}) = \texttt{size(}\boldsymbol{S}\texttt{)} = 24$ (it is indeed a sample correlation matrix).

```
prop_var_5 = colSums(load_5^2) / dim_gws[2]
```

	Factor1	Factor2	Factor3	Factor4	Factor5
prop_var_5	0.3159	0.0698	0.0548	0.04	0.0223

```
prop_var_6 = colSums(load_6^2) / dim_gws[2]
```

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
prop_var_6	0.3168	0.0711	0.0563	0.0417	0.0212	0.0175

We could get the associated cumulative proportion of total sample variance by applying the `cumsum()` function to the previous 2 variables. However, these computations are also performed as a part of the output of the command `factanal()`, together with the sum of the squares of the loadings:

```
faml_5
```

	Factor1	Factor2	Factor3	Factor4	Factor5
ss_load_5	7.5813	1.6743	1.3161	0.9589	0.5351
prop_var_5	0.3159	0.0698	0.0548	0.0400	0.0223
cum_var_5	0.3159	0.3856	0.4405	0.4804	0.5027

```
faml_6
```

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
ss_load_6	7.6024	1.7068	1.3515	1.0000	0.5086	0.4192
prop_var_6	0.3168	0.0711	0.0563	0.0417	0.0212	0.0175
cum_var_6	0.3168	0.3879	0.4442	0.4859	0.5071	0.5245

Both models seem to fit very poorly. A general criterion for the choice of the number of factors is to take the smallest m such that the total proportion of variance due to the m factors is at least 80%. However, in both our cases ($m = 5, 6$), the models explain about 50% (respectively 50.27% and 52.45%) of the total variance collectively. Hence, the result is not satisfactory.

Next, as requested, we report below the specific variances $(\psi_j)_{j=1}^{24}$, again for both $m = 5$ and $m = 6$. In this case we directly exploit the output of `factanal()` in order not to have to recalculate the values of the specific variances of the factors by hand. We report the results of the computation below:

```
psi_5 = faml_5$uniquenesses
```

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_10	t_11	t_12
0.4526	0.7766	0.6456	0.6477	0.3573	0.2907	0.2863	0.4812	0.2573	0.2082	0.4134	0.4361
t_13	t_14	t_15	t_16	t_17	t_18	t_19	t_20	t_21	t_22	t_23	t_24
0.2525	0.6489	0.7118	0.5654	0.5724	0.596	0.7607	0.5086	0.5695	0.5683	0.4474	0.4799

```
psi_6 = faml_6$uniquenesses
```

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_10	t_11	t_12
0.4474	0.7098	0.5771	0.6433	0.346	0.2946	0.2519	0.4288	0.2481	0.2164	0.3111	0.4262
t_13	t_14	t_15	t_16	t_17	t_18	t_19	t_20	t_21	t_22	t_23	t_24
0.2885	0.6925	0.732	0.5864	0.3473	0.5857	0.7583	0.5128	0.5233	0.5491	0.4494	0.4853

Finally, we need to assess the accuracy of the approximations of the correlation matrices. For this purpose, for both models we analyse the residual matrix given by the difference between the actual correlation matrix, $\boldsymbol{R}$, and the correlation matrix given by the approximation performed by the maximum likelihood method, i.e. $\boldsymbol{S} = \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^T + \hat{\boldsymbol{\Psi}}$, where $\hat{\boldsymbol{\Psi}} = \text{diag}\left((\psi_j)_{j=1}^{24}\right)$.
Let us compare the squared Frobenius norms of the approximation matrices in order to see which approximation is more accurate.

```
residual_5 = cor_gws - (load_5 %*% t(load_5) + diag(psi_5))
ss_residual_5 = sum(residual_5^2)
residual_6 = cor_gws - (load_6 %*% t(load_6) + diag(psi_6))
ss_residual_6 = sum(residual_6^2)
comparison = c(ss_residual_5, ss_residual_6)
```

	ss_residual_5	ss_residual_6
comparison	0.7335	0.602

It is evident that in both cases the approximation error of the correlation matrix is not negligible, however the second one is slightly lowe then the first one.

Another possibile way to see if 5 or 6 factors are enough to explain the observed covariances is to consider a test which is performed automatically by the command `factanal()`. The p-value of this test is displayed at the end of the output. We obtain respectively:

```
faml_5$PVAL
faml_6$PVAL
```

p-value_5	p-value_6
0.1558614	0.2642373

Let us briefly explain the meaning of the test performed above.
The function uses the model's likelihood estimation to check the quality of the fitting of our factors and it does it by testing

$$H_0 : \boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{\Psi} \quad vs \quad H_1 : \boldsymbol{\Sigma} \text{ generic positive definite matrix.}$$

Both our models have high p-values (> 0.05), hence it seems that the number of factors is reasonable in both cases.

In conclusion, both choices are acceptable, but in some sense inaccurate. The improvement given by the choice of $m = 6$ factors is not particularly significant, hence we tend to prefer $m = 5$. Indeed the additional factor obtained with $m = 6$ accounts only for the 1.75% of the total sample variance and the squared Frobenius norms of the residual matrices share the same order of magnitude.

1.2

We now want to give an interpretation to the common factors in the $m = 5$ solution. It is important to recall that factor loadings are identified up to a rotation. Thus, different rotations yield different factors which lead to different interpretations. A wise choice of the rotation can therefore ease the analysis significantly. For example, in the previous point, when $m = 5$ almost all variables load on the first factor higher than on the other four factors, revealing that the *uniqueness condition* through which the model was fit yields a poor choice of the rotation as for the understanding the factors. It is preferable to perform instead the `Varimax` rotation. It identifies the rotation matrix $\boldsymbol{T}$, thus the rotated loadings $\hat{\boldsymbol{L}}^* = \hat{\boldsymbol{L}}\boldsymbol{T}$. This is optimal in the following sense:

$$\max \left\{ \frac{1}{m} \sum_{k=1}^{m} \left[\frac{1}{p} \sum_{j=1}^{p} \left(\tilde{l}_{jk}^* \right)^4 - \left(\frac{1}{p} \sum_{j=1}^{p} \left(\tilde{l}_{jk}^* \right)^2 \right)^2 \right] \right\},$$

where $\forall j = 1, \dots, p,\ k = 1, \dots, m$, we denote $\tilde{l}_{jk}^* = \frac{\hat{l}_{jk}^*}{\sqrt{\hat{h}_j}}$ and $\hat{h}_j = \left\| \left(\hat{l}_{j,k} \right)_{k=1}^{m} \right\|_2$ (communality of the j^{th} variable).

We try to give a simple explaination of what we are maximizing: first we consider, for each factor k, the vector of their square loadings, normalized by the variable communality. We then consider the sample variance of such vectors and we maximize their average over all factors. This procedure tries to concentrate the mass of each column into a few points close to the extremes 0 and 1, and doing it "democratically" for all factors. This rotation is optimal to identify which variables are most affected by a specific factor and which are not, which is our main goal. Moreover, is it reasonable that, in doing so, the procedure also disperses the mass *row-wise*, so that each variable is high only on one or few factors. In the following we will favour this type of analysis, putting for each variable more emphasis on the one factor that is most influencial for it.

```
faml_5_var = factanal(gws, factors = 5, rotation = "varimax")
load_5_var = faml_5_var$loadings[, ]
```

	Factor1	Factor2	Factor3	Factor4	Factor5
t_1	0.1654	0.6549	0.1250	0.1810	0.2066
t_2	0.1079	0.4416	0.0871	0.0954	0.0024
t_3	0.1341	0.5595	-0.0473	0.1115	0.0934
t_4	0.2305	0.5333	0.0895	0.0811	0.0124
t_5	0.7383	0.1893	0.1916	0.1486	0.0547
t_6	0.7724	0.1867	0.0318	0.2477	0.1243
t_7	0.7983	0.2140	0.1427	0.0883	0.0502
t_8	0.5710	0.3429	0.2391	0.1275	0.0423
t_9	0.8079	0.2024	0.0332	0.2188	-0.0072
t_10	0.1807	-0.1082	0.8451	0.1803	0.0264
t_11	0.1952	0.0661	0.4233	0.4365	0.4177
t_12	0.0297	0.2322	0.6944	0.1022	0.1285
t_13	0.1863	0.4329	0.4793	0.0775	0.5382
t_14	0.1846	0.0614	0.0443	0.5522	0.0797
t_15	0.1043	0.1223	0.0586	0.5089	-0.0028

	Factor1	Factor2	Factor3	Factor4	Factor5
t_16	0.0698	0.4061	0.0559	0.5087	0.0540
t_17	0.1543	0.0716	0.2104	0.5947	-0.0269
t_18	0.0323	0.2999	0.3219	0.4576	0.0043
t_19	0.1563	0.2209	0.1440	0.3785	0.0451
t_20	0.3728	0.4614	0.1265	0.2930	-0.1939
t_21	0.1717	0.3980	0.4312	0.2382	-0.0004
t_22	0.3637	0.4232	0.1139	0.3204	-0.0689
t_23	0.3615	0.5421	0.2482	0.2307	-0.1147
t_24	0.3680	0.1786	0.4952	0.3208	-0.0683

We choose not to visualize the results in a plot since there are too many factors and variables and therefore it would not have been helpful.

After the rotation, things become a little better: as expected, the loadings are in general smaller or larger than the previous ones, and this facilitates the interpretation of the factors. In particular:

1. the variables t_5, t_6, t_7, t_8 and t_9 load highly on the first common factor. The psychological tests associated to these variables primarly assess the language-related capacities of an individual, including reading comprehension, vocabulary knowledge, word associations, sentence construction and general knowledge. Hence, we can interpret the first factor as *verbal ability*;
2. the second factor is determined by the variables from t_1 to t_4 togheter with t_{20}, t_{22} and t_{23}. The first four tests measure the spatial ability of an individual, while the last three tests assess the logical ability of an individual. Hence, we choose to assign to the second factor the label *logical and spatial ability*;
3. the variables t_{10} and t_{12} load highly on the third factor, which is also determined by the variables t_{21} and t_{24}. They refer to psychological tests that assess cognitive capacities related to numerical processing, mathematical reasoning and arithmetic skills. We refer to the fourth factor as *numerical/mathematical ability*;
4. the variables from t_{14} to t_{19} determine the fourth common factor. The tests associated to these variables measure an individual's capacity of recognising numbers, words and figures and of making associations among them. Hence, the fourth factor can be interpreted as *recognition and association ability*;
5. the fifth factor is solely determined by the variable t_{13}. Hence we label the factor as its representative test, i.e. *straight-curved capitals*. It is immediate to observe that it is the only factor without an abstract meaning. This could be due to the fact that the proportion of variance explained by the factor is 0.026, which is too low to have a significative impact.

Finally it is remarkable that the variable t_{11} loads uniformly on the last three common factors, hence it is influenced by them similarly. This could be reasonable taking into account the psychological test associated with the variable (`code`).

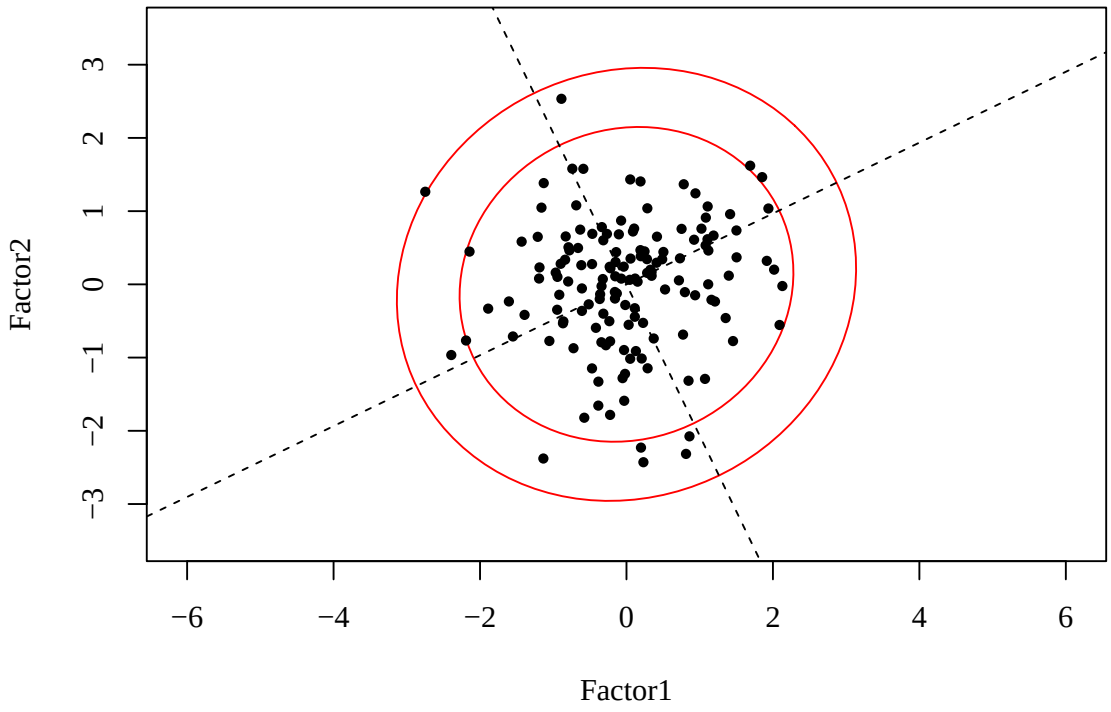
Before we move to the next step we want to underline that we decided not to fix a threshold value to assess significance of factor loadings. This choice is motivated by the fact that the total sample variance explained by the 5 factors is only 50.27%. Indeed this leads to the shortage of very high loadings and at the same time allows the presence of variables that have not much influence on any factor. Moreover by doing so we obtained a partition of our variables among the factors (with the only minor exception given by t_{11}).

1.3

We report below the scatterplot of the first two factor scores for the $m = 5$ solution obtained by the regression method, as requested.

```
faml_5_var_reg = factanal(gws, factors = 5, rotation = "varimax", scores = "regression")
score_5_var_reg = faml_5_var_reg$scores[, 1:2]
```

```
mu_5_var_reg = colMeans(score_5_var_reg[, 1:2])
sigma_5_var_reg = cov(score_5_var_reg[, 1:2])
eig_var_reg = eigen(sigma_5_var_reg, symmetric = T)
```



It seems there is no particular correlation between the two factors. In fact, if we compute it explicitly we obtain 0.074. Moreover, the covariance matrix turns out to be

$$\mathbf{\Sigma}_{\text{fs}} = \begin{pmatrix} 0.8673433 & 0.0607289 \\ 0.0607289 & 0.7712268 \end{pmatrix}.$$

As we can see from the scatterplot, the correlation is really close to 0. We should have excepted it since the factors scores are the estimated values of the common factors and in the theoretical model the covariance between any couple of common factors is 0, which implies that also their correlation is 0. In particular, the theoretical covariance matrix of the factors is equal to the identity matrix and our estimated covariance matrix is quite close to it: the estimated variance of the second factor is slightly smaller then what it should be, but is still acceptable taking into account that we are considering only 5 common factors which explain only the 50.27% of the total sample variance.

Finally, in order to analyse better the distribution of our data we decided to display the ellipsoids containing the 95% and the $\frac{n-0.5}{n}$% of the mass of a gaussian distribution with mean the sample mean and covariance matrix the sample covariance matrix (with n number of observations). We can see that the ellipsoids are in fact circles (nearly) which confirms that the first two common factors are jointly normally distributed.

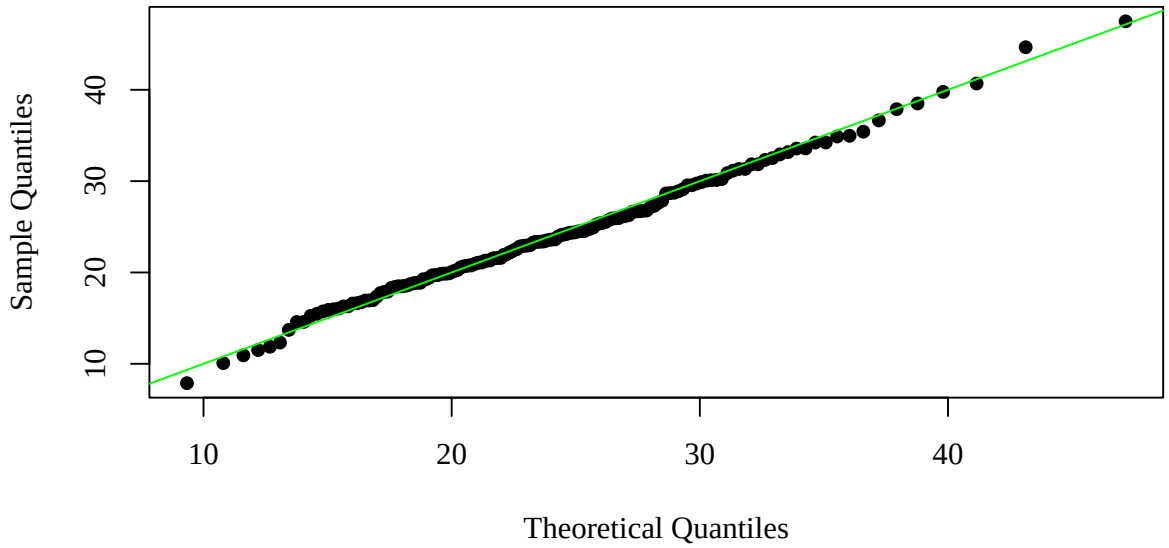
1.4

Let us now consider the `psych` dataset restricted to the Pasteur students.

```
pa = subset(psych_1, group == "pasteur", select = -group)
pas = scale(pa)
dim_pas = dim(pas)
```

Before obtaining the maximum likelihood solution (still with $m = 5$ factors), as we did for the Grant-White students data we first check if the normality assumption is satisfied.
Similarly to point 1 we look at the Q-Q plot of the squared Mahalanobis distances *vs* a χ^2_{24}.

```
par(family = "serif", mar = c(4, 4, 1, 1))
d = mahalanobis(pas, center = colMeans(pas), cov = cov(pas))
plot(qchisq(ppoints(d), df = ncol(pas)), sort(d), pch = 16,
    xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")
abline(0, 1, col = "green")
```



The Chi-squared Q-Q plot of the Mahalanobis distance shows that almos all the points lie on Q-Q line. Hence, we can say that the sum of the squares of our variables $(t_1, \ldots, t_{24})$ is χ^2_{24} distributed and so our variables can be considered jointly distributed as a multivariate gaussian.

As we did for the Grant-White students we double-check our result for the Pasteur students exploiting the Mardia test.

```
test_pas = mvn(data = pas, mvnTest = "mardia")
```

test	stat	p-value	result
mardia_skewness	2738.26661095053	0.0291360273355984	no
mardia_kurtosis	-0.43516106537097	0.663445521294024	yes
mvn	NA	NA	no

As before the data initially fails the test but by removing the most extreme outlier (i.e. the 43$^{\text{rd}}$ student) we obtain:

```
test_pas_out = mvn(data = pas[-out, ], mvnTest = "mardia")
```

test	stat	p-value	result
mardia_skewness	2695.60859510938	0.0935581297466659	yes
mardia_kurtosis	-0.836760265960978	0.402727336015076	yes
mvn	NA	NA	yes

We can again conclude that it is reasonable to assume that this portion of the `psych` dataset is also normal (in this case the number of outliers removed is even smaller: 1 out of 156).

We can now proceed with the computation of the maximum likelihood solution with `Varimax` rotation for $m = 5$.

```
faml_5_pas = factanal(pas, factors = 5, rotation = "varimax")
load_5_pas = faml_5_pas$loadings[, ]
```

	Factor1	Factor2	Factor3	Factor4	Factor5
t_1	0.3138	0.5777	0.1376	0.0960	0.0693
t_2	0.0369	0.5170	-0.0111	0.0582	-0.1445
t_3	0.0977	0.4437	-0.1772	0.0028	0.0996
t_4	-0.0161	0.6712	0.1896	0.1702	0.0115
t_5	0.8059	0.0428	-0.0668	0.0997	0.1429
t_6	0.7815	0.1574	0.0975	0.0566	0.2024
t_7	0.9043	0.0790	0.0051	0.1090	-0.0617
t_8	0.6838	0.1689	0.1388	0.1515	0.0140
t_9	0.7751	0.2488	0.0375	0.1015	0.1564
t_10	0.1408	-0.2079	0.1161	0.4998	0.6409
t_11	0.3490	0.0683	0.2313	0.6706	0.0649
t_12	0.0781	0.0967	-0.0064	0.5258	0.2168
t_13	0.0694	0.2719	0.0863	0.5436	0.0517
t_14	0.0418	0.0490	0.6900	-0.0261	0.0405
t_15	-0.1326	0.1253	0.6128	-0.1097	0.0973
t_16	0.0833	0.3864	0.4753	0.1758	0.1610
t_17	0.0675	-0.0544	0.5226	0.2891	0.0875
t_18	0.1004	-0.0052	0.4649	0.0891	-0.0076
t_19	0.0668	0.2442	0.3567	0.2405	0.0340
t_20	0.1231	0.5141	0.1888	0.0130	0.0963
t_21	0.2840	0.3871	0.1411	0.1951	0.4323
t_22	0.4685	0.4807	0.0289	0.1515	0.0513
t_23	0.3571	0.5871	0.1442	0.0916	0.2988
t_24	0.2180	0.2944	0.2265	0.2397	0.5296

In the analysis of the factor loadings we adopt the same strategy as before: we look at the matrix by rows and we do not set any threshold value. We obtain a perfect partition of the variables among the 5 common factors, in particular:

1. the first factor is determined by the same 5 variables as before, namely t_5, t_6, t_7, t_8 and t_9. Therefore, it can be interpreted in the exact same way, which is *verbal ability*;
2. as in the previous point, the second factor is influenced by the same 7 variables as before, that are the ones from t_1 to t_4 togheter with t_{20}, t_{22} and t_{23}. Hence, we can assign to the second factor the same label: *logical and spatial ability*;
3. the third factor is determined by the variables from t_{14} to t_{19}. These same variables previously formed the fourth common factor, hence there was just an exchange of order between the factors. We interpret it as *recognition and association ability*;
4. the variables from t_{11} to t_{13} form the fourth common factor. The tests associated to these variables are respectively *code*, *counting dots* and *straight-curved capitals* which are related to quick visualization skills. We label it *quick visualization/speed ability*;
5. finally the last factor is influenced by the variables t_{10}, t_{21} and t_{24} which previously formed the third common factor together with the variable t_{12}. Despite the absence of t_{12} the factor has not lost its

meaning, therefore we interpret it as *numerical/mathematical ability*.

A necessary remark is that the new factors can be viewed as a permutation of the ones we have obtained for the Grant-White students data, but we need to specify that the variable t_{12} moved from the third to the fifth factor (without following the permutation) and that, unlike before, we now menage to give an abstract meaning to all the common factors.

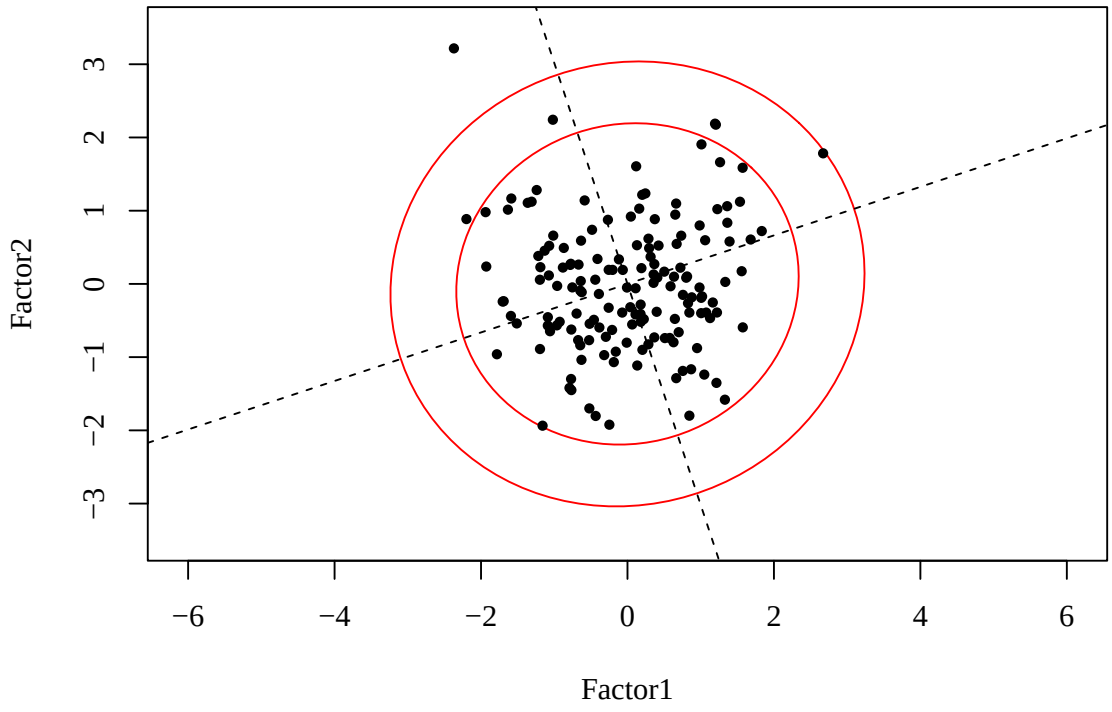
For the sake of completeness, we report the permutation of the factors in compact form:

$$\sigma \in S_5, \text{ such that } \sigma(4) = 3, \sigma(3) = 5, \sigma(5) = 4.$$

1.5

We have already made the scatterplot of the first two factor scores from the rotated MLFA solution for the Grant-School in the point 1.3. We now follow the exact same procedure for the Pasteur school and than we make a comparison between the results.

```
faml_5_pas_reg = factanal(pas, factors = 5, rotation = "varimax", scores = "regression")
score_5_pas_reg = faml_5_pas_reg$scores[, 1:2]
mu_5_pas_reg = colMeans(score_5_pas_reg[, 1:2])
sigma_5_pas_reg = cov(score_5_pas_reg[, 1:2])
eig_pas_reg = eigen(sigma_5_pas_reg, symmetric = T)
```



As for the Grant-White students data it seems there is no particular correlation between the two factors. The correlation between the factors is 0.047 and the covariance matrix is

$$\mathbf{\Sigma}_{\text{fs}} = \begin{pmatrix} 0.9125682 & 0.040457 \\ 0.040457 & 0.8038656 \end{pmatrix}.$$

Again the factors appears to be jointly normally distributed: almost every point (except for 1) falls inside the ellipsoid which is supposed to contain $\frac{n-0.5}{n}\%$ of the mass under a multivariate normal with the sample covariance matrix as covariance matrix and centre in the sample mean.

Note also that this covariance matrix is closer to the identity matrix then the previous one, however the gap between the variances of the first two factors still holds.

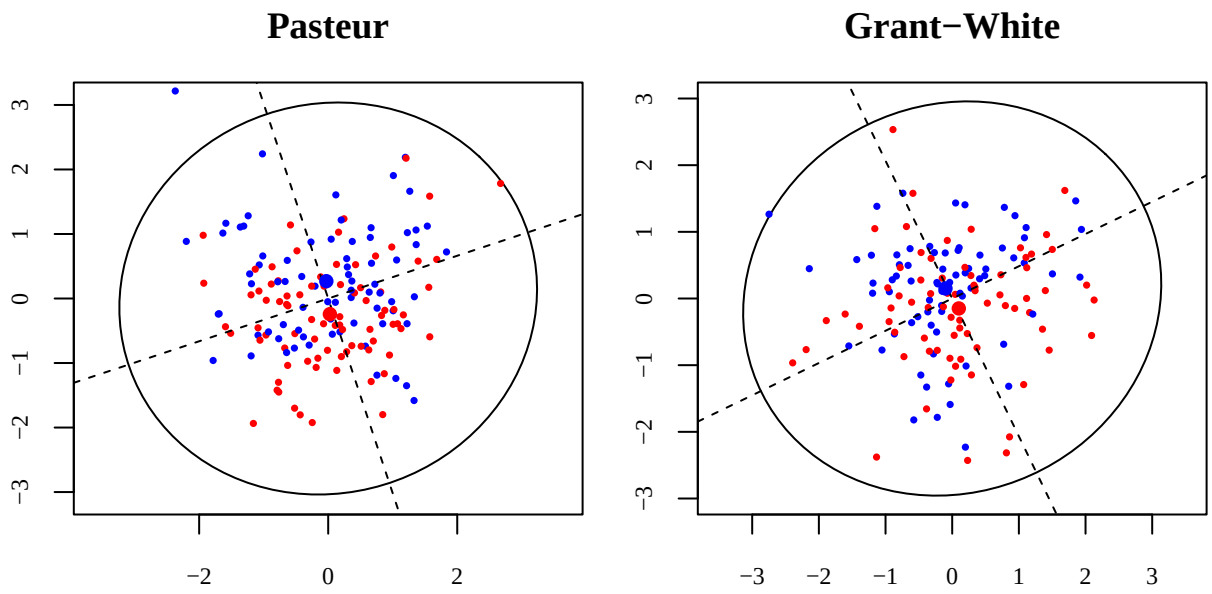
We now analyse the same scatterplots by grouping the data according to some of the initial variables of the dataset `psych` that we have not used in the factor analysis because they did not represent any psychological test, namely *sex* and *age*. Our aim is to see if we can extract any significant relationship between the groups and the results of the tests.

We first group the students according to the variable sex.

```
sex_pa = psych_0[1:156, 2]
col_pa = rep("blue", length(sex_pa))
col_pa[sex_pa == "f"] = "red"
mu_pa_sexm = colMeans(score_5_pas_reg[sex_pa == "m", 1:2])
mu_pa_sexf = colMeans(score_5_pas_reg[sex_pa == "f", 1:2])
```

```
sex_gw = psych_0[157:301, 2]
col_gw = rep("blue", length(sex_gw))
col_gw[sex_gw == "f"] = "red"
mu_gw_sexm = colMeans(score_5_var_reg[sex_gw == "m", 1:2])
mu_gw_sexf = colMeans(score_5_var_reg[sex_gw == "f", 1:2])
```

In the following plots the blue points (■) refer to male students while the red ones (■) to the female students. We also plot the respective mean points of the two groups with bigger dots (with the same colors). The ellipses cover the $\frac{n-0.5}{n}\%$ of the mass of a gaussian distribution with means the sample means and covariance matrix the sample covariance matrix.

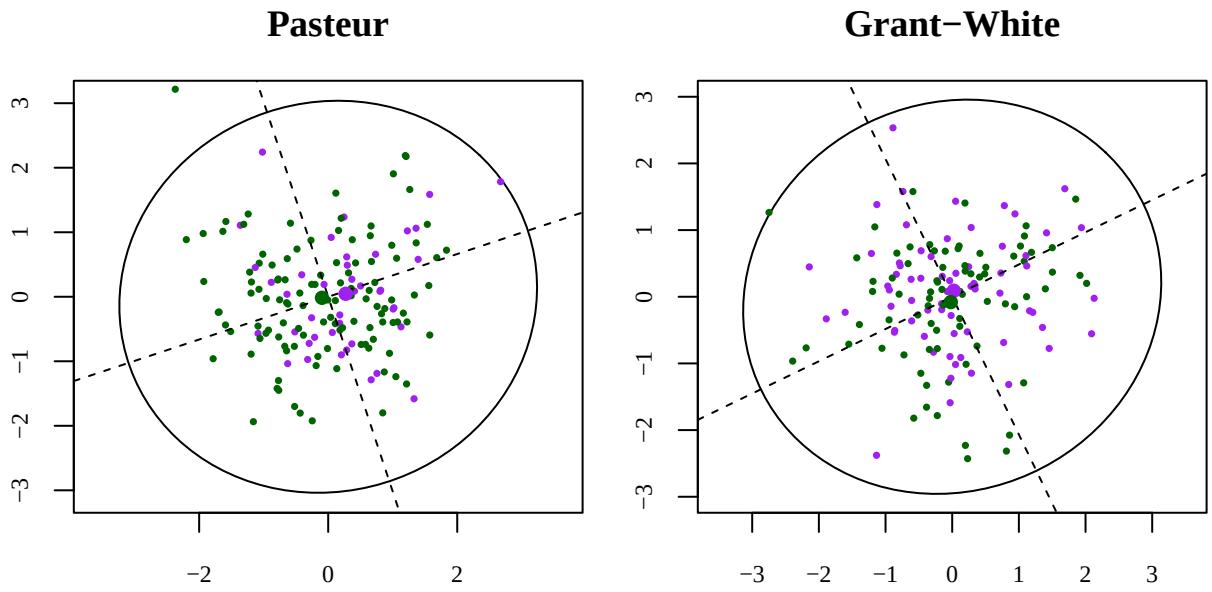


According to the scatterplots female and male students among both schools appears to share almost the same verbal skills, but in both cases male students seems to score higher in the second factor (*logical/spatial ability*).

Then we group the students by age, separing the younger ones (■) (with age < 13) from the olders (■).

```
age_pa = psych_0[1:156, 3]
col_pa = rep("black", length(age_pa))
col_pa[age_pa < 13] = "purple"
col_pa[age_pa >= 13] = "darkgreen"
```

```
mu_pa_age1 = colMeans(score_5_pas_reg[age_pa < 13, 1:2])
mu_pa_age2 = colMeans(score_5_pas_reg[age_pa >= 13, 1:2])

age_gw = psych_0[157:301, 3]
col_gw = rep("black", length(age_gw))
col_gw[age_gw < 13] = "purple"
col_gw[age_gw >= 13] = "darkgreen"
mu_gw_age1 = colMeans(score_5_var_reg[age_gw < 13, 1:2])
mu_gw_age2 = colMeans(score_5_var_reg[age_gw >= 13, 1:2])
```



For the Grant-White students we basically cannot see any particular difference between the distributions of the two groups: the mean points are very close. As for the Pasteur students we get that they score similarly on the second factor (*logical/spatial ability*) while there is a little difference in the first common factor in favour of the younger students. Since we do not know any additional information about the students and about the exact structure on the psychological tests we are not able to say whether it does make sense or not.

Exercise 2

Consider the dataset `pendigits` containining $n = 10992$ observations with 16 numerical variables and 1 categorical variable which is the class attribute (`digit` $\in \{0, \dots, 9\}$).

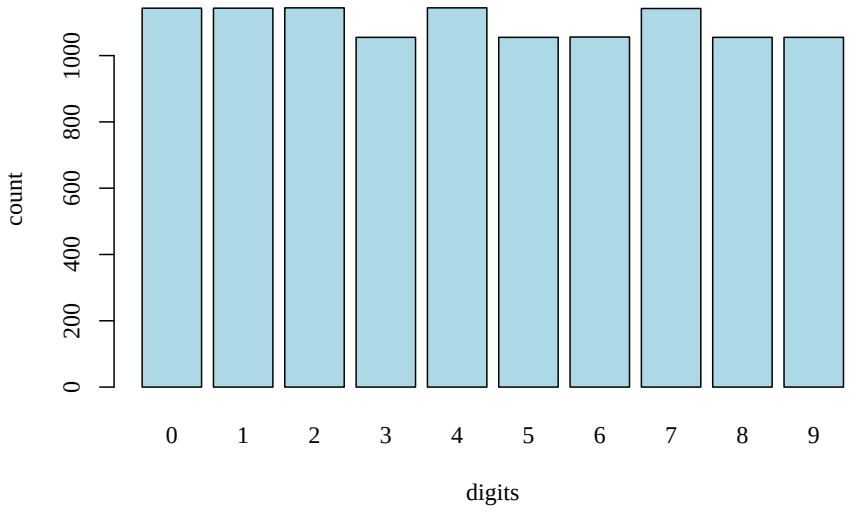
```
pendigits = read.table("data/pendigits.txt", sep = ",", head = F)
names(pendigits) = c(paste0(rep(c("x", "y"), 8), rep(1:8, each = 2)), "digit")
lookup = c("darkgreen",  "brown", "lightblue",  "magenta", "purple",
        "blue", "red", "lightgreen", "orange", "cyan")
names(lookup) = as.character(0:9)
digit_col = lookup[as.character(pendigits$digit)]
head(pendigits)
```

x1	y1	x2	y2	x3	y3	x4	y4	x5	y5	x6	y6	x7	y7	x8	y8	digit
47	100	27	81	57	37	26	0	0	23	56	53	100	90	40	98	8
0	89	27	100	42	75	29	45	15	15	37	0	69	2	100	6	2
0	57	31	68	72	90	100	100	76	75	50	51	28	25	16	0	1
0	100	7	92	5	68	19	45	86	34	100	45	74	23	67	0	4
0	67	49	83	100	100	81	80	60	60	40	40	33	20	47	0	1
100	100	88	99	49	74	17	47	0	16	37	0	73	16	20	20	6

It could be useful to have an idea of the class sizes hence we display them below.

```
count_dig = as.vector(table(pendigits$digit))
prop_dig = as.vector(table(pendigits$digit)) / sum(as.vector(table(pendigits$digit)))
```

```
plt_count_dig = t(as.matrix(count_dig))
colnames(plt_count_dig) = 0:9
bp = barplot(plt_count_dig,
    xlab = "digits", ylab = "count",
    col = "lightblue")
```

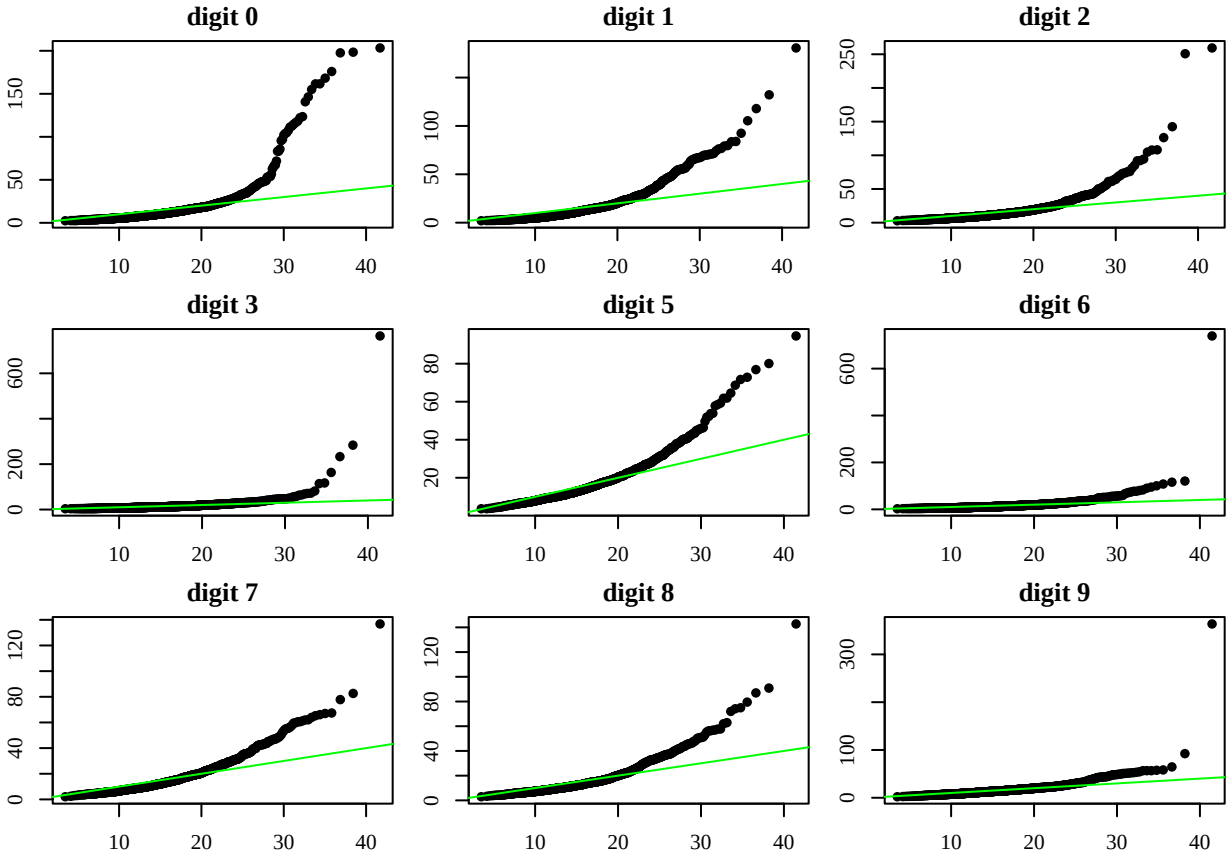


We can see that the distributions are pretty homogeneous: the classes corresponding to the digits 3, 5, 6, 8 and 9 are just slightly less numerous then the others.

2.1

The Linear Discriminant Analysis (LDA) technique relies on the assumption that each different class is multivariate gaussian distributed with different means μ_i (centroids) and with the same covariance matrix $\mathbf{\Sigma}$. Hence, we should first check if this assumption holds.

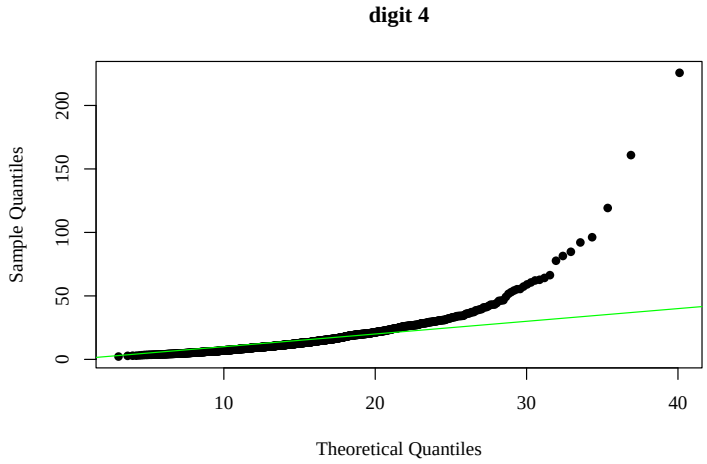
We look at the Q-Q plot of the squared Mahalanobis distances *vs* a χ^2_{16} for each different class.



Let us first note that we removed the digit 4 from our analysis. This is because the last variable of the dataset restricted to this class is entirely made of zeros. Hence, in this case the Mahalanobis distance is not well-defined since the covariance matrix is singular. This problem can be overcame by recalling that by definition a multivariate gaussian vector $\boldsymbol{X}$ should satisfy the following property:

$$a^T \boldsymbol{X} + b \sim \mathcal{N}\left(\mu, \sigma^2\right), \forall a, b \in \mathbb{R}^{\texttt{size(}\boldsymbol{X}\texttt{)}}.$$

This definition still holds if $\boldsymbol{X} = (\boldsymbol{Y}, 0)$ with $\boldsymbol{Y}$ multivariate gaussian, indeed the idea is that a constant random variable c can be considered a $\mathcal{N}(c, 0)$. Thus, for the digit 4, we test the normality assumption by just removing the last variable (`y_8`).



By looking at the Q-Q plots, it is clear that no class is normally distributed: all the digits, except for 6 and 9, seems to have heavy right tails. Actually, the plots of the remaining two digits appear to be closer to the Q-Q line but this is only because in both cases there is an extreme outlier which distorts the figure.

It is also possible to check whether the covariance matrices of the different classes are similar or not. In order to do so, we display below the matrix M such that

$$M_{i,j} = \left\| \text{Var}\left(\texttt{pendigits}\big|_{\texttt{digit}=i}\right) - \text{Var}\left(\texttt{pendigits}\big|_{\texttt{digit}=j}\right) \right\|_{\text{F}},$$

where $\|\cdot\|_{\text{F}}$ is the Frobenius norm of a matrix while $\text{Var}(\cdot)$ denotes the covariance matrix of a random vector.

```
sub_digits = list()
for (i in 1:10){
    sub_digits[[i]] = pendigits[pendigits[, 17] == (i - 1), 1:16]
}
sub_digits_cov = list()
for (i in 1:10){
    sub_digits_cov[[i]] = cov(sub_digits[[i]])
}
mat = matrix(rep(1, 100), ncol = 10)
for (i in 1:10){
    for (j in 1:10){
        mat[i, j] = norm(sub_digits_cov[[i]] - sub_digits_cov[[j]], "f")
    }
}
```

$$\begin{pmatrix}
0 & 5142 & 4321 & 4529 & 3537 & 12931 & 4174 & 4775 & 6987 & 4941 \\
5142 & 0 & 4301 & 4525 & 4143 & 12660 & 4479 & 3541 & 6711 & 4941 \\
4321 & 4301 & 0 & 1167 & 2026 & 12320 & 1815 & 2539 & 6235 & 3236 \\
4529 & 4525 & 1167 & 0 & 2155 & 12348 & 1597 & 2880 & 6219 & 3003 \\
3537 & 4143 & 2026 & 2155 & 0 & 12315 & 2366 & 3000 & 6429 & 3088 \\
12931 & 12660 & 12320 & 12348 & 12315 & 0 & 12346 & 11526 & 12410 & 11659 \\
4174 & 4479 & 1815 & 1597 & 2366 & 12346 & 0 & 2985 & 6211 & 3344 \\
4775 & 3541 & 2539 & 2880 & 3000 & 11526 & 2985 & 0 & 6185 & 3786 \\
6987 & 6711 & 6235 & 6219 & 6429 & 12410 & 6211 & 6185 & 0 & 5991 \\
4941 & 5023 & 3236 & 3003 & 3088 & 11659 & 3344 & 3786 & 5991 & 0
\end{pmatrix}$$

As we can see, the covariance matrices seem pretty far from being similar: the entries of the matrix M are

very large, in particular the ones corresponding to the sixth row.

The fact that our classes do not appear to be normally distributed and not to have similar covariance matrices does not compromise the applicability of the LDA, since these assumptions are only required for an optimal solution.

We now apply the LDA procedure (already implemented in `R`). It identitifies recursively 9 discriminant variables from the 10 classes.

The linear discriminant classifier can be obtained from two alternative derivation. Both consider each group as sample from distributions with the same variance - thus we speak of *pooled variance*. The first derivation further assumes that, for all groups, data are distributed as multivariate gaussian, with different means (the *centroids*) and pooled covariance matrix. Under this assumption, the LDA classifier is the bayes classifier and thus is optimal. If we now relax the assumption of gaussianity, the centroids and the pooled covariance matrix $\mathbf{\Sigma}_X$ do not fully identity the distribution within each group anymore. However, we can still recover the LDA classifier as solution to a different problem. Consider the matrix

$$\boldsymbol{B}_X \coloneqq \frac{1}{K}\sum_{j=1}^{K}(\mu_{X,j}-\mu_X)(\mu_{X,j}-\mu_X)^T,$$

which we call between-class sum of cross product, or simply between class variance, as it can also be seen as the sample variance of the vector of centroids $(\mu_{x,j})_{j=1}^{K}$. We want to determine a linear combination of the original p variables, such that observations coming from different groups are maximally distanced. Consider thus LD1 $\coloneqq$ aX. Along this direction each group has respectively mean $\mu_{\text{LD1,j}} = a\mu_{X,j}$ and variance $\mathbf{\Sigma}_{\text{LD1}} = a^T\mathbf{\Sigma}_X a$. We therefore look for the direction a that maximes the between class variance holding the within class variance constant:

$$a \coloneqq \text{argmax}\left(\frac{\text{Var}\left((\mu_{\text{LD1,j}})_{j=1}^{K}\right)}{\text{Var}\left(\text{LD1}\right)}\right) = \text{argmax}\left(\frac{\boldsymbol{B}_{LD1}}{\mathbf{\Sigma}_{\text{LD1}}}\right) = \text{argmax}\left(\frac{a^T\boldsymbol{B}_X a}{a^T\mathbf{\Sigma}_X a}\right).$$

We can simplify the problem by first sphering the data. It becomes equivalent to determining the direction that maximizes the operator norm of the the matrix $\mathbf{\Sigma}_X^{-\frac{1}{2}}\boldsymbol{B}_X\mathbf{\Sigma}_X^{-\frac{1}{2}}$, i.e. identifying the unit eigenvector associated to its largest eigenvalue. We call the variable LD1 first discriminant variable. Suppose now to iterate this problem looking for LD2 in the subspace of the span of the p variables orthogonal to LD1. The solution will be the unit eigenvector associated with the second largest eigenvalues (counted with multiplicity). If we iterate this procedure, restricting at each step to the ortoghonal to the span of the previous linear discriminant variables, we can get $K-1$ ortoghonal discriminant variables.

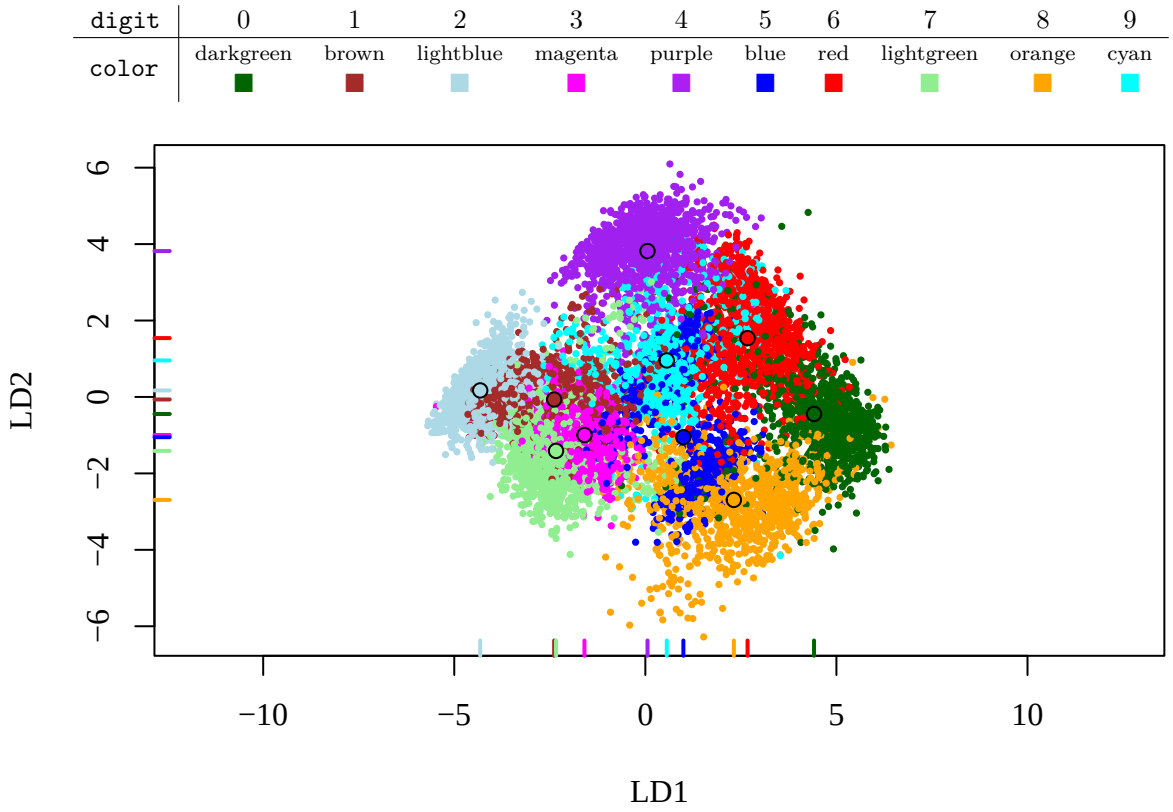
```
lda_fit = lda(digit ~ ., data = pendigits)
```

	LD1	LD2	LD3	LD4	LD5	LD6	LD7	LD8	LD9
x1	0.0170	0.0108	0.0011	-0.0302	-0.0096	-0.0052	0.0110	-0.0102	-0.0008
y1	0.0105	0.0302	-0.0186	0.0291	-0.0084	-0.0431	-0.0508	-0.0262	0.0195
x2	0.0062	-0.0024	-0.0024	0.0115	-0.0164	0.0098	-0.0270	-0.0052	0.0114
y2	-0.0341	-0.0386	-0.0114	0.0127	0.0423	-0.0100	0.0372	-0.0501	-0.0700
x3	-0.0243	-0.0223	-0.0123	-0.0172	0.0257	-0.0191	0.0180	-0.0098	0.0010
y3	0.0038	0.0047	-0.0184	-0.0137	-0.0351	-0.0186	-0.0334	0.0452	0.0951
x4	-0.0068	0.0055	0.0056	0.0070	-0.0137	0.0133	-0.0118	0.0015	0.0051
y4	0.0154	-0.0123	0.0102	-0.0004	0.0072	0.0509	-0.0443	-0.0629	-0.0584
x5	0.0029	-0.0088	-0.0244	0.0112	0.0330	0.0155	0.0155	-0.0166	0.0249
y5	-0.0118	-0.0095	-0.0311	-0.0133	-0.0214	-0.0289	0.0292	0.0250	-0.0283
x6	0.0115	0.0190	-0.0064	-0.0102	0.0073	-0.0204	-0.0271	0.0152	-0.0312
y6	-0.0016	0.0159	-0.0625	-0.0082	-0.0201	-0.0309	0.0093	-0.0062	0.0384
x7	-0.0021	-0.0062	0.0074	-0.0002	0.0045	0.0283	0.0366	-0.0359	0.0439

	LD1	LD2	LD3	LD4	LD5	LD6	LD7	LD8	LD9
y7	0.0235	-0.0081	0.0571	0.0383	0.0125	0.0629	-0.0496	-0.0171	-0.0183
x8	-0.0207	0.0178	-0.0157	0.0087	-0.0139	-0.0100	-0.0084	-0.0047	-0.0188
y8	0.0322	-0.0526	-0.0524	-0.0209	-0.0142	-0.0335	0.0125	-0.0040	-0.0073

Consider now the first two discriminant directions, that is the first two columns of the scaling matrix (LD1 and LD2). We display below the scatterplot of the data restricted to these two components, color coding the observations according to variable `digit_col` and adding the centroids for each class. In particular, as the discriminant variables are found iteratively as the direction of highest discrimination in smaller and smaller subspaces, they are ordered by their *power of discrimination*; hence, a plot of the first two discriminant variables should be rather informative of the degree by which classes can be discriminated.

```
pred = predict(lda_fit)
centroids = aggregate(pred$x, by = list(pendigits$digit), FUN = mean)
centroids = centroids[, -1]
covariances = list()
for (i in 1:10){
    covariances[[i]] = cov(pred$x[pendigits[, 17] == (i - 1), ])
}
```



Before commenting the scatterplot, we would like to say that we have also tried to compute the centroids of our 9 classes in a more theoretical way. First we have collected in a 16×10 matrix the means of our classes, then we have computed a *grand mean* by multiplying this matrix by the vector containing the proportion of observations for each class (`prop_dig`). Finally, we have "centered" the mean vector of each class by subtracting the grand mean and we have applied the linear trasformation given by the `scaling`, which is

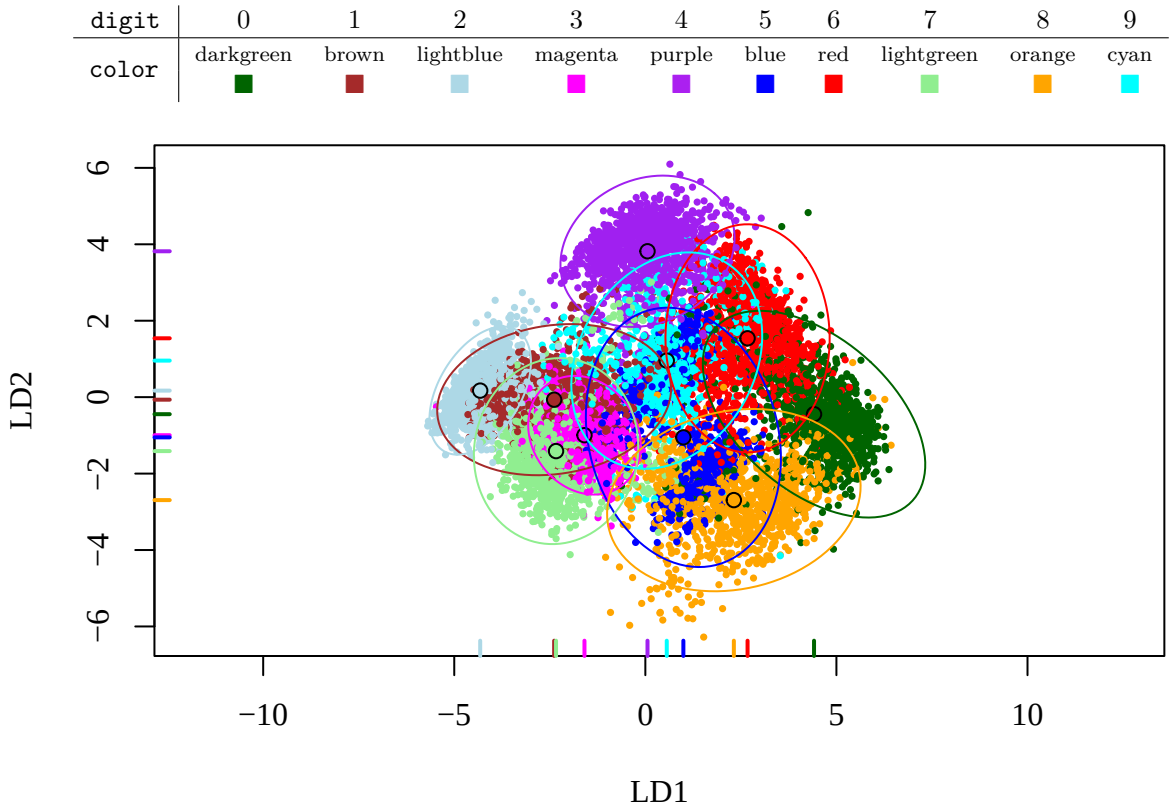
given as output by the `lda` procedure.

```
all = matrix(rep(0, 160), ncol = 10)
for (i in 1:10){
    all[, i] = as.matrix(colMeans(sub_digits[[i]]))
}
grand_mean = all %*% as.matrix(prop_dig)
centroids_other = matrix(rep(0, 20), ncol = 2)
for (i in 1:10){
    centroids_other[i, ] = t(t(lda_fit$scaling[, 1:2]) %*%
        (as.matrix(colMeans(sub_digits[[i]])) - grand_mean))
}

norm_diff = norm(centroids_other - as.matrix(centroids[, 1:2]), "i")
```

By doing so, we get the 10×2 matrix `centroids_other`, which is almost identical to the first two columns of the previously computed matrix `centroids`. Indeed, the uniform norm of the difference of the two distinct computation is $2.4910629 \times 10^{-15}$.

We can now comment the scatterplot. By looking at it, we can say that although the centroids are separated from one another (more or less distant), there is a considerable overlap between the classes. In particular, the ones corresponding to the digits 3 (■), 5 (■), 1 (■) and 9 (■) seem to be the most difficult to discriminate as they mix with almost all the others.

In order to have a better understanding of how much the classes overlap we add to the previous plot the ellipses covering the 95% of the mass of a gaussian distribution with means the centroids and covariance matrix the sample covariance matrix, for each of the 10 classes.



The plot confirms what we observed before. In particular:

- the ellipses corresponding to the digits 5 (■) and 9 (■) intersect all the others except for the one corresponding to the digit 2 (■), which is located on the extreme left of the cloud. Hence, we expect this two classes to be largely misclassified;
- the 4^{th} (■) and the 2^{nd} (■) classes are the only two whose ellipses do not contain any other centroid, although they intersect the ellipses of other classes (respectively three and two classes). However, unlike the purple one (■), the lightblue one (■) is almost entirely contained in the brown one (■). Therefore, we expect the digit 2 to be confused with the digit 1 while the digit 4 to be one of the best classified;
- the classes corresponding to magenta (■, i.e. 3), brown (■, i.e. 1), lightgreen (■, i.e. 7), blue (■, i.e. 5) and cyan (■, i.e. 9) have ellipses which contain at least three different centroids (the brown one even contains four different centroids and touches a fifth one). In particular, as it happens for the lightblue (■) ellipse, the magenta one (■) is completely contained in the lightgreen one (■), thus we expect the corresponding digits to be often confused;
- the ellipses corresponding to the digits 6 (■), 0 (■) and 8 (■), which are located on the extrems of the cloud, dispite being large, contain "only" two different centroids (actually the red one only touches the cyan one).

In conclusion, we can say that we expect there to be various missclassifications, particularly between those classes which overlap the most. On the contrary, we expect not to have any missclassification between the most distant classes, such as (4■, 8■), (2■, 0■) and (6■, 7■).

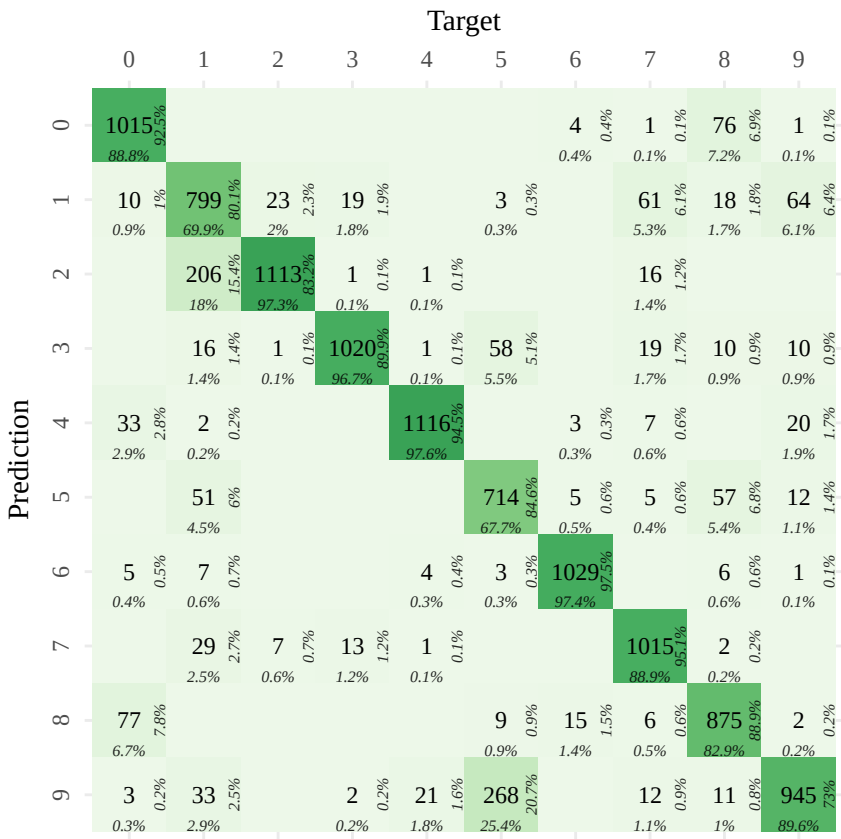
2.2

We now compute and display the confusion matrix on the training data, as requested. We use the `confusion_matrix()` function, which displays togheter with the number of correct/incorrect classifications the following percentages:

- the percentage of j's classified as i's is located at the bottom of the square corresponding to the entrance (i, j), with $i, j \in \{0, \dots, 9\}$;
- the percentage of predicted i's with target j is located at the right of the square corresponding to the entrance (i, j) (here with the word target we indicate the true belonging class).

In particular, the blank squares correspond to zeros.

```
conf_mat = confusion_matrix(targets = pendigits$digit, predictions = pred$class)
```



We can also compute the training error rate (we can extract it from `conf_mat` or compute it directly):

```
error_rate = 1 - conf_mat$"Overall Accuracy"
# error_rate = 1 - mean(pendigits$digit == pred$class)
```

train_error_rate	0.1229076

The training error rate is unexpectedly low: only the 12.29% of the observations are misclassified. In particular:

- it is conspicuous that the worst performances are indeed related to the classes corresponding to the digits 1 and 5: unlike the other classes, whose percentage of correct classifications exceeds the 80%, in these cases we respectively obtain 69.9% and 67.7%;
- moreover, we can observe that the percentage of obervations misclassified as 9's is 27% and the majority of it comes from the 5's (20.7%). This is consistent with what we previously observed in the scatterplot, since the cyan ellipse (■, i.e. 9) is located in the middle of the cloud and particularly overlaps with the blue one (■, i.e. 5);
- similarly to what happens with the classes 9 and 5 a considerable amount of 1's is misclassified as 2, which is coherent with the fact that the lightblue ellipse (■, i.e. 2) is almost entirely contained in the brown one (■, i.e. 1);
- surprisingly the digit 3 (associated to the color magenta ■) does not have as many misclassifications as we thought. In particular, we excepted an higher number of misclassifications between the digits 3 and 7 since the cloud corresponding the first one is completely contained in the ellipse related to the second one;

- there are no misclassifications among the classes which are the most distant in the LD1∼LD2 space, as we thought.

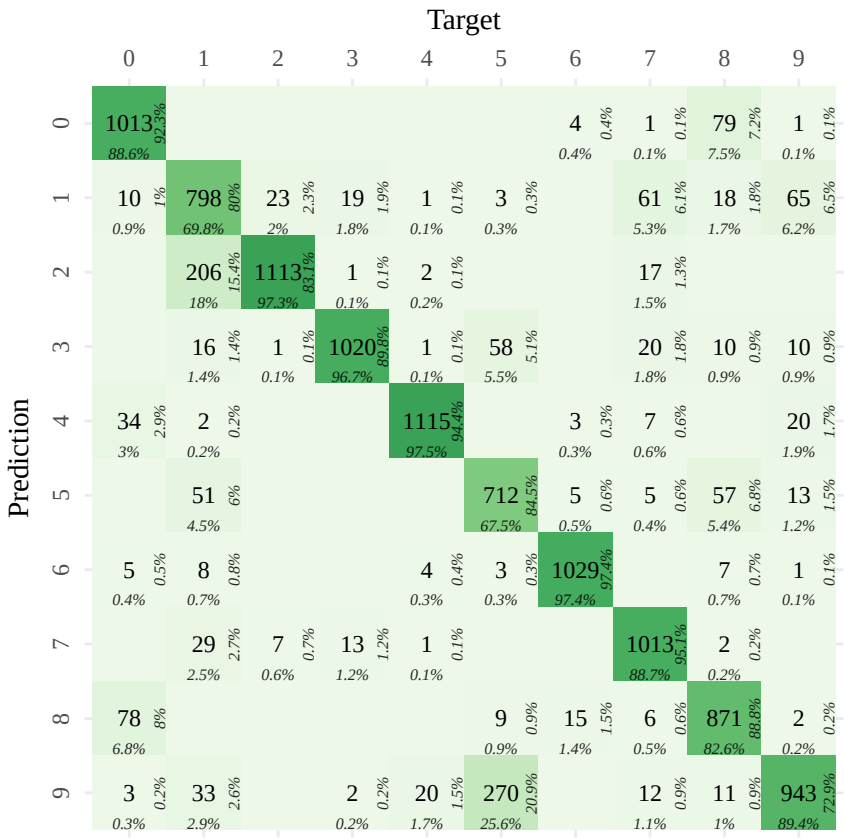
2.3

The *training error rate* computed in the previous point is a good indicator of how well the model can describe the data. However, this is potentially of little use in making predictions. It is more interesting to assess the performance of the model on novel observations. One way to do so is to compute the *leave-one-out cross-validation error rate* (*LOOCV error rate*), which can be interpreted as an estimator of the misclassification error for future observations. With this procedure, as the name suggests, the model is trained leaving out one obervation at a time, and using it for testing.

```
lda_cv = lda(digit ~ ., data = pendigits, CV = T)
```

We obtain the following confusion matrix

```
conf_mat_cv = confusion_matrix(targets = pendigits$digit, predictions = lda_cv$class)
```



and the following *LOOCV error rate*

```
loocv_error = 1 - conf_mat_cv$"Overall Accuracy"
```

loocv_error	0.1241812

The confusion matrix is almost identical to the previous one, which is reasonable because actually we are just "leaving out" one observation at a time.

Moreover, the error rate given by this procedure is slightly larger then the one we computed in the previous point. We might have expected it since for each *test* we are using all the dataset as training set except for one observation. In particular, we can compute the *relative error* between the two error rates in order to compare them. We obtain that our new error increases of roughly 1.036% with respect to the previous one.

2.4

In reduced-rank linear discriminant analysis, it is possible to perform the nearest centroid classification in a subspace of dimension $L < \min\{K-1, p\}$ by using for the prediction only the first L discriminant directions. When $L = \min\{K-1, p\}$ we call it full-rank linear discriminant analysis. In order to perform reduced-rank LDA, it is sufficient to specify the dimension of the space to be used for the prediction when calling the `predict()` function.

We are asked to implement the 44-*fold cross validation* for each reduced-rank LDA, including the full-rank LDA, by using the partition of the observations provided by the variable `group_cv` in order to estimate the error rate.

```
group_cv = rep(1:44, each = 250)
group_cv = group_cv[seq_along(pendigits$digit)]
```

The 44-*fold cross validation* fits the model on 43 out of 44 subsets (the training set) and uses the remaining subset as test set. This procedure is then repeated 44 times so that each subset will be used as test set for the model. The function below implements the process and returns the corresponding *CV error rate* and what we called `test_error`, that is the arithmetic mean of the test errors among the 44 different folds.

```
cv_44_fold = function(data, grouping, rank) {
    test_error = 0
    misclassified = 0
    for (group in 1:44){
        train_set = data[-which(grouping == group), ]
        test_set = data[which(grouping == group), ]
        model = lda(digit ~ ., data = train_set)
        pred = predict(model, test_set, dimen = rank)
        confusion_matrix = table(pred$class, test_set$digit)
        test_error = test_error + 1 - (sum(diag(confusion_matrix)) / dim(test_set)[1])
        misclassified = misclassified + dim(test_set)[1] - sum(diag(confusion_matrix))
    }
    cv_error = misclassified / dim(data)[1]
    test_error = test_error / 44
    return(c(test_error, cv_error))
}
```

We can now call the function and print the results. Note that we decided to compute also the training error rates in order to make a comparison.

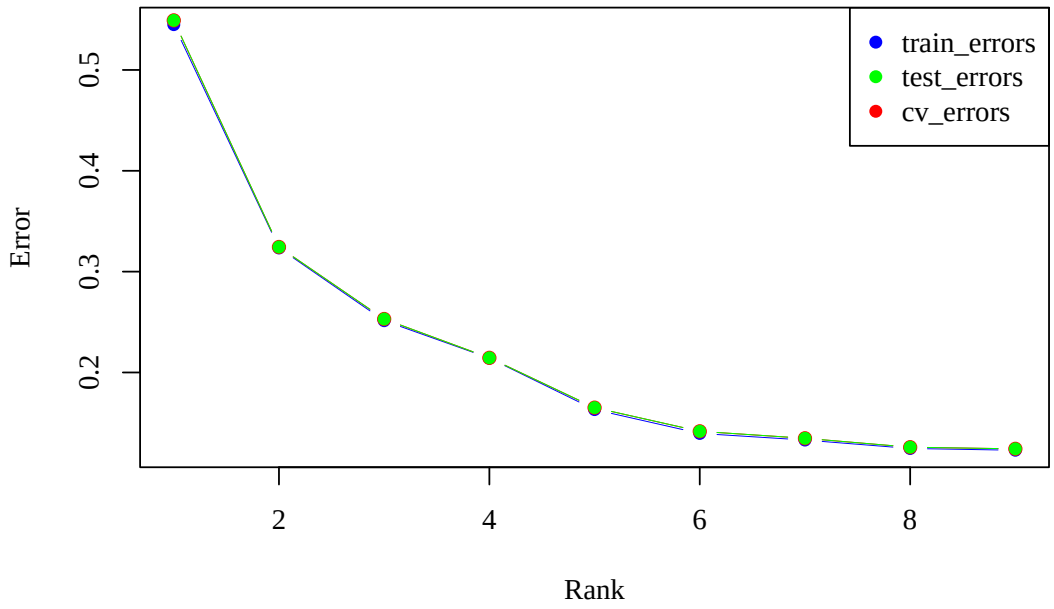
```
train_errors = 0
test_errors = 0
cv_errors = 0
for (rank in 1:9){
    pred_train = predict(lda_fit, dimen = rank)
    confusion_matrix_train = table(pred_train$class, pendigits$digit)
    train_errors[rank] = 1 - sum(diag(confusion_matrix_train) / dim(pendigits)[1])

    error = cv_44_fold(pendigits, group_cv, rank)
    test_errors[rank] = error[1]
```

```
    cv_errors[rank] = error[2]
}
```

	rank_1	rank_2	rank_3	rank_4	rank_5	rank_6	rank_7	rank_8	rank_9
train_errors	0.54494	0.32351	0.25136	0.21425	0.16321	0.13956	0.13301	0.12464	0.12291
test_errors	0.54913	0.32443	0.25301	0.21453	0.16514	0.14165	0.13474	0.12591	0.12428
cv_errors	0.54913	0.32442	0.25300	0.21452	0.16512	0.14165	0.13473	0.12591	0.12427

Note that the training error rate we get for the full-rank LDA is the same as the one we obtained in point 2.2. Moreover, the difference between the test errors and the CV errors is negligible (the order of magnitude is 10^{-5}).

We can now plot the error curves against the number of discriminant variables.



Surprisingly the *training error rate* is almost identical to the other two error rates: the first one is always lower than the others, in accordance with the theory. Note that the discrepancy between the *training error rate* and the *CV error rate* is at most 0.0042 (of course we obtain a similar value with the *test error rate*). Note also that in each case as the number L of discriminant directions increases, the corresponding error value decreases. Therefore, the full-rank LDA classifier should provide the best classifications, that is, when considering $L = 9$ discriminant directions, the projected centroids achieve the maximum spread relative to the within-class variance.

If we were interested in having a dimensionality reduction we could decide to keep only $L = 6$ dimensions, since the difference between dimension 6 and 9 is approximately 0.0174 in both *training error rate* and *CV error rate*. This leads to a reasonable dimensionality reduction.

Clearly, due to the high dimensionality it is not possible to have a graphical representation of observations and centroids. To this aim, $L = 2$ discriminant directions should be used, as previously done.

2.5 (optional)

We now use several classification methods in order to try to improve the estimate of the CV error rate found in the previous point with LDA.

We implemented the following procedures by using the libraries `MASS`, `klaR` and `randomForest`:

1. *KNN - K-nearest neighbours* for different values of the parameter K;
2. *Noised QDA* - since the actual *Quadratic Discriminant Analysis* cannot be performed (there is a rank deficiency in the group corresponding to the digit 4), we decided to add some noise to the values of the variable `y_8` of the observations belonging to the class 4. For the sake of completeness, we report below also the unsuccessful attempt of the QDA computation together with the `R` error message;
3. *RDA - Regularized Discriminant Analysis*, i.e. a sort of a trade-off between LDA and QDA. We choose as parameters $\gamma = 0$ and $\lambda = 0.2$;
4. *RF - Random Forest*, i.e. a classification method that combines multiple decision trees to make predictions by aggregating the results. It exploits random sampling of data and features to create different trees and improves accuracy while mitigating overfitting.

The function `cv_44_fold_mod()` reported below computes both the *44-fold CV error rate* and the *test error rate* (recall that it is the arithmetic mean of the test errors among the 44 different folds; we previously called it `test_error`). The function takes as input the following variables:

- `data` - the dataset of interest;
- `grouping` - the grouping variable, in our case `group_cv`;
- `mod` - it is a string that indicates the classification method we want to perform;
- `k` - additional variable containing the parameters optionally requested by some of the models.

```
cv_44_fold_mod = function(data, grouping, mod, k) {
    test_error = 0
    misclassified = 0
    for (group in 1:44){
        train_set = data[-which(grouping == group), ]
        test_set = data[which(grouping == group), ]
        if (mod == "lda") {
            model = lda(digit ~ ., data = train_set)
            pred = predict(model, test_set)
            confusion_matrix = table(pred$class, test_set$digit)
        }
        if (mod == "knn") {
            pred = knn(train_set[, -17], test_set[, -17], train_set[, 17], 2 * k - 1)
            confusion_matrix = table(pred, test_set$digit)
        }
        if (mod == "qda") {
            model = qda(digit ~ ., data = train_set)
            pred = predict(model, test_set)
            confusion_matrix = table(pred$class, test_set$digit)
        }
        if (mod == "rda") {
            model = rda(digit ~ ., data = train_set, gamma = k[1], lambda = k[2])
            pred = predict(model, test_set)
            confusion_matrix = table(pred$class, test_set$digit)
        }
        if (mod == "rf") {
            model = randomForest(digit ~ ., data = train_set,
                ntree = k)
            pred = predict(model, newdata = test_set, type = "class")
            confusion_matrix = table(pred, test_set$digit)
        }
```

```
        test_error = test_error + 1 - (sum(diag(confusion_matrix)) / dim(test_set)[1])
        misclassified = misclassified + dim(test_set)[1] - sum(diag(confusion_matrix))
    }
    cv_error = misclassified / dim(data)[1]
    test_error = test_error / 44
    return(c(test_error, cv_error))
}
```

First of all we set a randomization seed in order to make our computations reproducible.

```
set.seed(2023)
```

We report below the calls to the `cv_44_fold_mod()` function for each method introduced at the beginning of the paragraph. Note that we also compute the *training error rate*.

```
# LDA (linear discriminant analysis)

train_error = 0
test_error = 0
cv_error = 0

pred_train = predict(lda_fit)
confusion_matrix_train = table(pred_train$class, pendigits$digit)
train_error = 1 - sum(diag(confusion_matrix_train) / dim(pendigits)[1])

error = cv_44_fold_mod(pendigits, group_cv, "lda", 0)
test_error = error[1]
cv_error = error[2]
```

```
# KNN (K-nearest neighbour) for K = 1, ..., 25, K odd

max_k = 25
num_k = ceiling(max_k / 2)

train_errors = 0
test_errors = 0
cv_errors = 0

for (k in 1:num_k){
    pred_train = knn(pendigits[, -17], pendigits[, -17], pendigits[, 17], 2 * k - 1)
    confusion_matrix_train = table(pred_train, pendigits$digit)
    train_errors[k] = 1 - sum(diag(confusion_matrix_train) / dim(pendigits)[1])

    error = cv_44_fold_mod(pendigits, group_cv, "knn", k)
    test_errors[k] = error[1]
    cv_errors[k] = error[2]
}
```

```
# QDA (quadratic discriminant analysis)

train_error = 0
test_error = 0
cv_error = 0

qda_fit = qda(digit ~ ., data = pendigits)
```

```
pred_train = predict(qda_fit, pendigits)
confusion_matrix_train = table(pred_train$class, pendigits$digit)
train_error = 1 - sum(diag(confusion_matrix_train) / dim(pendigits)[1])

error = cv_44_fold_mod(pendigits, group_cv, "qda", 0)
test_error = error[1]
cv_error = error[2]

# > Error in qda.default(x, grouping, ...) : rank deficiency in group 4
```

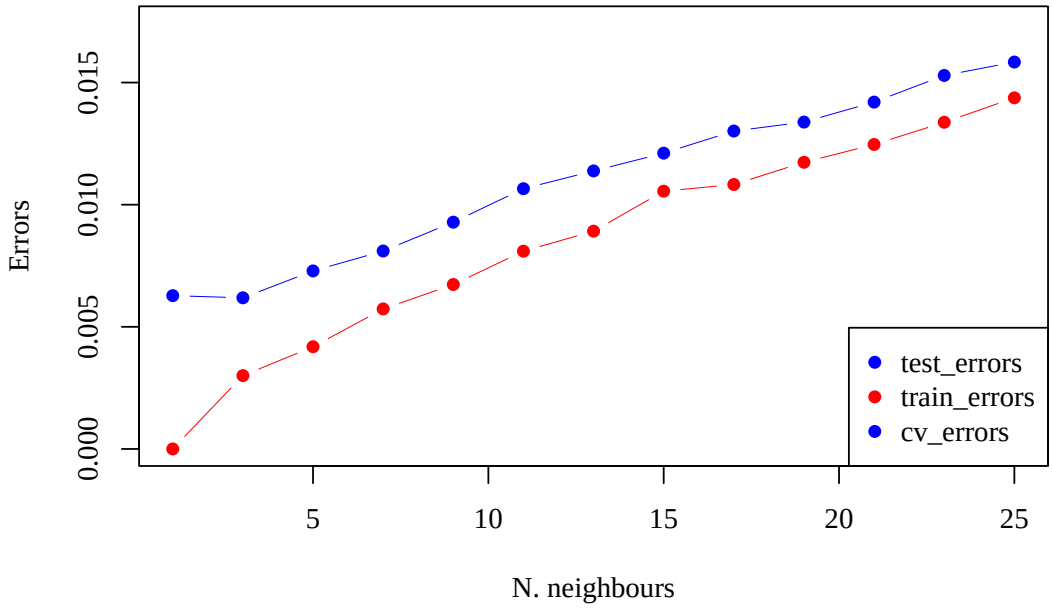
```
# Noised QDA

train_error = 0
test_error = 0
cv_error = 0

# adding noise to the dataset
pendigits_p = pendigits
count_4 = length(which(pendigits_p$digit == 4))
pendigits_p$y8[which(pendigits_p$digit == 4)] = runif(count_4, min = 0, max = 0.1)

qda_fit = qda(digit ~ ., data = pendigits_p)
pred_train = predict(qda_fit, pendigits_p)
confusion_matrix_train = table(pred_train$class, pendigits_p$digit)
train_error = 1 - sum(diag(confusion_matrix_train) / dim(pendigits_p)[1])

error = cv_44_fold_mod(pendigits_p, group_cv, "qda", 0)
test_error = error[1]
cv_error = error[2]
```

```
# RDA (regularized discriminant analysis)

train_error = 0
test_error = 0
cv_error = 0

gamma = 0
lambda = 0.2
k = c(gamma, lambda)

rda_fit = rda(digit ~ ., data = pendigits, gamma = k[1], lambda = k[2])
pred_train = predict(rda_fit, pendigits)
confusion_matrix_train = table(pred_train$class, pendigits$digit)
train_error = 1 - sum(diag(confusion_matrix_train) / dim(pendigits)[1])

error = cv_44_fold_mod(pendigits, group_cv, "rda", k)
test_error = error[1]
cv_error = error[2]
```

```
# RF (random forest)

train_error = 0
test_error = 0
cv_error = 0
```

```
pendigits_fac = pendigits
pendigits_fac$digit = as.factor(pendigits_fac$digit)

num_tree = 500

rf_fit = randomForest(digit ~ ., data = pendigits_fac,
    ntree = num_tree)
confusion_matrix_train = table(rf_fit$predicted, pendigits_fac$digit)
train_error = 1 - sum(diag(confusion_matrix_train) / dim(pendigits_fac)[1])

error = cv_44_fold_mod(pendigits_fac, group_cv, "rf", num_tree)
test_error = error[1]
cv_error = error[2]
```

Before displaying the table containining the various error rates we report the plot that helped us decide the best K parameter for the KNN classification method.



By looking at the plot, we can see that both errors increase as the number of neighbours K increases. In particular, for both errors the minimum is attained in $K = 1$. It is also remarkable that with $K = 3$ the *44-fold CV error rate* is the same as the one corresponding to $K = 1$, although the *training error rate* is of course larger (trivially because for $K = 1$ it is 0). However, we still prefer $K = 3$ as optimal choice since the model gives a more robust and reliable prediction than the one with $K = 1$.

We now display the table with the various error rates.

	LDA	3NN	Noised_QDA	RDA	RF
train_error	0.1229076	0.0030022	0.0156477	0.0192868	0.0070051
test_error	0.1242810	0.0061878	0.0172878	0.0207483	0.0076484
cv_error	0.1242722	0.0061863	0.0172853	0.0207424	0.0076419

The error rates achieved are all lower than the ones we found in point 2.4 (also reported for completeness in

the table above). In descending order of performance (with respect to the *44-fold CV error rate*), the models are *3NN*, *RF*, *Noised QDA*, *RDA* and *LDA*.

Finally, it is remarkable that *44-fold CV error rate* obtained with the *3NN* is nearly two orders of magnitude lower than the value obtained with LDA.