R-Codes

The information contained in these pages were created with the help of Helene Hopfer (and others in my laboratory) as well as Quick-R and other R-related resources on the web. A good resource is http://www.r-bloggers.com/computerworlds-beginners-guide-to-r/. I am assuming that you have R and R-Studio loaded on your computer.
To make the discussion in the file clearer I **bolded** the information that I typed into the R-Studio SOURCE EDITOR (top left side of screen).
To run code from the SOURCE EDITOR use CTRL-ENTER.
To toggle between the CONSOLE and the SOURCE EDITOR use CTRL-1 (and CTRL-2 in the opposite direction)
Output can be found in the CONSOLE (Bottom left side of screen) of R-Studio and I will place it on a grey background in this document.
To empty the CONSOLE use CTRL-L
The graphs were exported and copied from the PLOTS section of R-Studio (bottom right side).

Table of Contents

Data sets: torriFinalDa.csv; torriconsFinal.csv, torrimiss.csv; sorting_r1.csv; Author (from ca package); tea (from conjoint package). The first four data sets are available on smartsite.

Additional packages: agricolae; ca; candisc; cluster; conjoint; MASS; missmda; plotrix; pls; reshape; SensoMineR.

Additional Rcode: CVAellipses_new.R; distatiscode.R; mtable.R – these code snippets are on smartsite.

ANOVA

1. Import data into R-Studio by using the import button in the WORKSPACE (top right side of screen). Select Heading: YES; Separator: COMMA; Decimal: PERIOD; Quote: NONE

2. Check to see if the independent variables (NJ, ProductName and NR) are factors.
str(torriDAFinal)

```
> str(torriDAFinal)
'data.frame':    336 obs. of  23 variables:
 $ NJ             : Factor w/ 14 levels "1331","1400",..: 1 1 1 1 1 ...
 $ ProductName    : Factor w/ 8 levels "C_MERLOT",..: 1 3 4 2 5 8 6 ...
 $ NR             : Factor w/ 3 levels "7","8","9": 1 1 1 1 1 1 1  ...
 $ Red_berry      : num  5.1 5.6 4.9 5 3.3 5.7 2.9 3.2 0.1 1.6 ...
 $ Dark_berry     : num  5.8 1.9 2.6 1.9 7.2 3.6 5.1 6 0.1 0.7 ...
 $ Jam            : num  2.1 3.9 1.4 7.8 0.5 8.7 8.7 4 0.2 0 ...
 $ Dried_fruit    : num  4.7 1.2 5.9 0.6 5.8 1.9 0.4 0.7 2.9 6.4 ...
 $ Artificial_frui: num  1 7.9 0.8 6.6 0.7 7.4 6.2 4.1 0.1 0.1 ...
 $ Chocolate      : num  2.9 1 2 6.4 2.1 3.3 3.4 3.6 0.2 1 ...
 $ Vanilla        : num  5 8.3 2.7 5.5 1.3 6.9 8.1 4.8 2 0.8 ...
 $ Oak            : num  5 2.3 5.6 3.6 2.1 1.5 1.8 2.6 3 5.4 ...
 $ Burned         : num  1.4 1.8 1.9 3.2 5.6 0.2 0.4 4.7 7.5 5.1 ...
 $ Leather        : num  2.3 3.5 4.3 0.3 6.5 1.5 4.1 6.5 0.7 0.8 ...
 $ Earthy         : num  0.6 1 0.6 0.2 4.7 0.3 0.5 1.9 0.7 3 ...
 $ Spicy          : num  3.2 0.7 1.4 2.9 0.7 3.1 0.7 1.4 0.3 3.2 ...
 $ Pepper         : num  5.4 3 4.1 0.9 2.8 1.6 3.6 4.5 0.1 2 ...
 $ Grassy         : num  2.1 0.6 3.6 1.8 3.8 0.9 2.3 0.8 0.1 1.3 ...
 $ Medicinal      : num  0.4 2.2 1.7 0.2 2.6 0.5 0.2 3.8 0.1 2.1 ...
 $ Band.aid       : num  0.4 0.4 0.1 0.2 5.1 1.2 0.2 6.2 0.1 1.1 ...
 $ Sour           : num  5 9.7 7.8 8.3 7.6 7.2 5.9 6.3 5.7 6.4 ...
 $ Bitter         : num  5.9 5.2 3.5 3 1.9 9.8 2.9 0.2 0.6 2.9 ...
 $ Alcohol        : num  9 7.2 4.7 8.9 2.8 8.7 1.6 7 1.6 5.4 ...
 $ Astringent     : num  8.7 8.3 5 7.8 5.9 8 2.6 4.2 5.5 5.1
```

3. They were all factors. However, often one or more should be changed and you would type:
torriDAFinal$NJ=as.factor(torriDAFinal$NJ)
torriDAFinal$NR=as.factor(torriDAFinal$NR)

4. Now you should check to see if this worked by repeating step 2 or by typing
is.factor(torriDAFinal$NJ)
is.factor(torriDAfinal$NR)

```
>is.factor(torriDAFinal$NJ)
TRUE
>is.factor(torriDAFinal$NR)
TRUE
```

5. Now I created a matrix of the dependent variables (the sensory attributes)
   da.a=as.matrix(torriDAFinal [,-c(1:3)])

   In this case the three factors variables were adjacent and I could use [,–c(1:3)] but if they had been in say columns 1, 3 and 6, I would have used [,–c(1,3,6)]. Please note the MINUS sign before the c(…), this indicates that I am removing the specified attributes.

6. Now check that it had worked – we should have no independent variables in the matrix
   head(da.a)

```
> head(da.a)
     Red_berry Dark_berry Jam Dried_fruit Artificial_frui Chocolate Vanilla
[1,]       5.1        5.8 2.1         4.7             1.0       2.9     5.0
[2,]       5.6        1.9 3.9         1.2             7.9       1.0     8.3
[3,]       4.9        2.6 1.4         5.9             0.8       2.0     2.7
[4,]       5.0        1.9 7.8         0.6             6.6       6.4     5.5
[5,]       3.3        7.2 0.5         5.8             0.7       2.1     1.3 2
[6,]       5.7        3.6 8.7         1.9             7.4       3.3     6.9
     Burned Leather Earthy Spicy Pepper Grassy Medicinal Band.aid Sour Bitter
[1,]    1.4     2.3    0.6   3.2    5.4    2.1       0.4       0.4  5.0    5.9
[2,]    1.8     3.5    1.0   0.7    3.0    0.6       2.2       0.4  9.7    5.2
[3,]    1.9     4.3    0.6   1.4    4.1    3.6       1.7       0.1  7.8    3.5
[4,]    3.2     0.3    0.2   2.9    0.9    1.8       0.2       0.2  8.3    3.0
[5,]    5.6     6.5    4.7   0.7    2.8    3.8       2.6       5.1  7.6    1.9
[6,]    0.2     1.5    0.3   3.1    1.6    0.9       0.5       1.2  7.2    9.8
     Alcohol Astringent
[1,]     9.0        8.7
[2,]     7.2        8.3
[3,]     4.7        5.0
[4,]     8.9        7.8
[5,]     2.8        5.9
[6,]     8.7        8.0
```

7. Now we want to do a 3-way ANOVA with all 2-way interactions and we want to see the output from the analysis
   da.lm=lm(da.a~(NJ+ProductName+NR)^2, data=torriDAFinal)
   da.aov=aov(da.lm)
   summary(da.aov)

```
> da.lm = lm(da.a~(NJ+ProductName+NR)^2, data=torriDAFinal)
> da.aov=aov(da.lm)
> summary(da.aov)
 Response Red_berry :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 597.86  45.990 12.8214 < 2.2e-16 ***
ProductName      7  73.16  10.451  2.9137  0.006517 **
NR               2   2.86   1.429  0.3983  0.672014
NJ:ProductName  91 659.03   7.242  2.0190 3.177e-05 ***
NJ:NR           26  97.17   3.737  1.0419  0.415650
ProductName:NR  14  52.25   3.732  1.0405  0.415394
Residuals      182 652.82   3.587
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
 Response Dark_berry :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 812.64  62.511 15.5252 < 2.2e-16 ***
ProductName      7 126.10  18.014  4.4739 0.0001283 ***
NR               2  18.94   9.470  2.3520 0.0980671 .
NJ:ProductName  91 703.51   7.731  1.9200 0.0001040 ***
NJ:NR           26 108.58   4.176  1.0372 0.4216995
ProductName:NR  14  29.04   2.074  0.5152 0.9223301
Residuals      182 732.80   4.026
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Jam :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 505.16  38.859 15.0508 < 2.2e-16 ***
ProductName      7 280.56  40.079 15.5235 7.015e-16 ***
NR               2  12.51   6.254  2.4222  0.091585 .
NJ:ProductName  91 479.99   5.275  2.0430 2.377e-05 ***
NJ:NR           26 136.32   5.243  2.0308  0.003791 **
ProductName:NR  14  63.98   4.570  1.7701  0.046005 *
Residuals      182 469.90   2.582
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

I CUT OUT THE REST OF THE OUTPUT TO SAVE SPACE
```

8. Now I needed to test the pseudomixed model. I looked at the output and determined which attributes had a significant Productname effect and also a significant NJ*Productname and NJ*Productname effect. In this example I only did the first three attributes.
    a. First I had to create an output file of the da.aov that I could use for calculations
       aovsum = summary(da.aov)

    b. Then I had to tell R the degrees of freedom for the Productname, NJ and NR. I just told it to use the degrees of freedom for Red_berry (since they are the same for all the variables), hence the [1]. Then I told it to look in [2,1] which means row 2 column 1 for the product df, in [4,1] or row 4 column 1 for the judge product interaction df and in [6,1] or row 6, column 1 for the product rep interaction df.
       df_P = aovsum[1][2,1]
       df_PJ =aovsum[1][4,1]
       df_PR=aovsum[1][6,1]

    c. Then I had R find the new critical F-values for the pseudomixed model involving judges and for the pseudomixed model involving replications
       newF_crit1=qf(0.95,df_P, df_PJ)
       newF_crit2=qf(0.95,df_P, df_PR)

    d. Then I tested the models
       aovsum[[1]][2,3]/(aovsum[[1]][4,3])>newF_crit1
       aovsum[[2]][2,3]/(aovsum[[1]][4,3])>newF_crit1
       aovsum[[3]][2,3]/(aovsum[[1]][4,3])>newF_crit1
       aovsum[[3]][2,3]/(aovsum[[1]][6,3])>newF_crit1

and found that Red Berry becomes not significant with the pseudomixed model while the other Dark Fruit and Jam stay significant (Jam was tested for both the product judge and the product rep interactions.

```
> aovsum=summary(da.aov)

> df_P = aovsum[[1]][2,1]
> df_PJ = aovsum[[1]][4,1]
> df_PR = aovsum[[1]][6,1]
> newF_crit1 = qf(0.95, df_P, df_PJ)
> newF_crit2 = qf(0.95, df_P, df_PR)

> aovsum[[1]][2,3]/(aovsum[[1]][4,3]) > newF_crit1  #
[1] FALSE
> aovsum[[2]][2,3]/(aovsum[[2]][4,3]) > newF_crit1  #
[1] TRUE
> aovsum[[3]][2,3]/(aovsum[[3]][4,3]) > newF_crit1  #
[1] TRUE
> aovsum[[3]][2,3]/(aovsum[[3]][6,3]) > newF_crit1
[1] TRUE
```

9. Next I wanted to calculate the LSD values for the significant attributes. I need to load the agricolae package first and check that it was loaded **library(agricolae)**

10. Then I had to ask R to calculate the LSD for each attribute (this is a little tedious but cutting and pasting helps). Since I could not remember the exact names of the attributes I asked to see those with the head(da.a) request. I also asked for the HSD for Jam to show you how to do that.
    head(da.a)
    lsd.darkb =LSD.test(lm(Dark_berry~(NJ+ProductName+NR)^2, data=torriDAFinal), "ProductName")
    lsd.Jam=LSD.test(lm(Jam~(NJ+ProductName+NR)^2, data=torriDAFinal), "ProductName")
    lsd.Jam2 =HSD.test(lm(Jam~(NJ+ProductName+NR)^2, data=torriDAFinal), "ProductName")

```
> lsd.darkb =LSD.test(lm(Dark_berry~(NJ+ProductName+NR)^2,
data=torriDAFinal), "ProductName")

Study:
LSD t Test for Dark_berry
Mean Square Error:  7.431715

ProductName,  means and individual ( 95 %) CI
             Dark_berry   std.err  r       LCL      UCL Min. Max.
C_MERLOT       3.047619 0.4013658 42 2.257883 3.837355    0  8.3
C_REFOSCO      2.461905 0.3831217 42 1.708066 3.215743    0  8.6
C_SYRAH        2.933333 0.4487778 42 2.050309 3.816358    0  8.9
C_ZINFANDEL    3.059524 0.4142645 42 2.244408 3.874639    0  9.2
I__MERLOT      2.350000 0.3406670 42 1.679696 3.020304    0  7.2
I_PRIMITIVO    3.380952 0.4668355 42 2.462397 4.299508    0  9.6
I_REFOSCO      3.007143 0.3642311 42 2.290474 3.723812    0  8.5
I_SYRAH        4.483333 0.4993831 42 3.500737 5.465930    0  9.9
```

```
alpha: 0.05 ; Df Error: 311
Critical Value of t: 1.967621

Least Significant Difference 1.170513
Means with the same letter are not significantly different.

Groups, Treatments and means
a        I_SYRAH         4.483
ab       I_PRIMITIVO     3.381
b        C_ZINFANDEL     3.06
b        C_MERLOT        3.048
b        I_REFOSCO       3.007
b        C_SYRAH         2.933
b        C_REFOSCO       2.462
b        I__MERLOT       2.35
```

```
> lsd.Jam=LSD.test(lm(Jam~(NJ+ProductName+NR)^2, data=torriDAFinal),
"ProductName")

Study:
LSD t Test for Jam
Mean Square Error:  4.649768

ProductName,  means and individual ( 95 %) CI
                 Jam   std.err  r       LCL      UCL Min. Max.
C_MERLOT    1.3714286 0.2622224 42 0.8554743 1.887383    0  6.2
C_REFOSCO   1.0309524 0.2378887 42 0.5628775 1.499027    0  7.8
C_SYRAH     1.7452381 0.3783975 42 1.0006951 2.489781    0 10.0
C_ZINFANDEL 1.9785714 0.3940548 42 1.2032209 2.753922    0 10.0
I__MERLOT   0.8428571 0.1850116 42 0.4788244 1.206890    0  4.8
I_PRIMITIVO 3.6119048 0.4754141 42 2.6764698 4.547340    0  9.8
I_REFOSCO   1.5357143 0.2823047 42 0.9802456 2.091183    0  7.1
I_SYRAH     3.0976190 0.4525510 42 2.2071701 3.988068    0  9.5

alpha: 0.05 ; Df Error: 311
Critical Value of t: 1.967621

Least Significant Difference 0.9258646
Means with the same letter are not significantly different.

Groups, Treatments and means
a        I_PRIMITIVO     3.612
a        I_SYRAH         3.098
b        C_ZINFANDEL     1.979
bc       C_SYRAH         1.745
bc       I_REFOSCO       1.536
bc       C_MERLOT        1.371
c        C_REFOSCO       1.031
c        I__MERLOT       0.8429
```

```
> lsd.Jam2 =HSD.test(lm(Jam~(NJ+ProductName+NR)^2, data=torriDAFinal),
"ProductName")

Study:
HSD Test for Jam
Mean Square Error:  4.649768

ProductName,  means
                 Jam   std.err  r Min. Max.
C_MERLOT    1.3714286 0.2622224 42    0  6.2
C_REFOSCO   1.0309524 0.2378887 42    0  7.8
```

```
C_SYRAH     1.7452381 0.3783975 42    0 10.0
C_ZINFANDEL 1.9785714 0.3940548 42    0 10.0
I__MERLOT   0.8428571 0.1850116 42    0  4.8
I_PRIMITIVO 3.6119048 0.4754141 42    0  9.8
I_REFOSCO   1.5357143 0.2823047 42    0  7.1
I_SYRAH     3.0976190 0.4525510 42    0  9.5

alpha: 0.05 ; Df Error: 311
Critical Value of Studentized Range: 4.315746

Honestly Significant Difference: 1.435975

Means with the same letter are not significantly different.

Groups, Treatments and means
a        I_PRIMITIVO    3.612
ab       I_SYRAH        3.098
bc       C_ZINFANDEL    1.979
bc       C_SYRAH        1.745
c        I_REFOSCO      1.536
c        C_MERLOT       1.371
c        C_REFOSCO      1.031
c        I__MERLOT      0.8429
```

MISSING DATA

To do imputation I removed some random values from the torriDAFinal.csv and created a new file called torrimiss.csv which I imported into R-Studio.

1. First I checked that the NJ and NR were factors
   is.factor(torrimiss$NJ)
   is.factor(torrimiss$NJ)

```
> is.factor(torrimiss$NJ)
[1] TRUE
> is.factor(torrimiss$NR)
[1] TRUE
```

   Since they were I did not need to do anything else. However, if they had not been factors I would have followed the same process as on page 2.

2. I wanted to show you part of the missing data set and to do that I typed
   head(torrimiss)

```
> head(torrimiss)
    NJ ProductName NR Red_berry Dark_berry Jam Dried_fruit Artificial_frui
1 1331    C_MERLOT  7        NA        5.8 2.1         4.7             1.0
2 1331     C_SYRAH  7       5.6        1.9 3.9         1.2             7.9
3 1331 C_ZINFANDEL  7       4.9        2.6 1.4         5.9             0.8
4 1331   C_REFOSCO  7       5.0        1.9 7.8         0.6             6.6
5 1331   I__MERLOT  7       3.3        7.2 0.5         5.8             0.7
6 1331     I_SYRAH  7        NA         NA  NA          NA              NA
  Chocolate Vanilla Oak Burned Leather Earthy Spicy Pepper Grassy Medicinal
1       2.9      NA 5.0    1.4     2.3    0.6   3.2    5.4    2.1       0.4
2       1.0     8.3 2.3    1.8     3.5    1.0   0.7    3.0    0.6       2.2
3       2.0     2.7 5.6    1.9     4.3    0.6   1.4    4.1    3.6       1.7
4       6.4     5.5 3.6    3.2     0.3    0.2   2.9    0.9    1.8       0.2
5       2.1     1.3 2.1    5.6     6.5    4.7   0.7    2.8    3.8       2.6
6        NA      NA  NA     NA      NA     NA    NA     NA     NA        NA
  Band.aid Sour Bitter Alcohol Astringent
1      0.4  5.0    5.9     9.0        8.7
2      0.4  9.7    5.2     7.2        8.3
3      0.1  7.8    3.5     4.7        5.0
4      0.2  8.3    3.0     8.9        7.8
5      5.1  7.6    1.9     2.8        5.9
6       NA   NA     NA      NA         NA
```

3. Now I needed to impute the missing values by first loading the missmda package in the lower right side of the screen

4. I checked to see if missmda was loaded
   library(missmda)

5. Once missmda was loaded I typed
   da.imp=imputeFAMD(torrimiss, ncp = 2)

6. To check that this had worked I typed
   head(da.imp$completeObs)

```
> da.imp=imputeFAMD(torrimiss, ncp = 2)
> head(da.imp$completeObs)
    NJ ProductName NR Red_berry Dark_berry  Jam Dried_fruit Artificial_frui
1 1331  C_MERLOT  7  3.980503  5.800000  2.100000  4.700000  1.000000
2 1331  C_SYRAH  7  5.600000  1.900000  3.900000  1.200000  7.900000
3 1331  C_ZINFANDEL  7  4.900000  2.600000  1.400000  5.900000  0.800000
4 1331  C_REFOSCO  7  5.000000  1.900000  7.800000  0.600000  6.600000
5 1331  I_MERLOT  7  3.300000  7.200000  0.500000  5.800000  0.700000
6 1331  I_SYRAH  7  3.756728  4.240235  2.896095  2.990164  1.994639

  Chocolate  Vanilla   Oak   Burned   Leather   Earthy   Spicy   Pepper
1    2.9000 2.450956 5.000000 1.400000  2.3000 0.600000 3.200000 5.400000
2    1.0000 8.300000 2.300000 1.800000  3.5000 1.000000 0.700000 3.000000
3    2.0000 2.700000 5.600000 1.900000  4.3000 0.600000 1.400000 4.100000
4    6.4000 5.500000 3.600000 3.200000  0.3000 0.200000 2.900000 0.900000
5    2.1000 1.300000 2.100000 5.600000  6.5000 4.700000 0.700000 2.800000
6    2.2319 2.472598 3.279971 2.065501  2.6897 2.150186 2.192773 2.719767

    Grassy   Medicinal   Band.aid   Sour   Bitter   Alcohol   Astringent
1 2.100000  0.400000 0.400000 5.000000 5.900000 9.000000   8.700000
2 0.600000  2.200000 0.400000 9.700000 5.200000 7.200000   8.300000
3 3.600000  1.700000 0.100000 7.800000 3.500000 4.700000   5.000000
4 1.800000  0.200000 0.200000 8.300000 3.000000 8.900000   7.800000
5 3.800000  2.600000 5.100000 7.600000 1.900000 2.800000   5.900000
6 1.711911  2.463939 2.317472 5.831653 4.068066 4.471512   5.728887
```

7. If I had wanted to save this file I could have typed
   write.table(da.imp$completeObs, file="torrinm.csv", sep=",")
   This would have placed the torrinm.csv file in my home directory and I could then have opened it in EXCEL.

8. I wanted to do an ANOVA on the imputed file to see how different the results were from the original file so I repeated the steps in the ANOVA section
   danm.a=as.matrix(da.imp$completeObs [,-c(1:3)])
   head(danm.a)
   danm.lm = lm(danm.a~(NJ+ProductName+NR)^2, data=da.imp$completeObs)
   summary(danm.lm)
   danm.aov=aov(danm.lm)
   summary(danm.aov)

```
> danm.a=as.matrix(da.imp$completeObs [,-c(1:3)])
> head(danm.a)
     Red_berry Dark_berry      Jam Dried_fruit Artificial_frui Chocolate
[1,]  3.980503   5.800000 2.100000    4.700000        1.000000    2.9000
[2,]  5.600000   1.900000 3.900000    1.200000        7.900000    1.0000
[3,]  4.900000   2.600000 1.400000    5.900000        0.800000    2.0000
[4,]  5.000000   1.900000 7.800000    0.600000        6.600000    6.4000
[5,]  3.300000   7.200000 0.500000    5.800000        0.700000    2.1000
[6,]  3.756728   4.240235 2.896095    2.990164        1.994639    2.2319
      Vanilla      Oak   Burned Leather   Earthy    Spicy   Pepper   Grassy
[1,] 2.450956 5.000000 1.400000  2.3000 0.600000 3.200000 5.400000 2.100000
[2,] 8.300000 2.300000 1.800000  3.5000 1.000000 0.700000 3.000000 0.600000
[3,] 2.700000 5.600000 1.900000  4.3000 0.600000 1.400000 4.100000 3.600000
[4,] 5.500000 3.600000 3.200000  0.3000 0.200000 2.900000 0.900000 1.800000
```

```
[5,] 1.300000 2.100000 5.600000  6.5000 4.700000 0.700000 2.800000 3.800000
[6,] 2.472598 3.279971 2.065501  2.6897 2.150186 2.192773 2.719717 1.711911
     Medicinal Band.aid     Sour   Bitter  Alcohol Astringent
[1,]  0.400000 0.400000 5.000000 5.900000 9.000000   8.700000
[2,]  2.200000 0.400000 9.700000 5.200000 7.200000   8.300000
[3,]  1.700000 0.100000 7.800000 3.500000 4.700000   5.000000
[4,]  0.200000 0.200000 8.300000 3.000000 8.900000   7.800000
[5,]  2.600000 5.100000 7.600000 1.900000 2.800000   5.900000
[6,]  2.463939 2.317472 5.831653 4.068066 4.471512   5.728887
```

```
> danm.lm = lm(danm.a~(NJ+ProductName+NR)^2, data=da.imp$completeObs)
> #summary(danm.lm)
> danm.aov=aov(danm.lm)
> summary(danm.aov)
 Response Red_berry :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 580.55  44.658 13.0670 < 2.2e-16 ***
ProductName      7  81.26  11.608  3.3966  0.001962 **
NR               2   2.89   1.443  0.4223  0.656149
NJ:ProductName  91 638.56   7.017  2.0533 2.098e-05 ***
NJ:NR           26 102.00   3.923  1.1479  0.292610
ProductName:NR  14  47.95   3.425  1.0023  0.452935
Residuals      182 622.00   3.418
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Dark_berry :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 812.64  62.511 15.5695 < 2.2e-16 ***
ProductName      7 122.51  17.502  4.3592 0.0001717 ***
NR               2  17.02   8.512  2.1201 0.1229685
NJ:ProductName  91 694.39   7.631  1.9006 0.0001310 ***
NJ:NR           26 103.59   3.984  0.9923 0.4806501
ProductName:NR  14  28.27   2.019  0.5030 0.9294121
Residuals      182 730.73   4.015
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Jam :
                Df Sum Sq Mean Sq F value    Pr(>F)
NJ              13 478.63  36.818 14.0423 < 2.2e-16 ***
ProductName      7 277.35  39.621 15.1116 1.668e-15 ***
NR               2  10.85   5.426  2.0695 0.1292143
NJ:ProductName  91 439.70   4.832  1.8429 0.0002576 ***
NJ:NR           26 130.77   5.030  1.9183 0.0072351 **
ProductName:NR  14  62.40   4.457  1.7001 0.0587740 .
Residuals      182 477.19   2.622
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
I CUT OUT THE REST OF THE OUTPUT TO SAVE SPACE
```

Compare the MS and F-values to the original – some slight differences

MANOVA

9. I now decided to do a three-way MANOVA with two-way interactions on the complete torriDAFinal data set.
   da.man<-manova(da.a~(NJ+ProductName+NR)^2, data=torriDAFinal)
   summary(da.man, test="Wilks")

```
> da.man=manova(da.a~(NJ+ProductName+NR)^2, data=torriDAFinal)
> summary(da.man, test="Wilks")
               Df   Wilks approx F num Df den Df    Pr(>F)
NJ             13 0.00001  12.5493    260 1819.7 < 2.2e-16 ***
ProductName     7 0.04243   4.7575    140 1093.6 < 2.2e-16 ***
NR              2 0.64784   1.9756     40  326.0 0.0007056 ***
NJ:ProductName 91 0.00000   1.7583   1820 3331.1 < 2.2e-16 ***
NJ:NR          26 0.02215   1.3926    520 2672.6 1.873e-07 ***
ProductName:NR 14 0.22673   0.9414    280 1916.9 0.7387371
Residuals     182
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

10. I also decided to do a one-way MANOVA on the imputed dataset
    danm.man=manova(danm.a~ProductName, data=da.imp$completeObs)
    summary(danm.man, test="Wilks")

```
> danm.man=manova(danm.a~ProductName, data=da.imp$completeObs)
> summary(danm.man, test="Wilks")
            Df  Wilks  approx F num Df  den Df    Pr(>F)
ProductName 7   0.31412  2.8067  140     2063.5    < 2.2e-16 ***
Residuals   328
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

CANONICAL VARIATE ANALYSIS (CVA)

1. I wanted to do a CVA on the torriDAFinal.csv data set.

2. As discussed in class[1] for a CVA you want to use only a *one-way* MANOVA unless you are specifically interested in a specific two-way interaction where you would create a one-way MANOVA of the combined factors.
   da.cvaman=manova(da.a~ProductName, data=torriDAFinal)
   summary(da.cvaman, test="Wilks")

```
> da.man=manova(da.a~ProductName, data=torriDAFinal)
> summary(da.man, test="Wilks")
             Df   Wilks approx F num Df den Df    Pr(>F)
ProductName   7 0.30799   2.8588    140 2063.5 < 2.2e-16 ***
Residuals   328
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

3. Now I needed to load the package that will do a CVA. Load the candisc package in the lower right side of the screen.

4. I checked to see if the candisc package was loaded
   library(candisc)

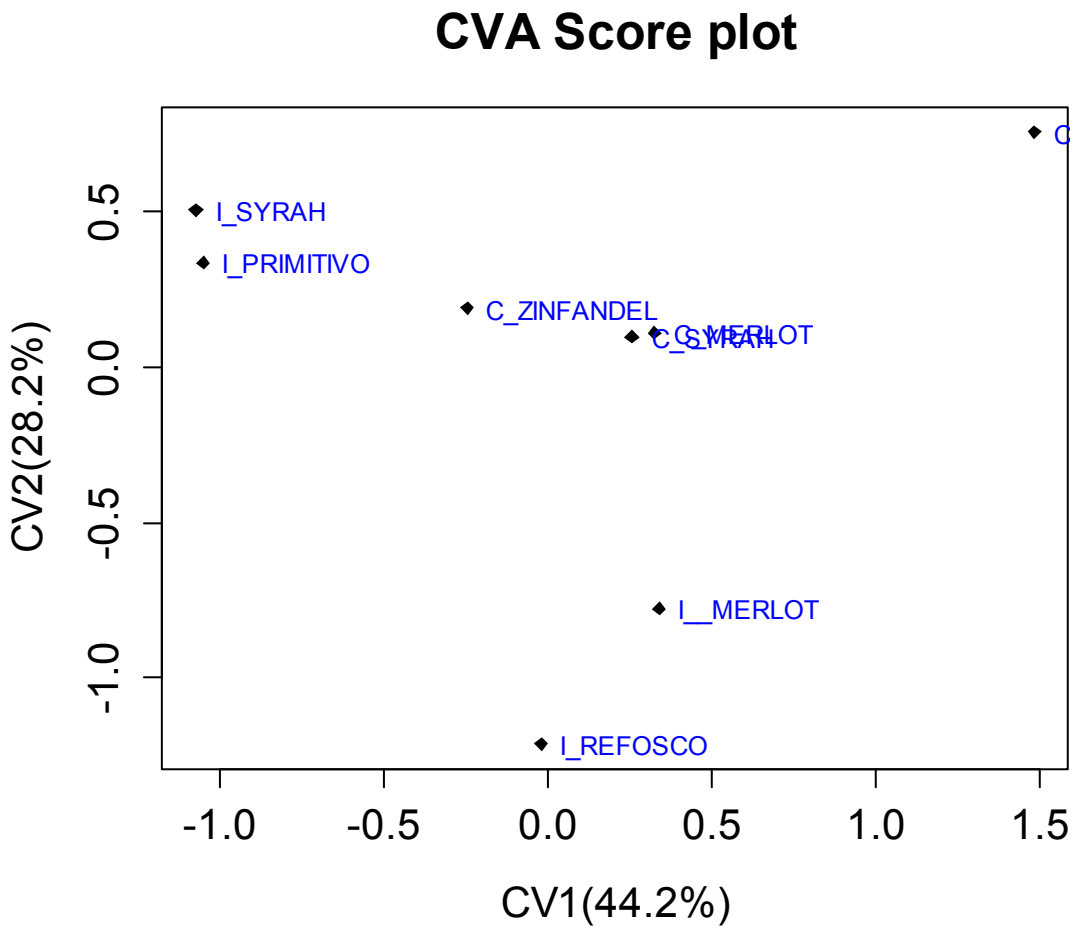
5. Then I ran the CVA and then asked to see the output
   da.cva=candisc(da.cvaman)
   da.cva

```
> library(candisc)
> da.cva=candisc(da.man)
> da.cva

Canonical Discriminant Analysis for ProductName:

    CanRsq Eigenvalue Difference Percent Cumulative
1 0.381405   0.616566    0.22367 44.2257     44.226
2 0.282071   0.392895    0.22367 28.1820     72.408
3 0.121339   0.138095    0.22367  9.9054     82.313
4 0.109005   0.122340    0.22367  8.7754     91.089
5 0.059767   0.063566    0.22367  4.5595     95.648
6 0.041480   0.043275    0.22367  3.1041     98.752
7 0.017100   0.017397    0.22367  1.2479    100.000

Test of H0: The canonical correlations in the
current row and all that follow are zero
```

[1] see Monrozier, R.; Danzart, M. (2001) Food Quality and Preference 12:393–406

```
  LR test stat approx F num Df  den Df   Pr(> F)
1      0.30799   8.7340     49 1639.16 < 2.2e-16 ***
2      0.49788   6.7946     36 1421.15 < 2.2e-16 ***
3      0.69349   4.9913     25 1205.11 2.346e-14 ***
4      0.78926   5.0014     16  993.53 4.680e-10 ***
5      0.88582   4.5036      9  793.55 8.593e-06 ***
6      0.94213   4.9467      4  654.00 0.0006202 ***
7      0.98290   5.7063      1  328.00 0.0174689 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

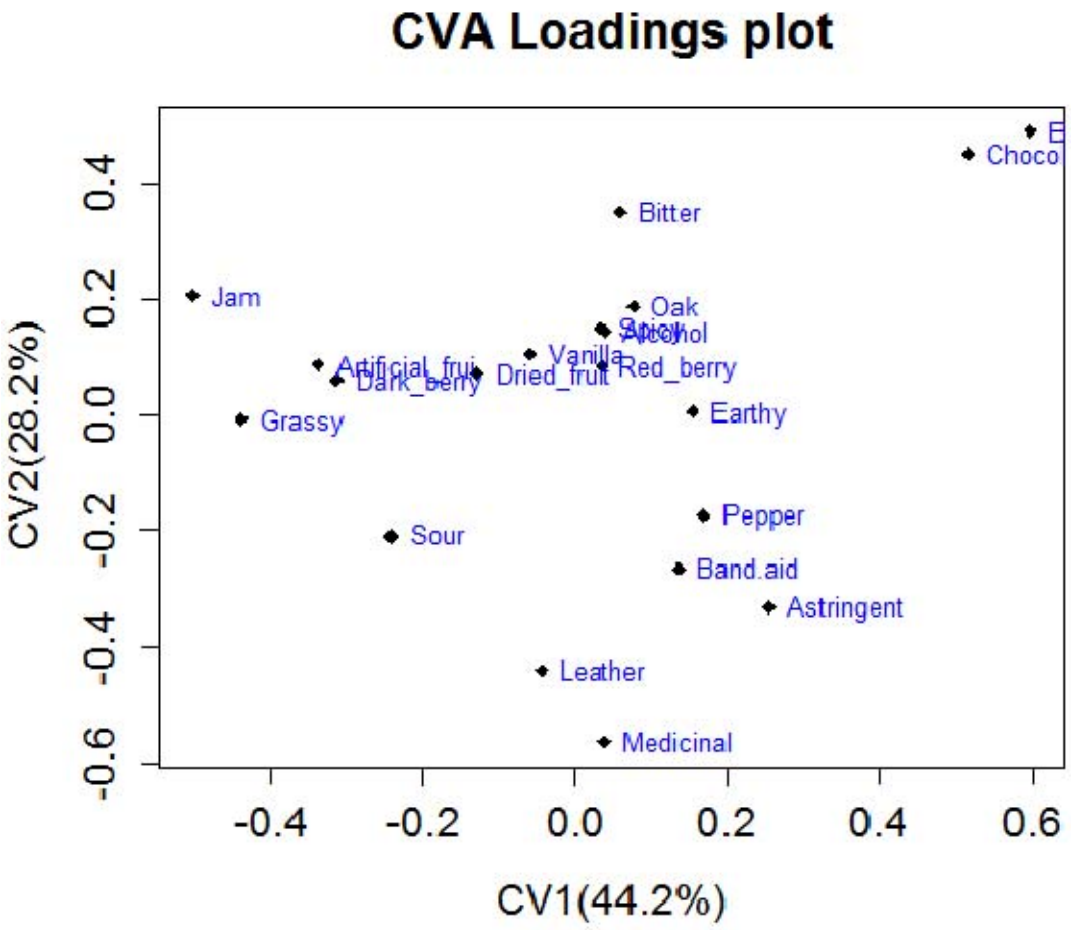
6. To plot the score plot I typed
   plot(da.cva$means[,1:2], main="CVA Score plot", xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
   text(da.cva$means[,1:2], row.names(da.cva$means), cex=0.6, pos=4, col="blue")



7. To plot the loadings plot I needed to find out what candisc calls the TOTAL STRUCTURE, so I typed the following and found out that it is structure
attributes(da.cva)

```
> attributes(da.cva)
$names
 [1] "dfh"         "dfe"         "eigenvalues" "canrsq"      "pct"
 [6] "rank"        "ndim"        "means"       "factors"     "term"
[11] "terms"       "coeffs.raw"  "coeffs.std"  "structure"   "scores"

$class
[1] "candisc
```

8. I asked R to plot the loadings by typing
plot(da.cva$structure[,1:2], main="CVA Loadings plot",
xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
text(da.cva$structure[,1:2], row.names(da.cva$structure), cex=0.6, pos=4,
col="blue")



9. For interest and comparison I then repeated the entire CVA analysis on the imputed file by typing

danm.man=manova(danm.a~ProductName, data=da.imp$completeObs)
summary(danm.man, test="Wilks")
danm.cva=candisc(danm.man)
danm.cva

plot(danm.cva$means[,1:2], main="CVA Score plot",
xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
text(danm.cva$means[,1:2], row.names(danm.cva$means), cex=0.6, pos=4, col="blue")

plot(danm.cva$coeffs.std[,1:2], main="CVA Loadings plot",
xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
text(danm.cva$coeffs.std[,1:2], row.names(danm.cva$coeffs.std), cex=0.6, pos=4, col="blue")

```
> danm.cva=candisc(danm.man)
> danm.cva

Canonical Discriminant Analysis for ProductName:

    CanRsq Eigenvalue Difference Percent Cumulative
1 0.366660   0.578930    0.18809 42.5504     42.550
2 0.281013   0.390845    0.18809 28.7264     71.277
3 0.122353   0.139410    0.18809 10.2464     81.523
4 0.114022   0.128696    0.18809  9.4590     90.982
5 0.059901   0.063718    0.18809  4.6832     95.665
6 0.039695   0.041336    0.18809  3.0381     98.704
7 0.017334   0.017639    0.18809  1.2965    100.000

Test of H0: The canonical correlations in the
current row and all that follow are zero

  LR test stat approx F num Df  den Df    Pr(> F)
1      0.31412   8.5706     49 1639.16 < 2.2e-16 ***
2      0.49597   6.8352     36 1421.15 < 2.2e-16 ***
3      0.68981   5.0676     25 1205.11 1.155e-14 ***
4      0.78598   5.0930     16  993.53 2.663e-10 ***
5      0.88713   4.4473      9  793.55 1.051e-05 ***
6      0.94366   4.8101      4  654.00 0.0007889 ***
7      0.98267   5.7857      1  328.00 0.0167100 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Compare to the original results – some slight differences

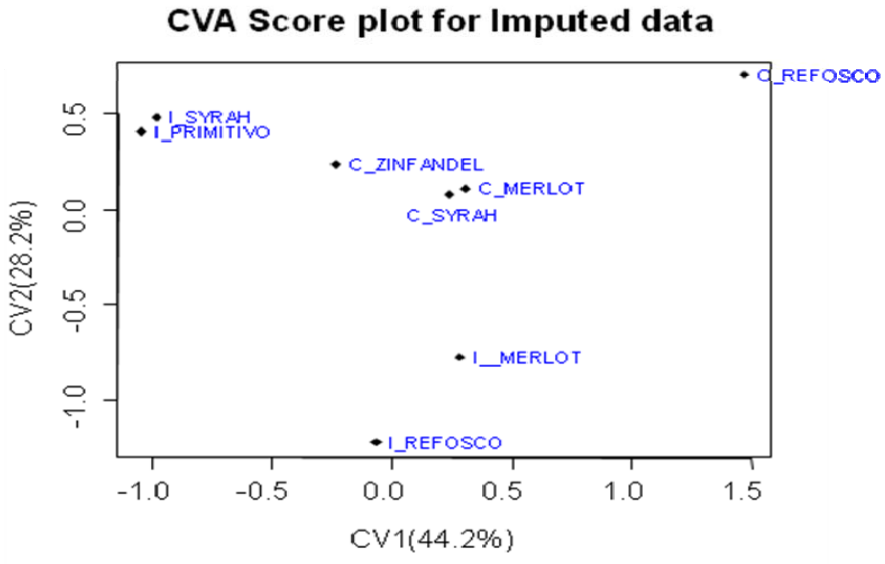
```
> plot(danm.cva$means[,1:2], main="CVA Score plot for Imputed data",
+      xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
> text(danm.cva$means[,1:2], row.names(danm.cva$means), cex=0.6, pos=4,
col="blue")
> plot(danm.cva$structure[,1:2], main="CVA Loadings plot for Imputed Data",
+      xlab="CV1(44.2%)", ylab="CV2(28.2%)", pch=18, col="black")
> text(danm.cva$structure[,1:2], row.names(danm.cva$structure), cex=0.6,
pos=4, col="blue")
```

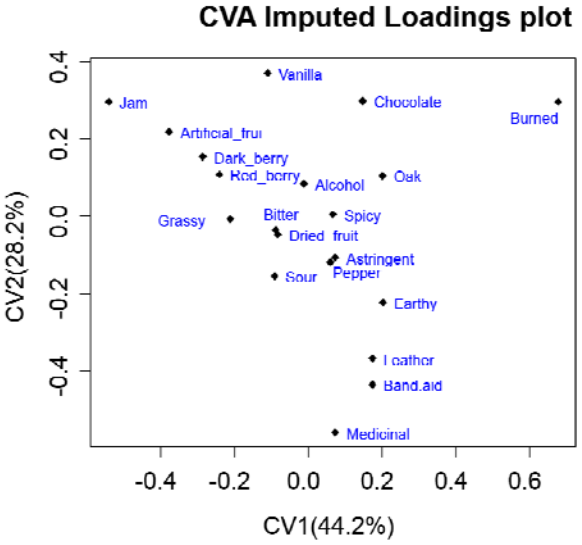
These plots follow but I did some editing in PowerPoint to make them ‘prettier’.

PLOTS to POWERPOINT

As an aside:

You can export the plots from the lower right hand screen by going to EXPORT. It is easiest to ask to COPY TO CLIPBOARD as a METAFILE. Then paste in PowerPoint. In PowerPoint you right click on the plot and click on EDIT GRAPH and then say yes to convert the graph. Now you can move the titles around, add information, etc. You can then paste the completed graph into WORD, etc. from PPT..





10. CVA graphs with 95% confidence circles based on Chatfield and Collins, 1986 with thanks to Helene Hopfer.
    a. In the first instance I set the margins for the graphs and then I tell it I want both graphs on the same page
    par(mar=c(4,4,.5,0))
    par(mfcol=c(1,2))

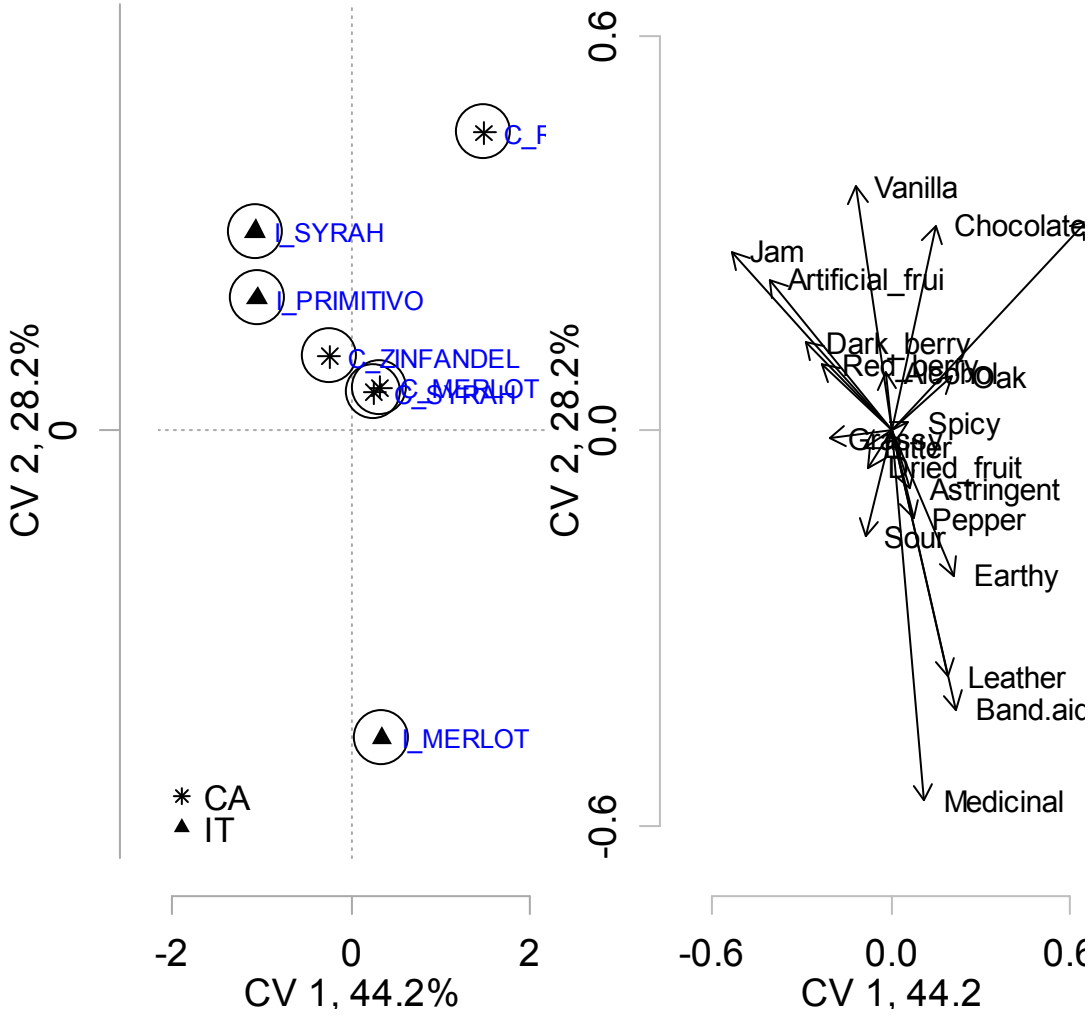
    b. Now I plot the SCORE PLOT as before, but I control more of the settings. I ask for a specific symbol for the CA wines and a different symbol for the IT wines. Then I specify the axis labels, the size of the fonts, the axes widths and the label font sizes, as well as the indications of the size of the axes (in this case -2 to 2). The abline adds the dotted line through the (0,0). I also added a legend.
    plot(da.cva$means[,1:2], pch=c(8,8,8,8,17,17,17,17), xlab='CV 1, 44.2%', ylab='CV 2, 28.2%', cex=1.1, xlim=c(-2,2), ylim=c(-1,1),axes=FALSE, cex.lab=.9)
    axis(side = 1, at = c(-2,0,2), line=1, col='darkgray', cex.axis=.9)
    axis(side = 2, at = c(-2,0,2), line=1, col='darkgray', cex.axis=.9)
    abline(h=0,v=0, col='darkgray', lty=3, lwd=.8)
    legend(x='bottomleft', c('CA','IT'), cex=0.8, pch=c(8,17), bty='n')

    c. Then I added the 95% confidence intervals using the Chapman and Collins method and the text to label each circle. I could have added an abline statement here to but I chose not to.
    symbols(x=da.cva$means[,1], y=da.cva$means[,2], circles=2/sqrt(table(da.cva$scores[,1])), add=TRUE, inches=FALSE, lty=2, lwd=.5, fg='black')
    text(da.cva$means[,1:2], row.names(da.cva$means), cex=0.6, pos=4, col="blue")

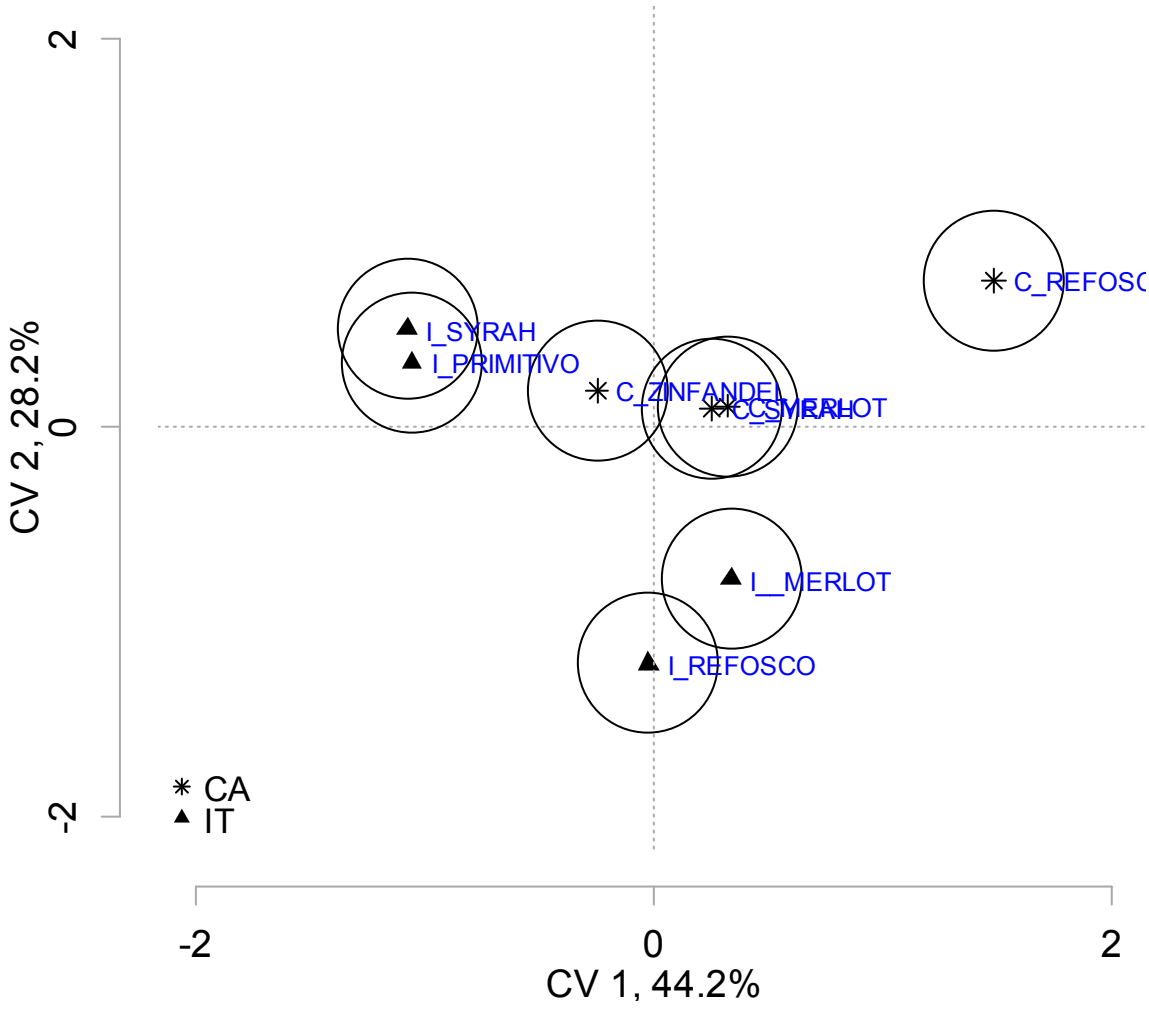
    d. The next section specifies the information for the LOADINGS PLOT where I added the vectors
    plot(da.cva$structure[,1:2], pch='', xlab="CV 1, 44.2", ylab="CV 2, 28.2%", xlim=c(-.6,.6), ylim=c(-.6,.6), axes=FALSE, cex.lab=.9)
    Axis(side=1, at=c(-.6,0,.6), line=1, cex.axis=.9, col='gray')
    Axis(side=2, at=c(-.6,0,.6), line=1, cex.axis=.9, col='gray')
    arrows(0,0,da.cva$structure[,1],da.cva$structure[,2], length=0.1)
    text(da.cva$structure[,1:2], labels=row.names(da.cva$structure),pos=4, cex=.7)

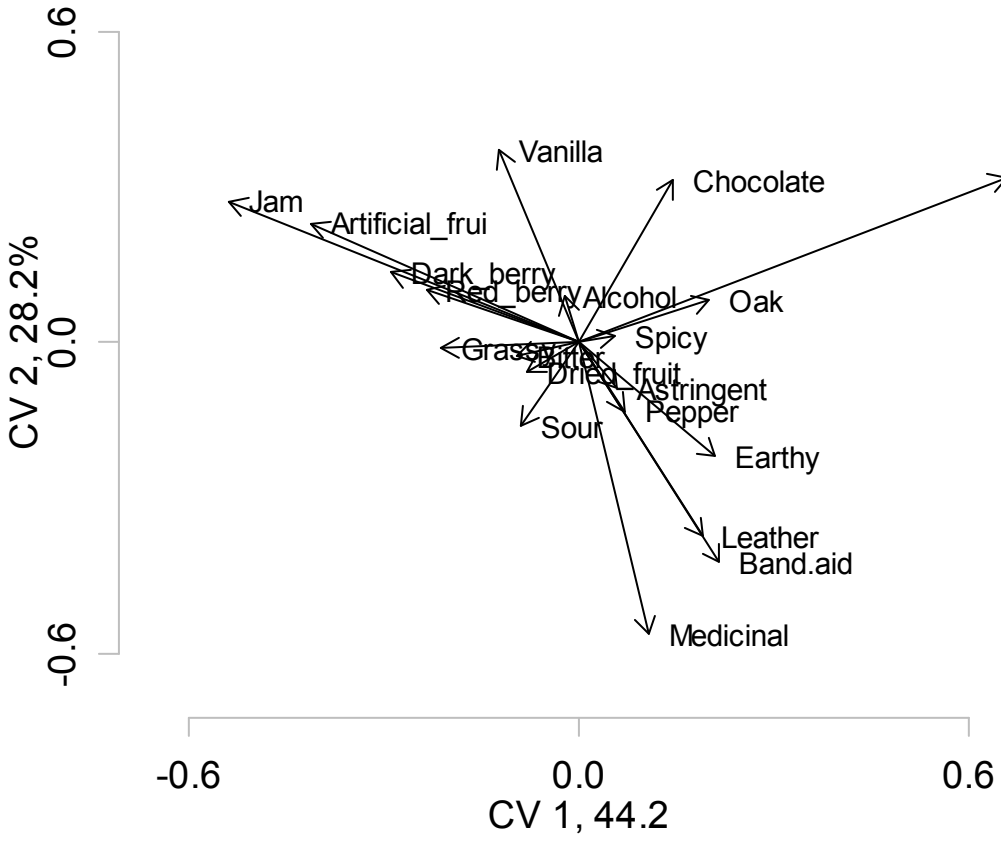
```
# nice CVA plot
>
> par(mar=c(4,4,.5,0))
> par(mfcol=c(1,2))
>
> # CV12
>
> plot(da.cva$means[,1:2], pch=c(8,8,8,8,17,17,17,17),
```

```
+      xlab='CV 1, 44.2%', ylab='CV 2, 28.2%', cex=1.1, xlim=c(-2,2),
ylim=c(-2,2),axes=FALSE, cex.lab=.9)
> axis(side = 1, at = c(-2,0,2), line=1, col='darkgray', cex.axis=.9)
> axis(side = 2, at = c(-2,0,2), line=1, col='darkgray', cex.axis=.9)
> abline(h=0,v=0, col='darkgray', lty=3, lwd=.8)
> legend(x='bottomleft', c('CA','IT'), cex=0.8, pch=c(8,17), bty='n')
> symbols(x=da.cva$means[,1], y=da.cva$means[,2],
circles=2/sqrt(table(da.cva$scores[,1])),
+         add=TRUE, inches=FALSE, lty=2, lwd=.5, fg='black')
> text(da.cva$means[,1:2], row.names(da.cva$means), cex=0.6, pos=4,
col="blue")
> plot(da.cva$struucture[,1:2], pch='', xlab="CV 1, 44.2", ylab="CV 2,
28.2%", xlim=c(-.6,.6), ylim=c(-.6,.6), axes=FALSE, cex.lab=.9)
> Axis(side=1, at=c(-.6,0,.6), line=1, cex.axis=.9, col='gray')
> Axis(side=2, at=c(-.6,0,.6), line=1, cex.axis=.9, col='gray')
> arrows(0,0,da.cva$structure[,1],da.cva$structure[,2], length=0.1)
> text(da.cva$structure[,1:2], labels=row.names(da.cva$structure),pos=4,
cex=.7)
```



11. If I only wanted one graph per page I needed to clear all in the PLOT section of the screen (lower right hand) by clicking on the broom. I then repeated the above code WITHOUT the following code: **par(mfcol=c(1,2)).** Everything else stays the same and then your two graphs will look like this:




0.6
0.0
-0.6
CV 2, 28.2%
Vanilla
Chocolate
Jam
Artificial_frui
Dark_berry
Red_berry
Alcohol
Oak
Spicy
Dried_fruit
Astringent
Pepper
Sour
Earthy
Leather
Band.aid
Medicinal
-0.6
0.0
0.6
CV 1, 44.2

12. CVA plot with 95% confidence ellipses based on covariance matrices with thanks to Helene Hopfer, Peter Buffon and Vince Buffalo.
    a. I loaded the plotrix package and then I ran the entire code of the CVAellipses_new function written by Helene Hopfer, Peter Buffon and Vince Buffalo

```
# Ellipse function Peter Buffon & Helene Hopfer
# use equations (4) and (5) instead of (7) and (8)
# from Systematic Zoology 1985, 34:366-374

# linearModel ... MANOVA model from lm()
# factorColumn ... for which factor the CVA should be run, e.g. 'product'
# xlim, ylim ... manually input the x and y axis limits

# CV 1 vs. CV 2
Ellipses12 <- function(linearModel,factorColumn,xlim,ylim, ...) {
 # run CVA
 library(candisc)
 data.cva <- candisc(linearModel)
 xlab <- paste("CV 1 ",round(data.cva$pct[1],1),"%",sep="")
 ylab <- paste("CV 2 ",round(data.cva$pct[2],1),"%",sep="")

 # cva scores x=> divide scores for each treatment
 X <- by(data.cva$scores[,c(2:3)],factorColumn,function(x) return(x))
 Y <- combn(X,2,FUN=NULL,simplify=TRUE)
 Z <- list()
  for(i in 1:(length(Y)/2)) {
        Z[[i]] <- cov(Y[,i][[1]],Y[,i][[2]])
  }

 # determine mean for each sample score matrix
 X.m <- sapply(X, colMeans)
 X.m1 <- X.m[1,]
 X.m2 <- X.m[2,]

 # calculate eigenvector and eigenvalue of each covariance matrix
 eigen.comps <- lapply(Z, eigen)

 #eigvalues <- do.call(rbind, lapply(eigen.comps, function(x) x$values))
 eigvalues <- lapply(eigen.comps, function(x) x$values)
 eigvectors <- lapply(eigen.comps, function(x) x$vectors)

 # calculate angle for direction of major ellipse axis
 # Formula: acos(first standardized eigenvector)
 Ang <- sapply(eigvectors, function(x) acos(x[1,1])*180/pi)
```

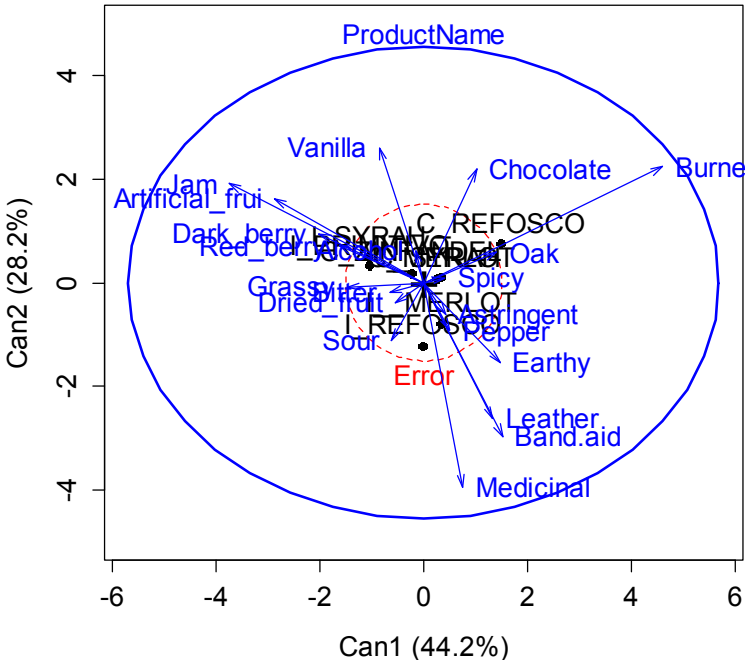
```
# calculate half of the major ellipse axis length, n observations per sample
# Formula = 2*sqrt(eigenvalue1)*sqrt([2*(n-1)/(n*(n-2))*Fcrit(2,n-2)])
 n <- length(factorColumn)/length(levels(factorColumn))
 a <- sapply(eigvalues, function(x)
      2*sqrt(abs(x[1]))*sqrt(((2*(n-1)/(n*(n-2)))*qf(0.95,2, (n-2)))))

# calculate minor ellipse axis length, n observations per sample
# Formula = 2*sqrt(eigenvalue2)*sqrt([1+1/n]*[2*(n-1)/(n-2)*Fcrit(2,n-2)])
 b <- sapply(eigvalues, function(x)
      2*sqrt(abs(x[2]))*sqrt(((2*(n-1)/(n*(n-2)))*qf(0.95,2, (n-2)))))

min(X.m1)

library(plotrix)
 par(mar=c(4,4,.2,0.2))

 plot(X.m1,X.m2, pch="+", xlim=xlim, ylim=ylim, xlab=xlab, ylab=ylab,
  col=rainbow(n=length(levels(factorColumn))), axes=FALSE, cex.lab=.9)

 axis(side = 1, at = c(min(xlim), 0, max(xlim)), line = 1, col='darkgray',
     cex.axis=.9)
 axis(side = 2, at = c(min(ylim), 0, max(ylim)), line = 1, col='darkgray',
     cex.axis=.9)
 abline(h=0,v=0, col='darkgray', lty=3, lwd=.8)

 text(X.m1,X.m2, labels=levels(factorColumn), pos=1, cex=0.8,
  col='black')

 draw.ellipse(x=X.m1, y=X.m2, a=a, b=b, angle=Ang, deg=TRUE,
        col=rainbow(n=length(levels(factorColumn))),
        lwd=.5)
}
```

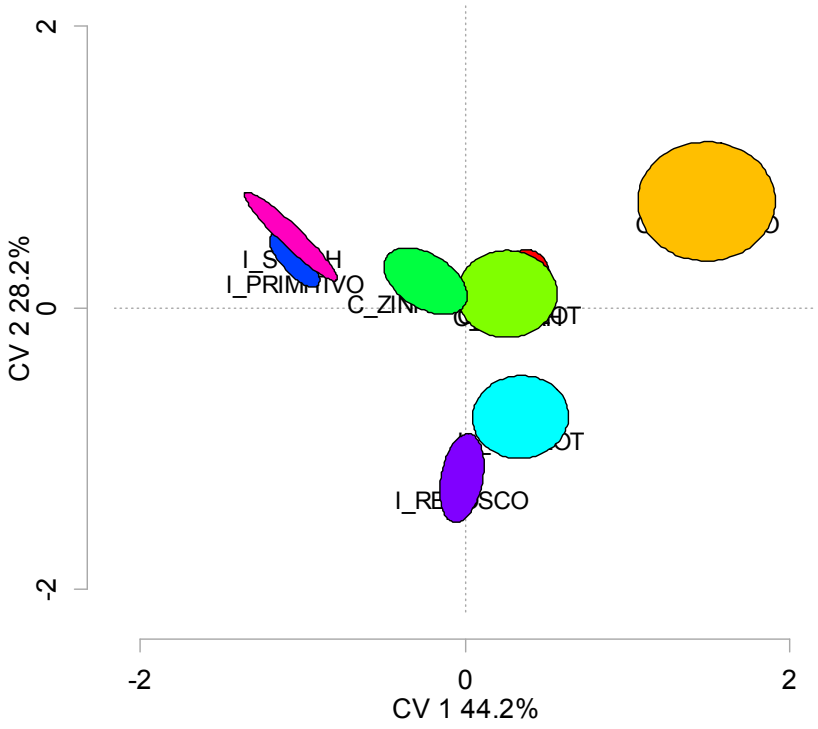
b. Then before I ran the function for dimensions 1 and 2 I went back and reran the linear models, MANOVA and CVA for the factor ProductName – mostly because it had been 10+ pages of instructions since I had done so. I also ran the heplot(da.cva) to get a sense of the axes limits for my ellipses plot.

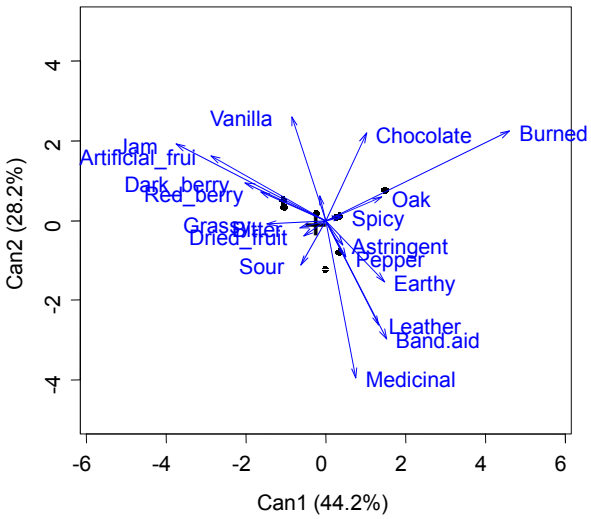
```
da.lm = lm(da.a ~ ProductName, data=torriDAFinal)
da.man = manova(da.lm)
summary(da.man, test='Wilks')
library(candisc)
da.cva = candisc(da.man)
da.cva
heplot(da.cva)
```

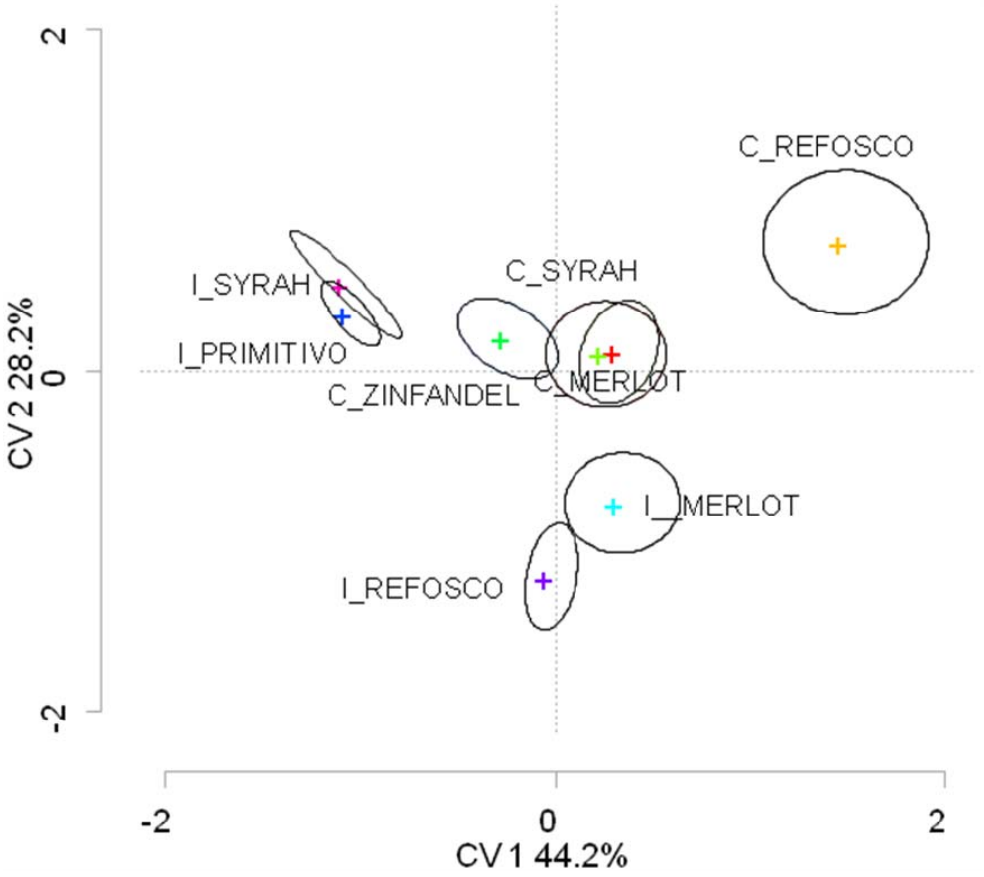
c. Based on the heplot axis limits of -2 and 2 for both x and y would work **Ellipses12(da.lm, torriDAFinal$ProductName, xlim = c(-2, 2), ylim=c(-2,2))**

The score and loadings plots are shown below. I could have changed some of the settings in the CVAellipses_new function to make the plots prettier or I could have exported and copied the plots to PowerPoint. The first two plots are exported directly from R-Studio and the second pair had some manipulation in PowerPoint.






Vanilla
Chocolate
Burned
Jam
Artificial_frui
Dark_berry
Red_berry
Oak
Spicy
Astringent
Pepper
Earthy
Leather
Band.aid
Medicinal
Sour
Can2 (28.2%)
Can1 (44.2%)


C_REFOSCO
I_SYRAH
C_SYRAH
I_PRIMITIVO
C_ZINFANDEL
C_MERLOT
I_MERLOT
I_REFOSCO
CV 2 28.2%
CV 1 44.2%

PRINCIPAL COMPONENT ANALYSIS (PCA)

To do a PCA I need the mean values for ProductName averaged across NJ and NR. To do this I need to use the mtable function. The mtable function was written by Greg Hirson.

1. First you need to run the mtable function as written

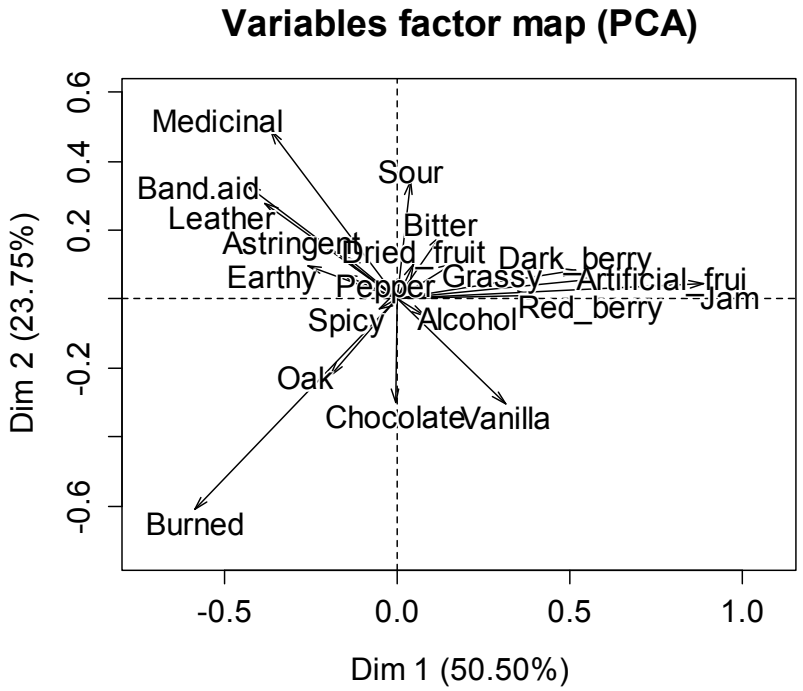
```
mtable<- function (x, bycol, firstvarcol){

#A function to compute a means table for a matrix.
#x - the data frame with the data
#bycol - the row or rows used for grouping (usually products)
#use c(col1,col2) as the bycol option if using more than one column.
#firstvarcol - the col containing the first variable

if (length(bycol)==1){
        mns<-matrix(nrow=0, ncol=length(levels(as.factor(x[,bycol]))))
        for (n in firstvarcol:length(x)){
                m.r<-with(x, tapply(x[,n], x[,bycol], mean))
                mns<-rbind(mns,m.r[])
                }
        mns<-as.data.frame(mns)
        names(mns)<-names(m.r)
        rownames(mns)<-names(x[firstvarcol:length(x)])
        mns<-t(mns)
        return(mns)
        }
        else

        bc<-paste(x[,bycol[1]],names(x)[bycol[2]],x[,bycol[2]])
        x.2<-as.data.frame(cbind(bc,x))
        mns<-matrix(nrow=0, ncol=length(levels(as.factor(x.2$bc))))

        for (n in (firstvarcol+1):length(x.2)){
                m.r<-with(x.2, tapply(x.2[,n], x.2$bc, mean))
                mns<-rbind(mns,m.r[])
                }
        mns<-as.data.frame(mns)
        names(mns)<-names(m.r)
        rownames(mns)<-names(x.2[(firstvarcol+1):length(x.2)])
        mns<-t(mns)
        titl<-paste("Means by", names(x)[bycol[1]], "and",
        names(x)[bycol[2]])
        return(mns)

        }
```

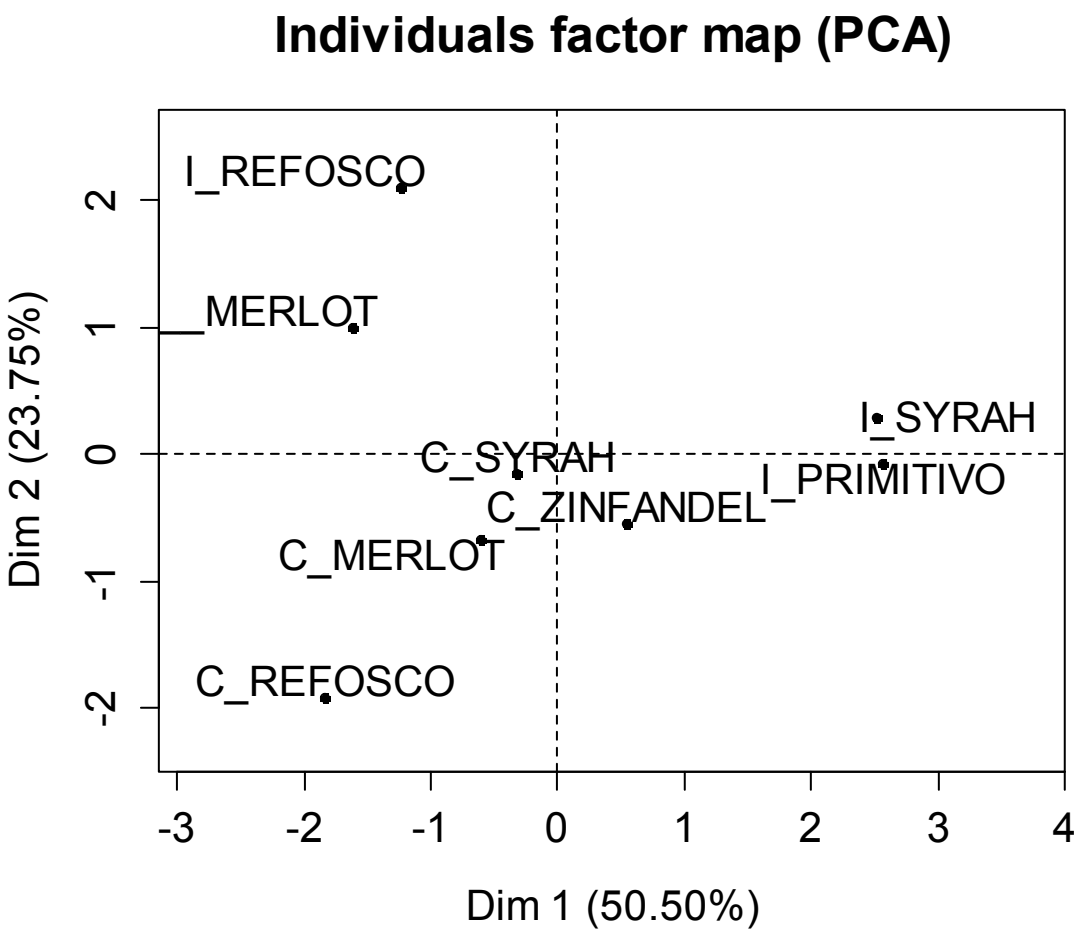
2. Then you run the function for your data set and ask for the mean values. You must specify the first column containing data for your dependent variables. In my case this was column 4
   da.means=mtable(torriDAFinal, bycol="ProductName", firstvarcol=4)
   da.means

```
> da.means=mtable(torriDAFinal, bycol="ProductName", firstvarcol=4)
> da.means
            Red_berry Dark_berry Jam Dried_fruit Artificial_frui Chocolate
C_MERLOT     2.464286  3.047619  1.3714286  1.857143  0.7761905  1.1904762
C_REFOSCO    2.466667  2.461905  1.0309524  1.423810  0.9238095  1.9976190
C_SYRAH      2.464286  2.933333  1.7452381  1.683333  0.8833333  1.4190476
C_ZINFANDEL  3.076190  3.059524  1.9785714  2.061905  0.8642857  0.9690476
I__MERLOT    2.790476  2.350000  0.8428571  1.850000  0.5738095  0.7833333
I_PRIMITIVO  3.850000  3.380952  3.6119048  1.435714  2.1904762  1.3809524
I_REFOSCO    2.478571  3.007143  1.5357143  1.873810  1.1095238  0.8095238
I_SYRAH      3.173810  4.483333  3.0976190  2.164286  2.4333333  1.1976190
I CUT OF THE REST TO SAVE SPACE
```

3. Now you can run the PCA on the mean values. First you need to load the SensoMineR package which includes the FactoMineR package that will actually run the PCA
   library(SensoMineR)

4. Next you run the PCA on the means. I asked for scale.unit=FALSE since I wanted to run a covariance matrix PCA. If I had wanted to do a correlation matrix PCA I would have asked for scale.unit=TRUE
   da.pca = PCA(da.means, scale.unit=FALSE, ncp=5, graph=TRUE)





I just copied and pasted these from the PLOT section of RStudio. I could have made them 'prettier' in PowerPoint as discussed on page 16.

5. I then used the request to see the correlations between attributes and components. Only correlations with p-values <0.20 will be printed and this could help in interpretation. **dimdesc(da.pca, axes=c(1,2))**

```
> dimdesc(da.pca, axes=c(1,2))
$Dim.1
$Dim.1$quanti
                correlation      p.value
Jam               0.9723336 5.184944e-05
Artificial_frui   0.8819989 3.752736e-03
Dark_berry        0.8492159 7.630518e-03
Red_berry         0.8418534 8.752510e-03
Band.aid         -0.7259831 4.144504e-02
Leather          -0.7504104 3.195742e-02
Earthy           -0.8635921 5.713937e-03
$Dim.2
$Dim.2$quanti
           correlation     p.value
Sour         0.8501775 0.007491159
Medicinal    0.7412352 0.035345156
Chocolate   -0.8121299 0.014329245
```

6. I then wanted to calculate the communalities of the variables for the first two dimensions of the PCA.. Communality is the sum of the squared loadings for the number of dimensions that you would like to keep. [If the number of observations is larger than the number of variables then the communality would be 100 for each variable if you kept all PCs). In this case we have many more variables than observations so the communality for the 7 PCs that we can calculate does NOT sum to 100].
   I typed the following
   list(da.pca)

```
> list(da.pca)
[[1]]
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 8 individuals, described by 7 variables
*The results are available in the following objects:

   name               description
1  "$eig"             "eigenvalues"
2  "$var"             "results for the variables"
3  "$var$coord"       "coord. for the variables"
4  "$var$cor"         "correlations variables - dimensions"
5  "$var$cos2"        "cos2 for the variables"
6  "$var$contrib"     "contributions of the variables"
7  "$ind"             "results for the individuals"
8  "$ind$coord"       "coord. for the individuals"
9  "$ind$cos2"        "cos2 for the individuals"
10 "$ind$contrib"     "contributions of the individuals"
11 "$call"            "summary statistics"
12 "$call$centre"     "mean of the variables"
13 "$call$ecart.type" "standard error of the variables"
14 "$call$row.w"      "weights for the individuals"
15 "$call$col.w"      "weights for the variables"
```

7. I wanted to use the contributions – which are the squared loadings for each variable for each dimension but to make the process simpler I renamed the variable and printed it.
   contrib=(da.pcavarcontrib)
   print(contrib)

```
> print(da.pca$var$contrib)
               Dim.1       Dim.2        Dim.3        Dim.4        Dim.5
Red_berry   5.812495e+00  0.01183650  2.766625104 2.100816e+00  7.858052268
Dark_berry  1.019445e+01  0.54747656  7.479649615 1.071446e+01  3.007620528
Jam         2.973572e+01  0.15743214  1.023885421 9.131359e+00  2.021857949
Dried_fruit 9.497174e-02  0.80254114  0.072847141 8.539416e+00  5.856804510
Art_frui    1.233758e+01  0.26493541 10.718856979 4.654448e-03 9.004500406
Chocolate   4.548927e-04  7.22238107  5.454104517 2.716533e+00  1.960587302
Vanilla     3.725204e+00  7.37761508  0.062435270 7.751445e+00  0.492798838
Oak         1.408041e+00  3.92877247  0.956828308 2.326615e-02 10.035753473
Burned      1.294826e+01 29.82893271 21.463723348 6.212810e-05  5.836562533
Leather     5.521471e+00  6.23093561  2.021816156 2.668093e+00  0.161397949
Earthy      2.499701e+00  0.77817259  1.186326059 6.488936e-02  0.514722497
Spicy       9.111685e-02  0.06515814  0.183549138 7.950668e-01  0.791951168
Pepper      8.900846e-01  0.41778874  3.361449514 1.306723e+01  4.873216096
Grassy      9.946277e-01  0.84523668  0.001188436 3.183185e+00  3.766804104
Medicinal   5.020278e+00 19.10790265  0.181157797 1.809724e+01  0.004065457
Band.aid    7.031442e+00  8.79845894  3.161687179 1.865489e+00  9.762658260
Sour        6.119743e-02  9.45311141  6.522316456 1.317608e-02  1.328921754
Bitter      6.024608e-01  2.73147842 15.266505793 6.556316e+00  0.276521541
```

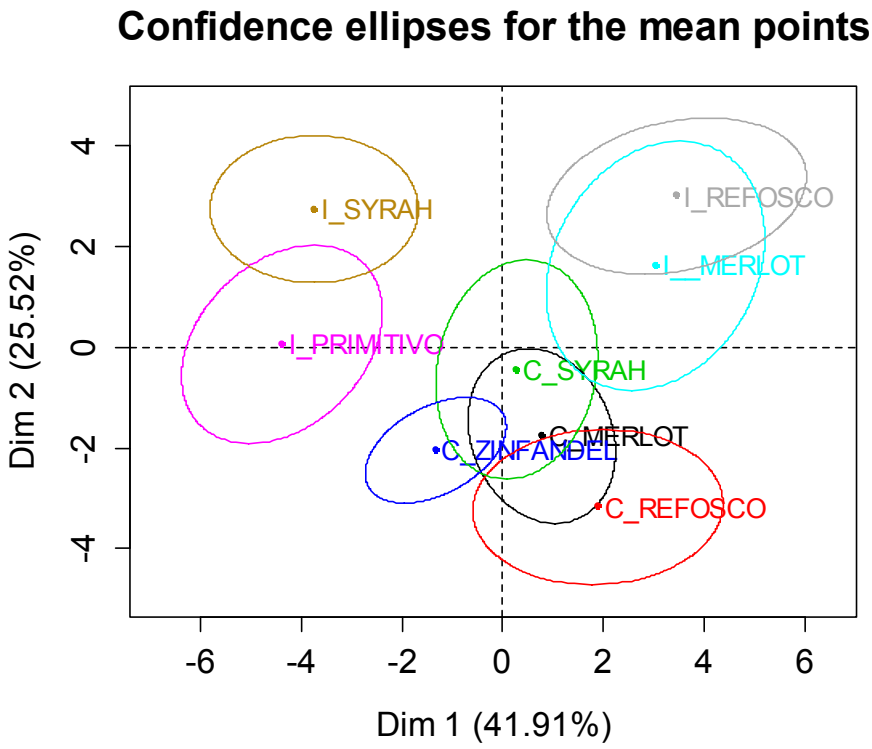
```
Alcohol       2.590159e-01  0.18957369  9.205952495 5.954490e+00 32.429453147
Astringent    7.714358e-01  1.24026006  8.909095273 6.752807e+00  0.015750219
                     Dim.6      Dim.7
Red_berry         2.81675712 23.9220293
Dark_berry        1.86737832 20.3839090
Jam               0.40542945  2.9276599
Dried_fruit       2.39003773  0.6746280
Artificial_frui   2.71752129  2.2854343
Chocolate         1.49159264  1.1160252
Vanilla           5.47492682  3.6166504
Oak              32.19904591  0.2794998
Burned            0.05553048  0.4479311
Leather          12.66314532  0.1292195
Earthy            3.45757276  0.7217016
Spicy             7.79068554  8.8675430
Pepper            0.07470830  3.1112380
Grassy            3.52218567  1.8117753
Medicinal         0.49998147  3.6759718
Band.aid          3.00258267  0.3152018
Sour              3.56526982  0.3522765
Bitter            5.01001826 20.1584391
Alcohol           8.05756261  2.9319205
Astringent        2.93806780  2.2709460
```

8. Then I picked the dimensions I wanted to keep (PC1 and PC2) and summed across them to get the communality.
   contrib=(da.pcavarcontrib [,1:2])
   print(contrib)
   comm=rowSums(contrib)
   print(comm)

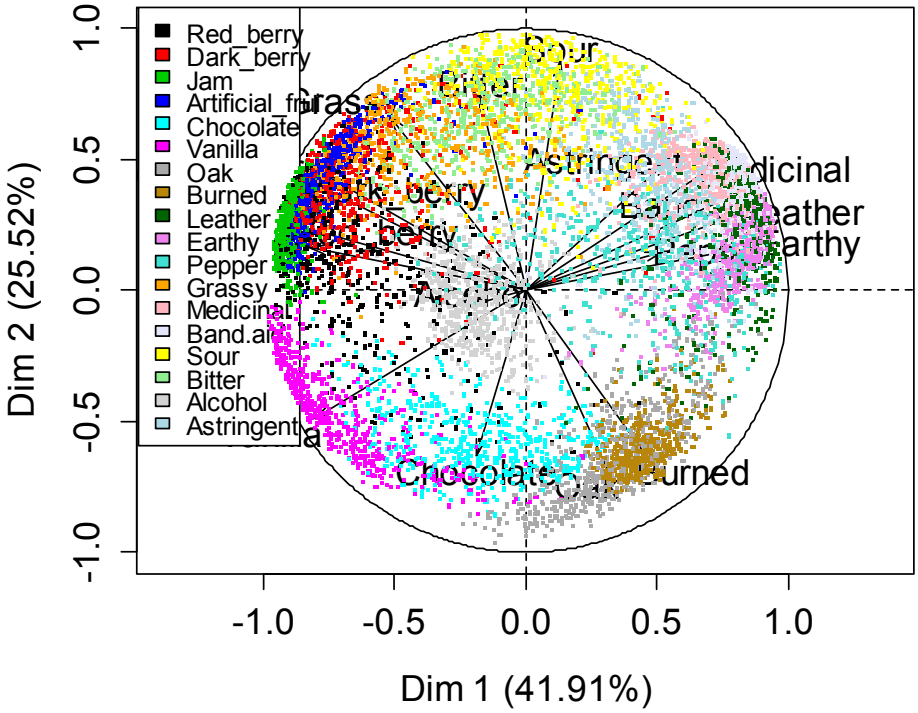
```
> contrib=(da.pca$var$contrib [,1:2])
> print(contrib)
                      Dim.1       Dim.2
Red_berry        5.812495e+00  0.01183650
Dark_berry       1.019445e+01  0.54747656
Jam              2.973572e+01  0.15743214
Dried_fruit      9.497174e-02  0.80254114
Artificial_frui  1.233758e+01  0.26493541
Chocolate        4.548927e-04  7.22238107
Vanilla          3.725204e+00  7.37761508
Oak              1.408041e+00  3.92877247
Burned           1.294826e+01 29.82893271
Leather          5.521471e+00  6.23093561
I CUT THESE TO SAVE SPACE
Alcohol          2.590159e-01  0.18957369
Astringent       7.714358e-01  1.24026006
> comm=rowSums(contrib)
> print(comm)
   Red_berry       Dark_berry             Jam     Dried_fruit Artificial_frui
    5.8243317       10.7419267       29.8931479        0.8975129       12.6025166
   Chocolate          Vanilla              Oak          Burned         Leather
    7.2228360       11.1028186        5.3368139       42.7771880       11.7524067
      Earthy            Spicy             Pepper           Grassy       Medicinal
    3.2778731        0.1562750        1.3078734        1.8398644       24.1281808
     Band.aid             Sour            Bitter          Alcohol      Astringent
   15.8299008        9.5143088        3.3339392        0.4485896        2.0116959
```

9. SensoMineR also allows one to do a PCA with 95% confidence ellipses using the raw data. To do this I typed the following. In this case col.p=2 indicates the product column and col.j=1 indicates the judges column. I also indicated that the attributes started in column 4.
da.panelellipses=panellipse(torriDAFinal, col.p=2, col.j=1, firstvar=4)

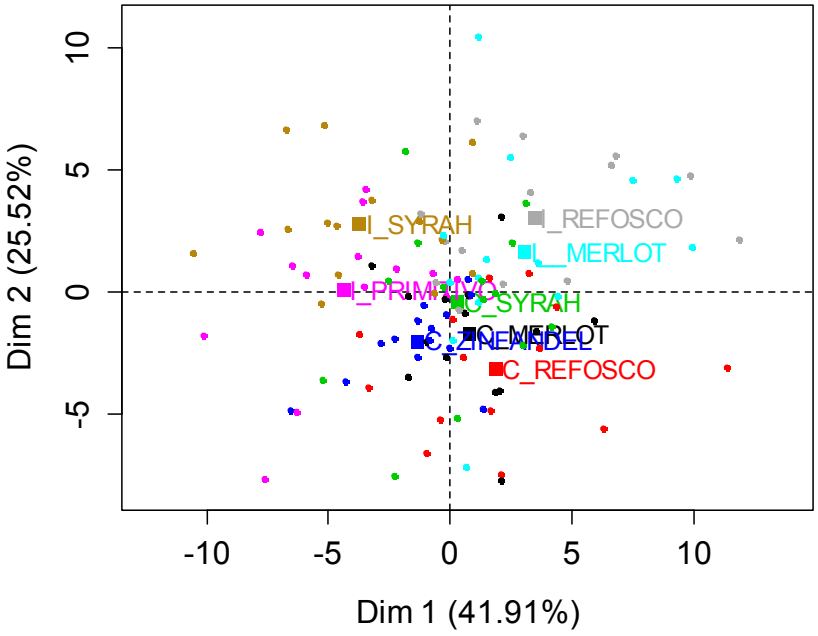
You will get 5 plots in the output. I did not change the defaults for this analysis and thus this PCA is correlation based.

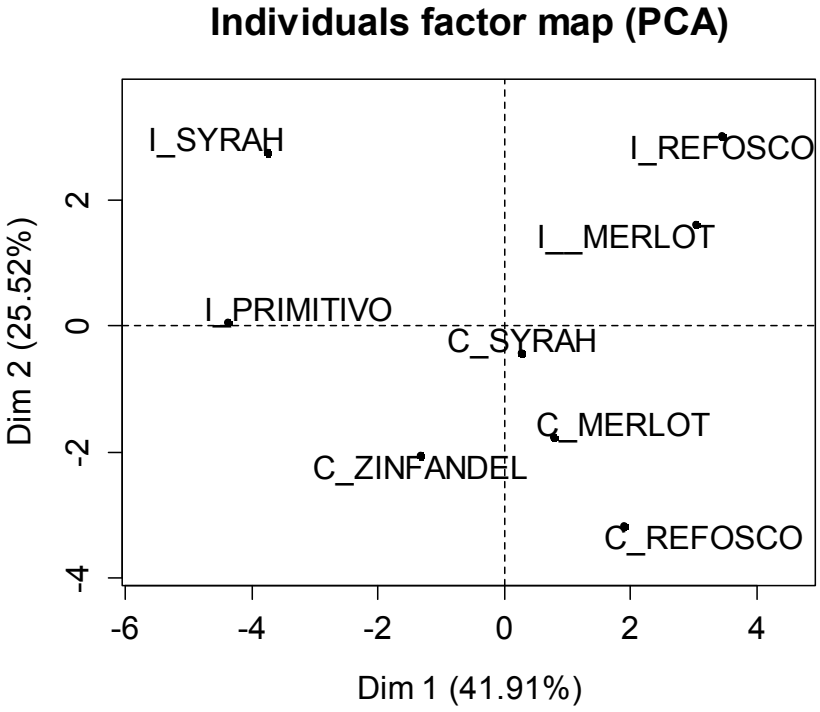


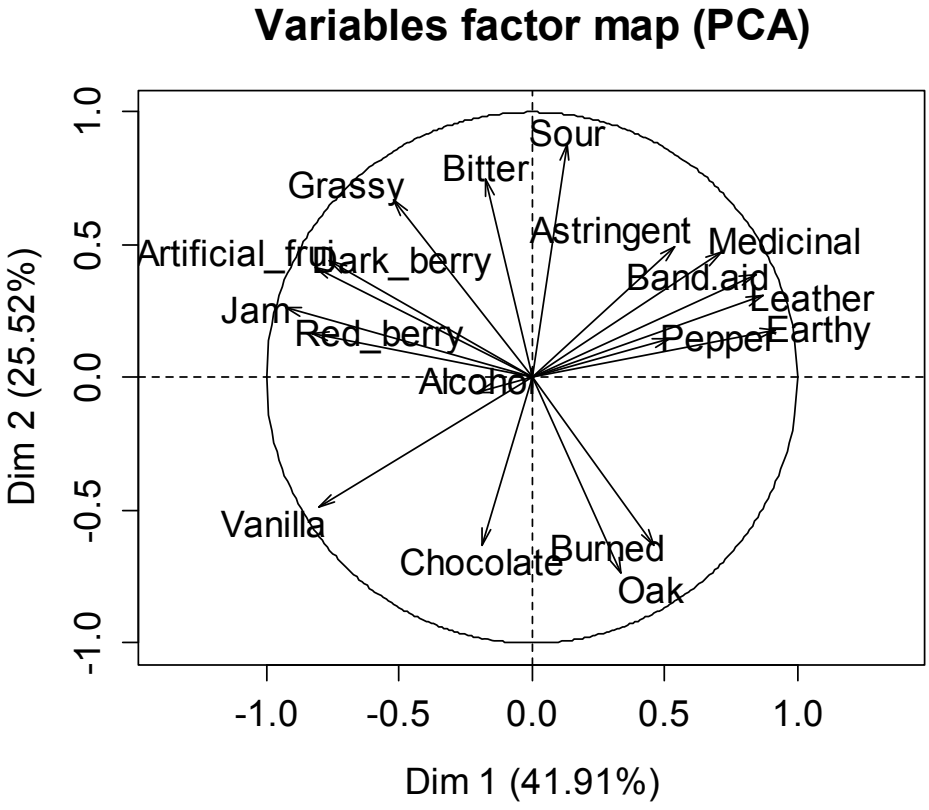
Variables factor map (PCA)



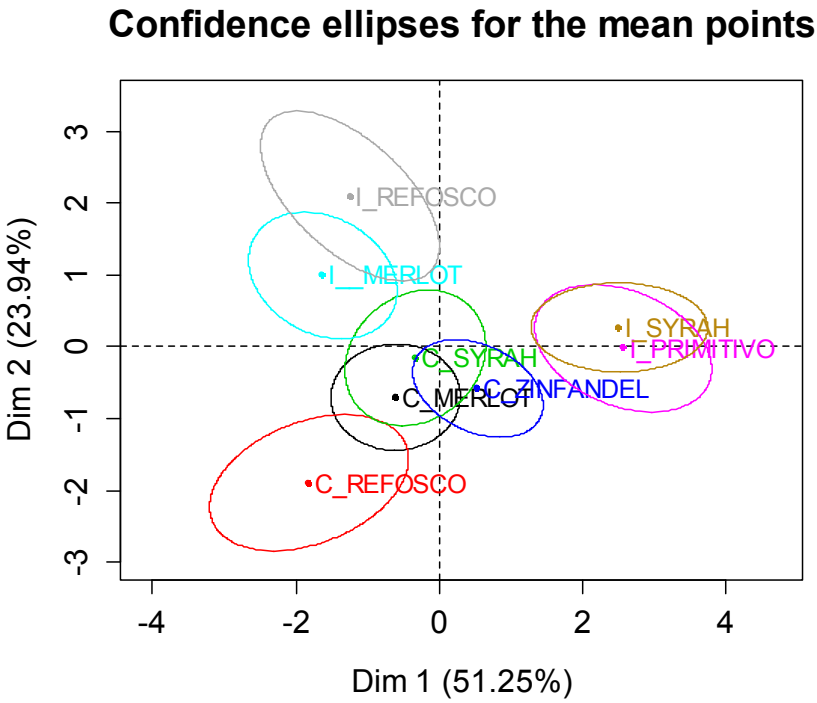
Individual description

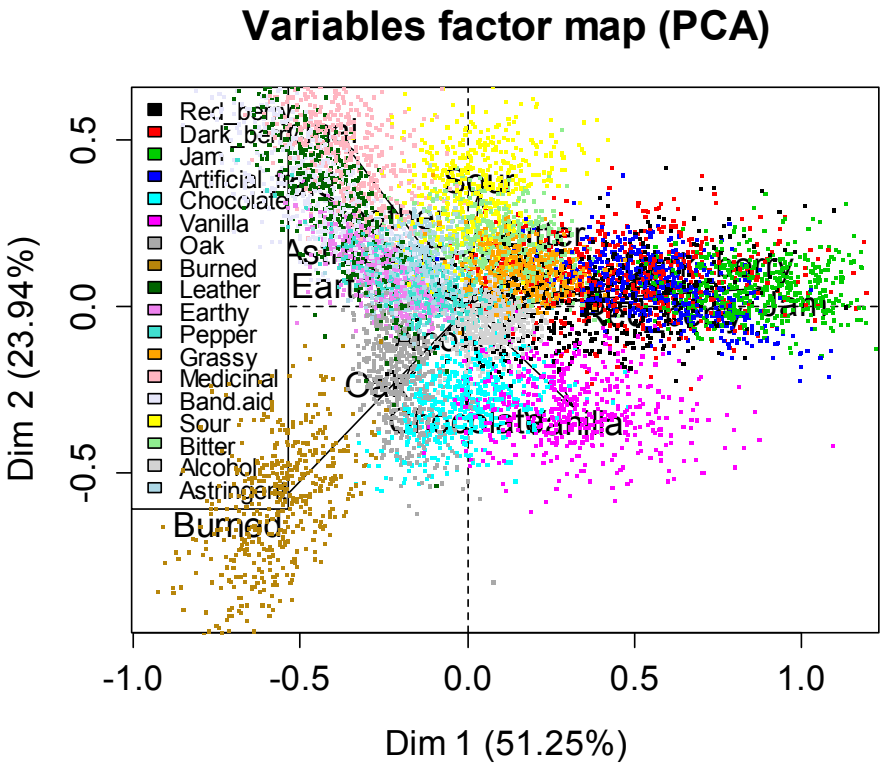



Individuals factor map (PCA)
I_SYRAH
I_REFOSCO
I__MERLOT
I_PRIMITIVO
C_SYRAH
C_MERLOT
C_ZINFANDEL
C_REFOSCO
Dim 2 (25.52%)
Dim 1 (41.91%)
2
0
-2
-4
-6
-4
-2
0
2
4

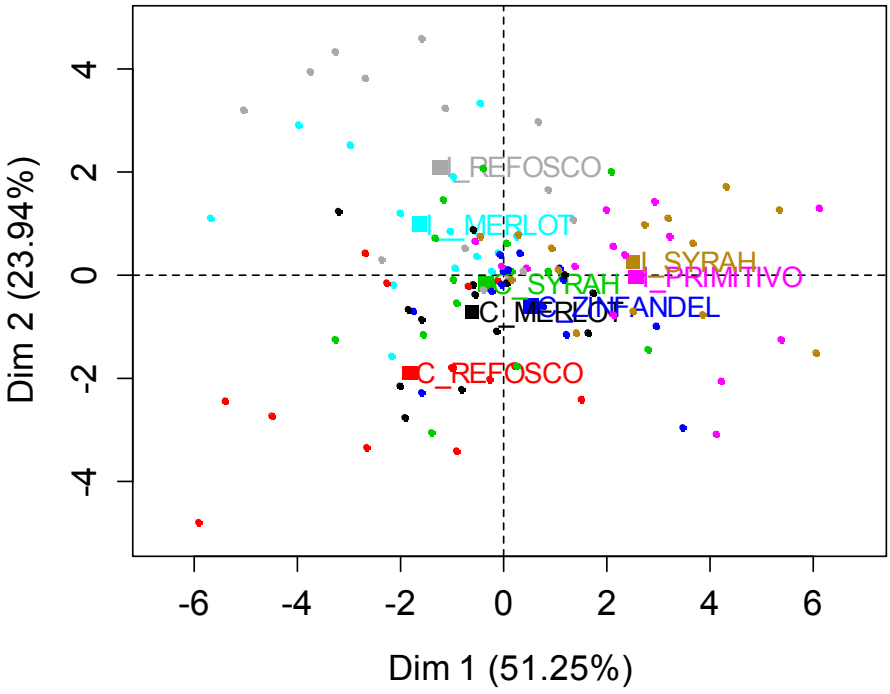

Variables factor map (PCA)
Sour
Bitter
Grassy
Astringent
Medicinal
Artificial_fruit
Dark_berry
Band.aid
Leather
Jam
Red_berry
Pepper
Earthy
Alcohol
Vanilla
Chocolate
Burned
Oak
Dim 2 (25.52%)
Dim 1 (41.91%)
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0

10. I reran the panellipse function with scale.unit=FALSE and the following resulted

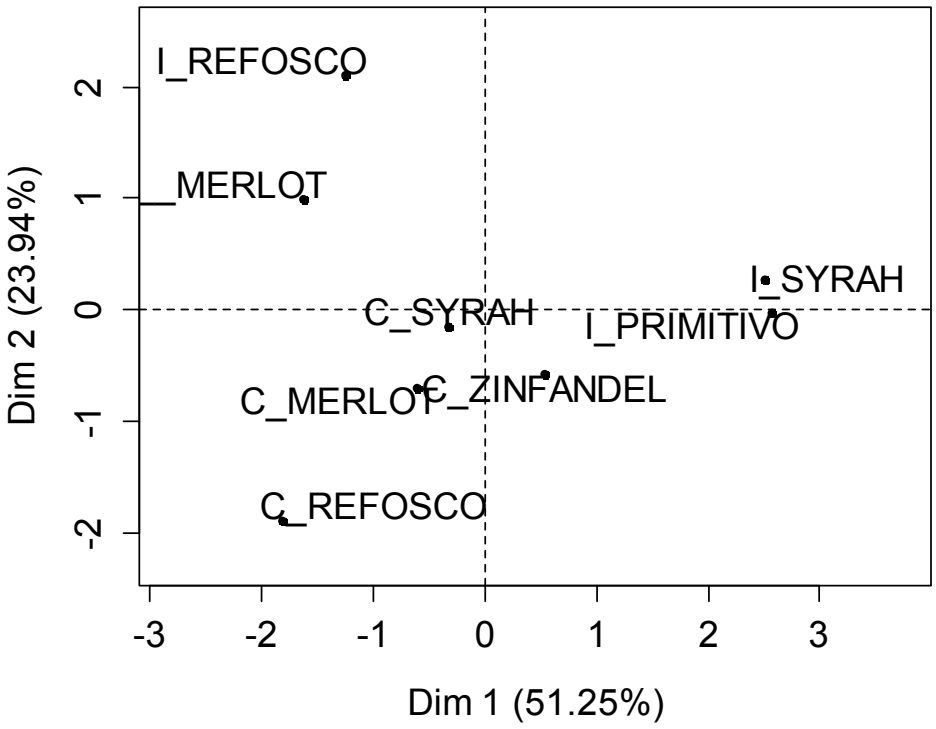


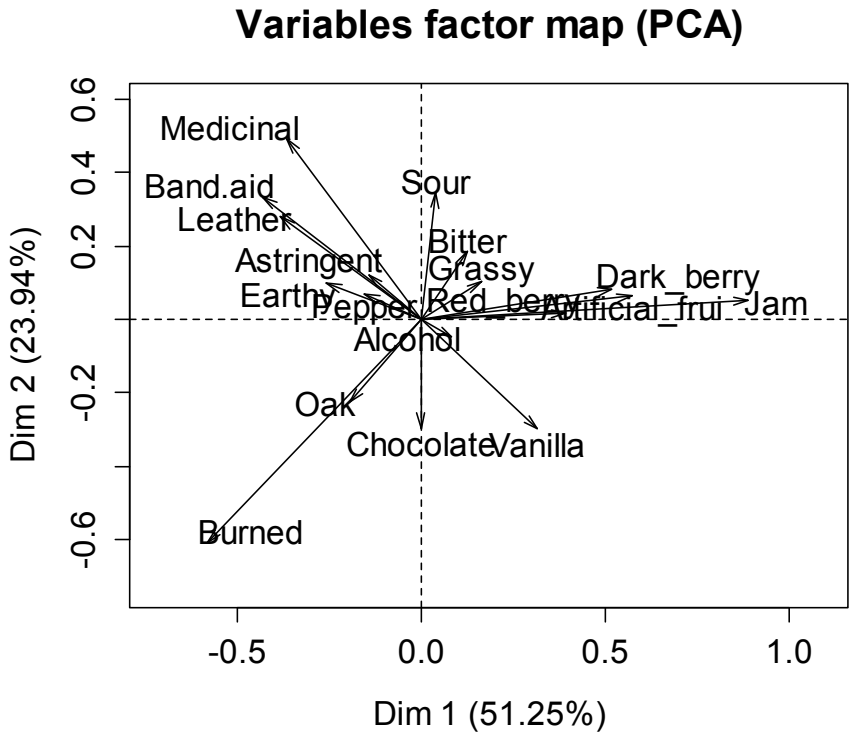


Individual description



Individuals factor map (PCA)





11. I wanted to look at some of the output from panellipse. To see what was available I typed **summary(da.panelellipses)**

```
> summary(da.panelellipses)
             Length Class      Mode
eig           3     data.frame list
coordinates   1     -none-     list
hotelling    64     -none-     numeric
correl        3     -none-     list
```

12. I decided to create a table of the Hotelling T values. This a matrix with the P-values of the Hotelling's T2 tests for each pair of products: this matrix allows one to find the products which are significantly different from each other for the 2-components sensory description (sort of an lsd for the PCA)
coltable(da.panelellipses$hotelling, main.title = "P-values for the Hotelling's T2 tests")

P-values for the Hotelling's T2 tests

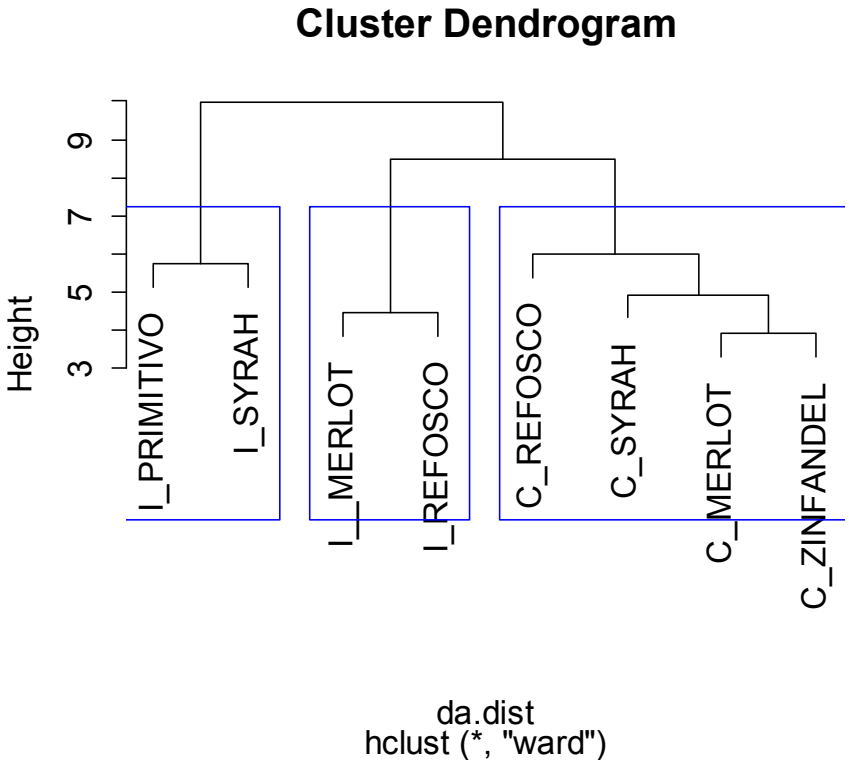
	C_MERLOT	C_REFOSCO	C_SYRAH	C_ZINFANDEL	I__MERLOT	I_PRIMITIVO	I_REFOSCO	I_SYRAH
C_MERLOT	1	0.05215	0.5451	0.1273	0.003156	8.821e-05	0.0002263	0.000104
C_REFOSCO	0.05215	1	0.01246	0.002463	7.147e-05	7.915e-06	5.228e-06	6.472e-06
C_SYRAH	0.5451	0.01246	1	0.2356	0.02246	0.001006	0.005852	0.001447
C_ZINFANDEL	0.1273	0.002463	0.2356	1	0.0006153	0.003376	0.0002946	0.002035
I__MERLOT	0.003156	7.147e-05	0.02246	0.0006153	1	1.375e-05	0.08825	1.923e-05
I_PRIMITIVO	8.821e-05	7.915e-06	0.001006	0.003376	1.375e-05	1	8.722e-05	0.8095
I_REFOSCO	0.0002263	5.228e-06	0.005852	0.0002946	0.08825	8.722e-05	1	0.0001061
I_SYRAH	0.000104	6.472e-06	0.001447	0.002035	1.923e-05	0.8095	0.0001061	1

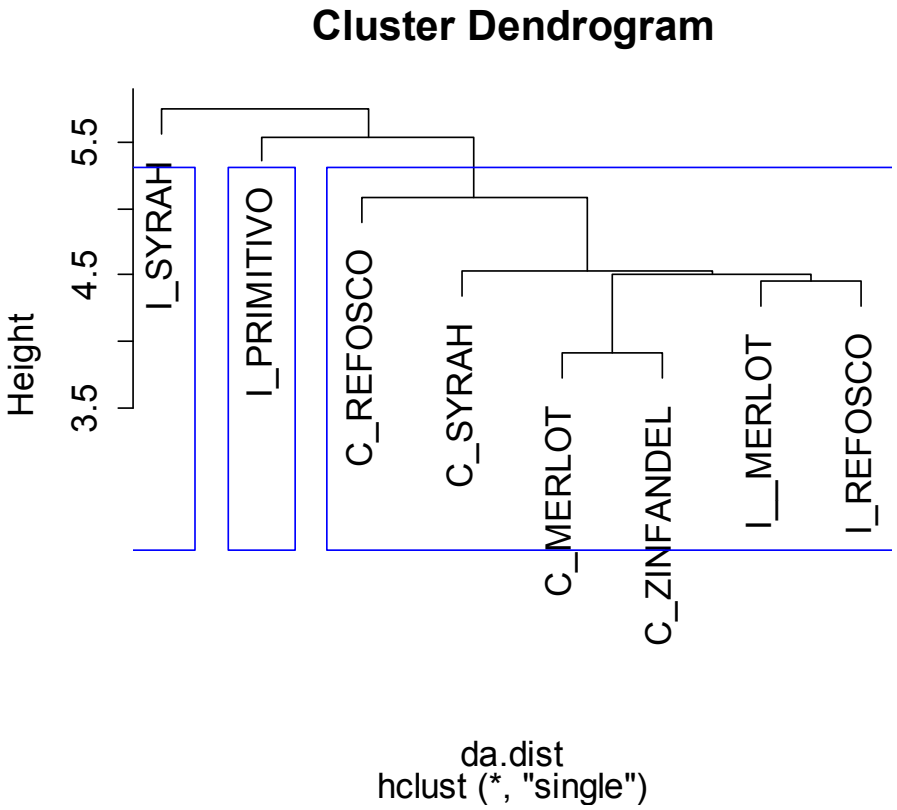
CLUSTER ANALYSES

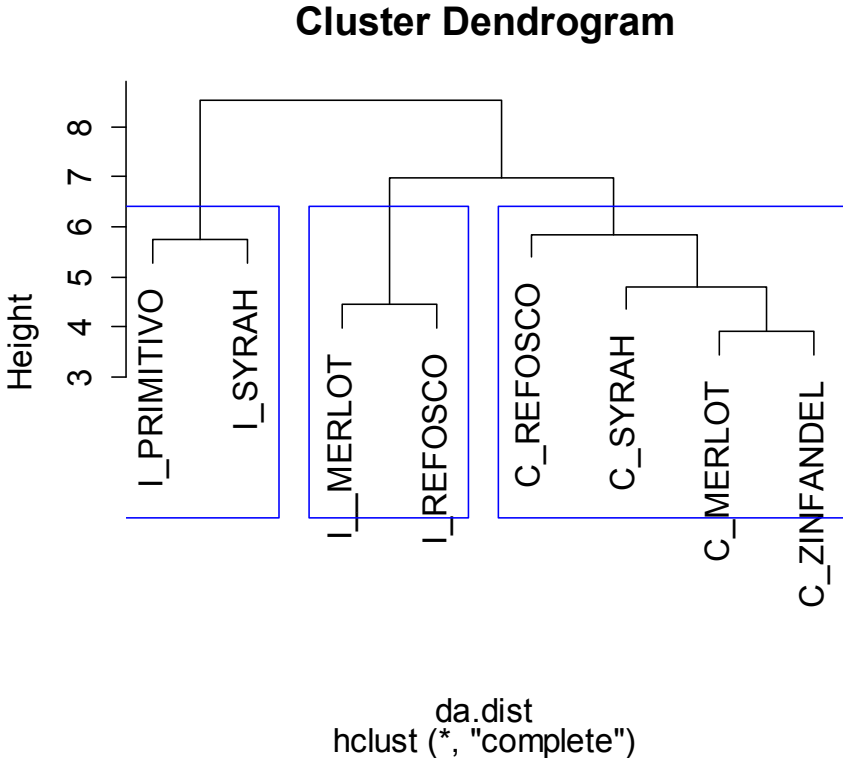
1. To do clustering I need to scale the data
   da.meansscaled <- scale(da.means)

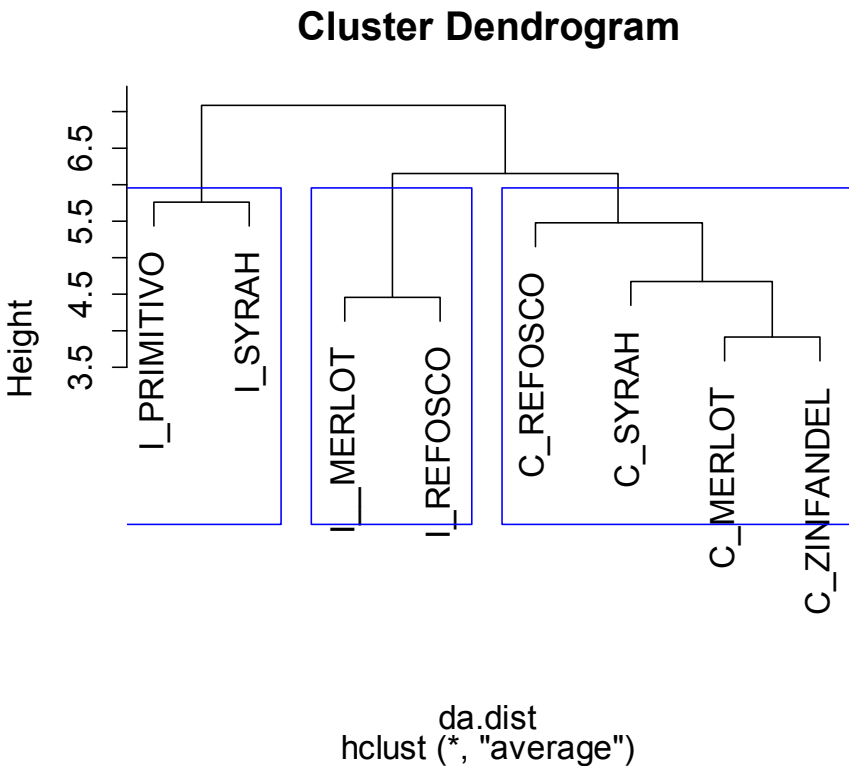
2. Then I created a distance matrix. In this example I chose to use Euclidean distances but I could also have used others such as maximum, manhattan, binary and minkowski.
   da.dist <- dist(da.meansscaled, method = "euclidean")

3. Then I ran four different hierarchical clustering methods: Wards, Single Linkage, Complete Linkage and Average Linkage. I did some fancy stuff to the dendograms.
   da.ward <- hclust(da.dist, method="ward")
   plot(da.ward)
   groups <- cutree(da.ward, k=3)
   rect.hclust(da.ward, k=3, border="blue")
   da.single <- hclust(da.dist, method="single")
   plot(da.single)
   groups <- cutree(da.single, k=3)
   rect.hclust(da.single, k=3, border="blue")
   da.complete <- hclust(da.dist, method="complete")
   plot(da.complete)
   groups <- cutree(da.complete, k=3)
   rect.hclust(da.complete, k=3, border="blue")
   da.aver <- hclust(da.dist, method="aver")
   plot(da.aver)
   groups <- cutree(da.aver, k=3)
   rect.hclust(da.aver, k=3, border="blue")


Cluster Dendrogram
Height
3
5
7
9
I_PRIMITIVO
I_SYRAH
I__MERLOT
I_REFOSCO
C_REFOSCO
C_SYRAH
C_MERLOT
C_ZINFANDEL
da.dist
hclust (*, "ward")


Cluster Dendrogram
Height
3.5
4.5
5.5
I_SYRAH
I_PRIMITIVO
C_REFOSCO
C_SYRAH
C_MERLOT
C_ZINFANDEL
I__MERLOT
I_REFOSCO
da.dist
hclust (*, "single")


Cluster Dendrogram
Height
3
4
5
6
7
8
I_PRIMITIVO
I_SYRAH
I_MERLOT
I_REFOSCO
C_REFOSCO
C_SYRAH
C_MERLOT
C_ZINFANDEL
da.dist
hclust (*, "complete")


Cluster Dendrogram
Height
3.5
4.5
5.5
6.5
I_PRIMITIVO
I_SYRAH
I_MERLOT
I_REFOSCO
C_REFOSCO
C_SYRAH
C_MERLOT
C_ZINFANDEL
da.dist
hclust (*, "average")

4. I was interested to see if the clusters differed significantly in the raw data set. To do that I had to add cluster membership to the raw data file. I am showing the Ward's example. In the cutree statement the output went to 'groups', so I typed
   list(groups)

```
> list(groups)
[[1]]
C_MERLOT  C_REFOSCO  C_SYRAH  C_ZINFANDEL  I__MERLOT  I_PRIMITIVO
       1          1        1            1          2            3
  I_REFOSCO    I_SYRAH
          2          3
```

5. I created a data frame of the raw data and the group membership and then I checked that it had worked.
   da.wclus=data.frame(torriDAFinal, groups)
   head(da.wclus)

```
head(da.wclus)
    NJ ProductName NR Red_berry Dark_berry Jam Dried_fruit Artificial_frui
1 1331    C_MERLOT  7       5.1        5.8 2.1         4.7             1.0
2 1331     C_SYRAH  7       5.6        1.9 3.9         1.2             7.9
3 1331 C_ZINFANDEL  7       4.9        2.6 1.4         5.9             0.8
4 1331   C_REFOSCO  7       5.0        1.9 7.8         0.6             6.6
5 1331   I__MERLOT  7       3.3        7.2 0.5         5.8             0.7
6 1331     I_SYRAH  7       5.7        3.6 8.7         1.9             7.4
  Chocolate Vanilla Oak Burned Leather Earthy Spicy Pepper Grassy Medicinal
1       2.9     5.0 5.0    1.4     2.3    0.6   3.2    5.4    2.1       0.4
2       1.0     8.3 2.3    1.8     3.5    1.0   0.7    3.0    0.6       2.2
3       2.0     2.7 5.6    1.9     4.3    0.6   1.4    4.1    3.6       1.7
4       6.4     5.5 3.6    3.2     0.3    0.2   2.9    0.9    1.8       0.2
5       2.1     1.3 2.1    5.6     6.5    4.7   0.7    2.8    3.8       2.6
6       3.3     6.9 1.5    0.2     1.5    0.3   3.1    1.6    0.9       0.5
  Band.aid Sour Bitter Alcohol Astringent groups
1      0.4  5.0    5.9     9.0        8.7      1
2      0.4  9.7    5.2     7.2        8.3      1
3      0.1  7.8    3.5     4.7        5.0      1
4      0.2  8.3    3.0     8.9        7.8      1
5      5.1  7.6    1.9     2.8        5.9      2
6      1.2  7.2    9.8     8.7        8.0      3
```

6. Then I did a one-way ANOVA on the raw data by first making the group variable a factor (and changing its name to something easier to type), and then creating the attribute matrix and checking that.
   cluster=as.factor(da.wclus$groups)
   da.a=as.matrix(da.wclus [,-c(1:3,24)])
   head(da.a)

```
cluster=as.factor(da.wclus$groups)
> da.a=as.matrix(da.wclus [,-c(1:3,24)])
> head(da.a)
   Red_berry Dark_berry Jam Dried_fruit Artificial_frui Chocolate Vanilla Oak
[1,]       5.1        5.8 2.1         4.7             1.0       2.9     5.0 5.0
[2,]       5.6        1.9 3.9         1.2             7.9       1.0     8.3 2.3
[3,]       4.9        2.6 1.4         5.9             0.8       2.0     2.7 5.6
[4,]       5.0        1.9 7.8         0.6             6.6       6.4     5.5 3.6
[5,]       3.3        7.2 0.5         5.8             0.7       2.1     1.3 2.1
[6,]       5.7        3.6 8.7         1.9             7.4       3.3     6.9 1.5
```

```
   Burned Leather Earthy Spicy Pepper Grassy Medicinal Band.aid Sour Bitter
[1,]    1.4     2.3    0.6   3.2    5.4    2.1       0.4       0.4  5.0    5.9
[2,]    1.8     3.5    1.0   0.7    3.0    0.6       2.2       0.4  9.7    5.2
[3,]    1.9     4.3    0.6   1.4    4.1    3.6       1.7       0.1  7.8    3.5
[4,]    3.2     0.3    0.2   2.9    0.9    1.8       0.2       0.2  8.3    3.0
[5,]    5.6     6.5    4.7   0.7    2.8    3.8       2.6       5.1  7.6    1.9
[6,]    0.2     1.5    0.3   3.1    1.6    0.9       0.5       1.2  7.2    9.8
     Alcohol Astringent
[1,]     9.0        8.7
[2,]     7.2        8.3
[3,]     4.7        5.0
[4,]     8.9        7.8
[5,]     2.8        5.9
[6,]     8.7        8.0
```

7. I then ran a one-way ANOVA with cluster as the factor.
   da.lm=lm(da.a~cluster, data=da.wclus)
   da.aov=aov(da.lm)
   summary(da.aov)

```
> da.lm=lm(da.a~cluster, data=da.wclus)
> da.aov=aov(da.lm)
> summary(da.aov)

I CUT OUT ALL THE NON-SIGNIFICANT VARIABLES

 Response Dark_berry :
             Df  Sum Sq Mean Sq F value  Pr(>F)
cluster       2   48.02 24.0108  3.2194 0.04123 *
Residuals   333 2483.59  7.4582
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Jam :
             Df  Sum Sq Mean Sq F value  Pr(>F)
cluster       2   46.39 23.1962  4.0611 0.01809 *
Residuals   333 1902.03  5.7118
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Artificial_frui :
             Df  Sum Sq Mean Sq F value   Pr(>F)
cluster       2   49.29 24.6467  6.4404 0.001802 **
Residuals   333 1274.35  3.8269
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Oak :
             Df  Sum Sq Mean Sq F value Pr(>F)
cluster       2   35.08  17.542  4.5623 0.0111 *
Residuals   333 1280.37   3.845
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Burned :
             Df  Sum Sq Mean Sq F value    Pr(>F)
cluster       2   97.94  48.970  12.571 5.457e-06 ***
Residuals   333 1297.24   3.896
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
 Response Grassy :
            Df Sum Sq Mean Sq F value  Pr(>F)
cluster      2  13.38  6.6921  3.2702 0.03922 *
Residuals   333 681.45  2.0464
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Sour :
            Df  Sum Sq Mean Sq F value Pr(>F)
cluster      2   44.84 22.4222  2.5603 0.0788 .
Residuals   333 2916.26  8.7575
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

8. Next I wanted to do K-means clustering. To do a K-means clustering I have to specify the number of clusters. In this case I specified 3 (based on what I had seen in the Ward's).
   fit <- kmeans(da.meansscaled, 3)

9. Then I calculated the cluster means and added the cluster assignment to my original data file.
   aggregate(da.meansscaled, by=list(fit$cluster), FUN=mean)
   da.kmeans <- data.frame(da.meansscaled, fit$cluster)

```
> fit <- kmeans(da.meansscaled, 3)
> aggregate(da.meansscaled,by=list(fit$cluster),FUN=mean
> da.kmeans<-data.frame(da.meansscaled, fit$cluster)
  Group.1  Red_berry Dark_berry        Jam Dried_fruit Artificial_frui
1       1  1.3358512  1.2851815  1.4873826  0.02338458       1.5750207
2       2 -0.4230096 -0.6289573 -0.7293720  0.25500328      -0.5444569
3       3 -0.4564208 -0.3281121 -0.3790053 -0.13919393      -0.5152819
I CUT THE REST
```

10. I then asked to see the mean data with the appended cluster assignments
    da.kmeans

```
> da.kmeans
I CUT EVERYTING EXCEPT THE LAST FEW ATTRIBUTES
                     Sour     Bitter     Alcohol Astringent fit.cluster
C_MERLOT    -1.0699557289 -1.1456269 -0.84814288 -0.5356434           3
C_REFOSCO   -0.9542103571 -0.4725206 -0.03433262  0.6244981           3
C_SYRAH     -0.0006889605  0.4698282  1.82217204 -0.3344628           3
C_ZINFANDEL -0.4195769729 -0.8386904  0.46921248 -0.8374143           3
I__MERLOT   -0.0282473824  0.1521220 -1.48393215  0.5641440           2
I_PRIMITIVO -0.4030419198 -0.5856025 -0.53787772 -1.6689608           1
I_REFOSCO    1.9228888857  0.4213645  0.13351575  1.1743918           2
I_SYRAH      0.9528324360  1.9991257  0.47938511  1.0134473           1
```

11. Since I wanted to use the raw data with the cluster information to do an ANOVA on clusters I then appended the cluster groupings to the RAW data and checked that it had worked
    da.kclus=data.frame(torriDAFinal, fit$cluster)
    head(da.kclus)

```
> head(da.kclus)
    NJ ProductName NR Red_berry Dark_berry Jam Dried_fruit Artificial_frui
1 1331    C_MERLOT  7       5.1        5.8 2.1         4.7             1.0
2 1331     C_SYRAH  7       5.6        1.9 3.9         1.2             7.9
3 1331 C_ZINFANDEL  7       4.9        2.6 1.4         5.9             0.8
4 1331   C_REFOSCO  7       5.0        1.9 7.8         0.6             6.6
5 1331   I__MERLOT  7       3.3        7.2 0.5         5.8             0.7
6 1331     I_SYRAH  7       5.7        3.6 8.7         1.9             7.4
I CUT SOME ATTRIBUTES HERE
  Band.aid Sour Bitter Alcohol Astringent fit.cluster
1      0.4  5.0    5.9     9.0        8.7           2
2      0.4  9.7    5.2     7.2        8.3           2
3      0.1  7.8    3.5     4.7        5.0           2
4      0.2  8.3    3.0     8.9        7.8           3
5      5.1  7.6    1.9     2.8        5.9           2
6      1.2  7.2    9.8     8.7        8.0           3
```

12. Then I changed the cluster assignment into a factor (and I changed the name to something easier to type)
    cluster=as.factor(da.kclus$fit.cluster)

13. Then I created the attribute matrix by removing NJ, ProductName, NR and cluster and I checked it. Please note the MINUS sign before the c(…).
    da.a=as.matrix(da.kclus [,-c(1:3,24)])
    head(da.a)

```
> cluster=as.factor(da.kclus$fit.cluster)
> da.a=as.matrix(da.kclus [,-c(1:3,24)])
> head(da.a)
 Red_berry Dark_berry Jam Dried_fruit Artificial_frui Chocolate Vanilla Oak
[1,]   5.1        5.8 2.1         4.7             1.0       2.9     5.0 5.0
[2,]   5.6        1.9 3.9         1.2             7.9       1.0     8.3 2.3
[3,]   4.9        2.6 1.4         5.9             0.8       2.0     2.7 5.6
[4,]   5.0        1.9 7.8         0.6             6.6       6.4     5.5 3.6
[5,]   3.3        7.2 0.5         5.8             0.7       2.1     1.3 2.1
[6,]   5.7        3.6 8.7         1.9             7.4       3.3     6.9 1.5
I CUT SOME ATTRIBUTES HERE
     Alcohol Astringent
[1,]     9.0        8.7
[2,]     7.2        8.3
[3,]     4.7        5.0
[4,]     8.9        7.8
[5,]     2.8        5.9
[6,]     8.7        8.0
```

14. Then I ran the ANOVA and printed it out
    da.lm=lm(da.a~cluster, data=da.kclus)
    da.aov=aov(da.lm)
    summary(da.aov)

```
> da.lm=lm(da.a~cluster, data=da.kclus)
> da.aov=aov(da.lm)
> summary(da.aov)
```

```
I CUT ALL THE NON-SIGNIFCANT ATTRIBUTES TO SAVE SPACE

 Response Artificial_frui :
             Df  Sum Sq Mean Sq F value  Pr(>F)
cluster       2   23.71 11.8569  3.0374 0.04929 *
Residuals   333 1299.93  3.9037
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Chocolate :
             Df  Sum Sq Mean Sq F value  Pr(>F)
cluster       2   20.13 10.0627  3.3357 0.03678 *
Residuals   333 1004.56  3.0167
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Response Burned :
             Df  Sum Sq Mean Sq F value    Pr(>F)
cluster       2   74.29  37.147   9.365 0.0001104 ***
Residuals   333 1320.88   3.967
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Leather :
             Df  Sum Sq Mean Sq F value    Pr(>F)
cluster       2   58.76 29.3810  7.3834 0.0007286 ***
Residuals   333 1325.12  3.9794
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


 Response Medicinal :
             Df  Sum Sq Mean Sq F value    Pr(>F)
cluster       2  113.09  56.546  15.307 4.369e-07 ***
Residuals   333 1230.17   3.694
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Response Band.aid :
             Df  Sum Sq Mean Sq F value    Pr(>F)
cluster       2   59.81 29.9058  7.4434 0.0006879 ***
Residuals   333 1337.91  4.0177
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

MULTIDIMENSIONAL SCALING (MDS)

In the first example I am doing metric MDS. One would NOT usually do an MDS on descriptive analysis data but since I wanted to continue to use the Torri data I created a distance matrix from the mean values of the Torri data set – similarly to the process used to do clustering.

1. I used the mtable function to create the means. I scaled the means and created a distance matrix using the Euclidean method.
   da.means=mtable(torriDAFinal, bycol="ProductName", firstvarcol=4)
   da.meansscaled <- scale(da.means)
   da.dist <- dist(da.meansscaled, method = "euclidean")

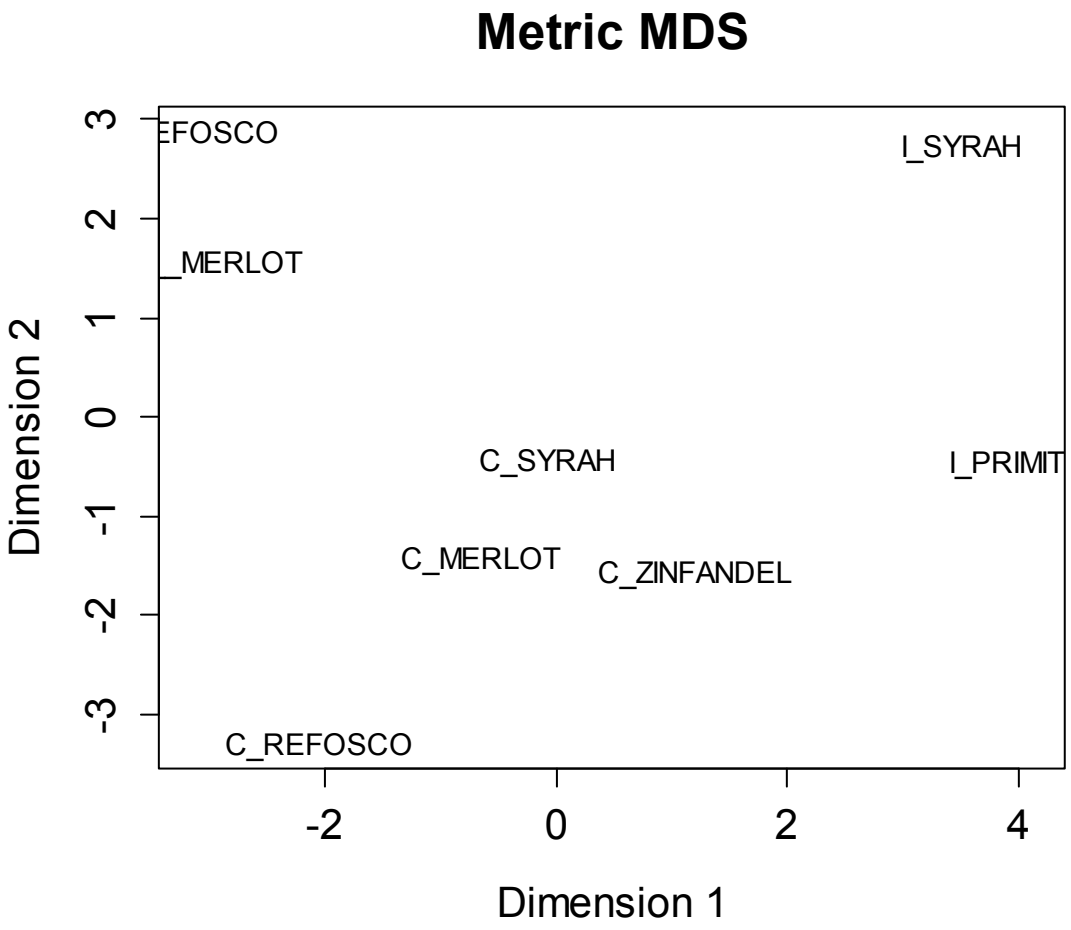
2. I ran a metric MDS for 2 dimensions (k=2) and printed the output
   da.mds <- cmdscale(da.dist,eig=TRUE, k=2)
   da.mds

```
$points
                      [,1]       [,2]
C_MERLOT    -0.66017528 -1.4236819
C_REFOSCO   -2.05394351 -3.2971758
C_SYRAH     -0.08165706 -0.4392567
C_ZINFANDEL  1.20071498 -1.5563951
I__MERLOT   -2.87771919  1.5668345
I_PRIMITIVO  4.11149279 -0.4593610
I_REFOSCO   -3.14226246  2.8789771
I_SYRAH      3.50354973  2.7300588

$eig
[1]  5.343721e+01  3.392126e+01  1.755089e+01  1.363281e+01
1.225350e+01
[6]  7.141861e+00  2.062458e+00 -7.819123e-16

$x
NULL

$ac
1] 0

$GOF
[1]  0.6239891  0.6239891
```

GOF = Goodness of Fit

3. I plotted the points
   x <- da.mds$points[,1]
   y <- da.mds$points[,2]
   plot(x, y, xlab="Dimension 1", ylab="Dimension 2",
   main="Metric MDS", type="n")
   text(x, y, labels = row.names(da.means), cex=.7)

```
x <- da.mds$points[,1]
> y <- da.mds$points[,2]
> plot(x, y, xlab="Dimension 1", ylab="Dimension 2",
+      main="Metric MDS", type="n")
> text(x, y, labels = row.names(da.means), cex=.7)
```



4. To do non-metric MDS I loaded the MASS package.
library(MASS)

```
> library(MASS)
```

5. I ran the non-metric MDS and asked for the output
da.nmmds <- isoMDS(da.dist, k=2)
da.nmmds

```
> da.nmmds <- isoMDS(da.dist, k=2)
initial  value 9.444389
iter   5 value 7.065995
iter  10 value 6.940918
final  value 6.928604
converged
> da.nmmds
$points
                [,1]       [,2]
_MERLOT    -0.6458262 -0.7683492
C_REFOSCO  -2.3142239 -3.3534599
C_SYRAH     0.4367199  0.5796763
C_ZINFANDEL 0.8963048 -1.0218355
```

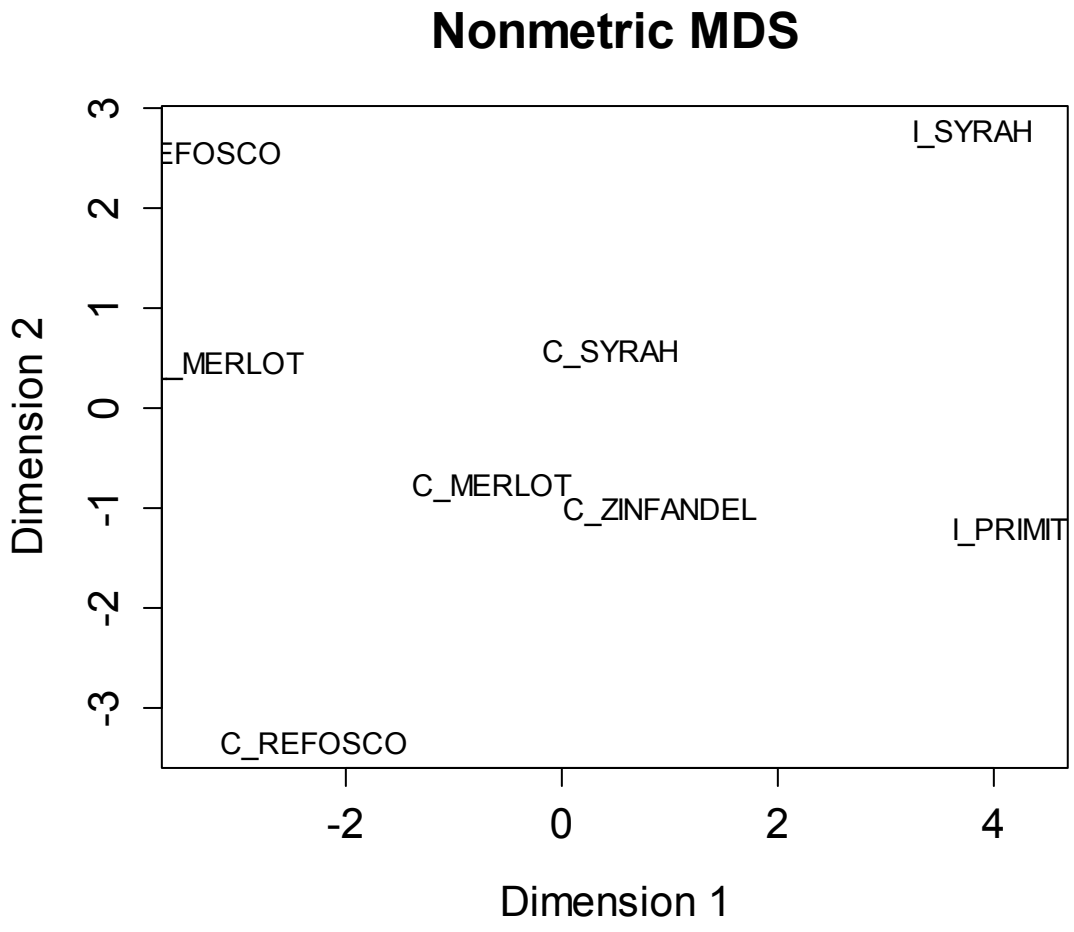
```
I__MERLOT    -3.1396778  0.4523043
I_PRIMITIVO   4.3784341 -1.2124140
I_REFOSCO    -3.3992202  2.5489715
I_SYRAH       3.7874893  2.7751065

$stress
[1]    6.928604
```

Unlike the METRIC MDS this output does give the Kruskal Stress

6. I plotted the results

x <- da.nmmds$points[,1]
y <- da.nmmds$points[,2]
plot(x, y, xlab="Dimension 1", ylab="Dimension 2",
main="Nonmetric MDS", type="n")
text(x, y, labels = row.names(da.means), cex=.7)

```
> x <- da.nmmds$points[,1]
> y <- da.nmmds$points[,2]
> plot(x, y, xlab="Dimension 1", ylab="Dimension 2",
+      main="Nonmetric MDS", type="n")
> text(x, y, labels = row.names(da.means), cex=.7)
```



DISTATIS

I did the DISTATIS on color data that came from sorting the wines into similarly colored groups. First I had to run the program named **distatiscode.R**. To do this I opened the file and ran the entire code.

1. Next I imported the data set named sorting_r1.csv making sure that I had set header = TRUE, and I named it sort.r1.d
   sort.r1.d = read.table("sorting_r1.csv", sep = ',', header=TRUE, row.names=1)

2. Then I checked to see if everything had worked I asked for a lot of information.
   head(sort.r1.d)
   str(sort.r1.d)
   dim(sort.r1.d)
   colnames(sort.r1.d)

```
> head(sort.r1.d)
            X263 X1331 X1400 X1401 X1402 X1404 X1405 X1408 X1409 X1412 X1413
I_REFOSCO     G6    G3    G5    G4    G5    G4    G2    G2    G1    G1    G1
I_MERLOT      G1    G3    G4    G3    G2    G3    G4    G3    G1    G2    G1
I_SYRAH       G5    G4    G2    G1    G1    G2    G1    G4    G1    G2    G1
I_PRIMITIVO   G2    G1    G3    G1    G4    G5    G3    G4    G3    G2    G2
C_SYRAH       G3    G2    G4    G4    G5    G6    G1    G3    G2    G4    G3
C_REFOSCO     G4    G3    G1    G2    G5    G5    G2    G1    G2    G3    G3
            X1414 X1415 X1416 X1417
I_REFOSCO      G6    G1    G4    G1
I_MERLOT       G2    G2    G2    G2
I_SYRAH        G3    G1    G1    G3
I_PRIMITIVO    G6    G1    G3    G4
C_SYRAH        G1    G2    G2    G5
C_REFOSCO      G5    G1    G1    G6
```

```
> str(sort.r1.d)
'data.frame':  8 obs. of  15 variables:
 $ X263 : Factor w/ 6 levels "G1","G2","G3",..: 6 1 5 2 3 4 1 1
 $ X1331: Factor w/ 5 levels "G1","G2","G3",..: 3 3 4 1 2 3 2 5
 $ X1400: Factor w/ 5 levels "G1","G2","G3",..: 5 4 2 3 4 1 3 3
 $ X1401: Factor w/ 4 levels "G1","G2","G3",..: 4 3 1 1 4 2 3 3
 $ X1402: Factor w/ 5 levels "G1","G2","G3",..: 5 2 1 4 5 5 3 2
 $ X1404: Factor w/ 6 levels "G1","G2","G3",..: 4 3 2 5 6 5 1 1
 $ X1405: Factor w/ 4 levels "G1","G2","G3",..: 2 4 1 3 1 2 4 1
 $ X1408: Factor w/ 4 levels "G1","G2","G3",..: 2 3 4 4 3 1 3 3
 $ X1409: Factor w/ 3 levels "G1","G2","G3": 1 1 1 3 2 2 2 1
 $ X1412: Factor w/ 4 levels "G1","G2","G3",..: 1 2 2 2 4 3 2 2
 $ X1413: Factor w/ 3 levels "G1","G2","G3": 1 1 1 2 3 3 1 1
 $ X1414: Factor w/ 6 levels "G1","G2","G3",..: 6 2 3 6 1 5 1 4
 $ X1415: Factor w/ 2 levels "G1","G2": 1 2 1 1 2 1 2 2
 $ X1416: Factor w/ 4 levels "G1","G2","G3",..: 4 2 1 3 2 1 1 4
 $ X1417: Factor w/ 8 levels "G1","G2","G3",..: 1 2 3 4 5 6 7 8
```

```
> dim(sort.r1.d)
[1]  8 15
```

I knew I had 15 panelists and 8 wines and this checked out.

```
> colnames(sort.r1.d)
 [1] "X263"  "X1331" "X1400" "X1401" "X1402" "X1404" "X1405" "X1408" "X1409"
[10] "X1412" "X1413" "X1414" "X1415" "X1416" "X1417"
```

3. Next I needed to create distance matrices for each panelist using the cluster package

library(cluster)
r1.1 = as.matrix(daisy(sort.r1.d['X263']))
r1.2 = as.matrix(daisy(sort.r1.d['X1331']))
r1.3 = as.matrix(daisy(sort.r1.d['X1400']))
r1.4 = as.matrix(daisy(sort.r1.d['X1401']))
r1.5 = as.matrix(daisy(sort.r1.d['X1402']))
r1.6 = as.matrix(daisy(sort.r1.d['X1404']))
r1.7 = as.matrix(daisy(sort.r1.d['X1405']))
r1.8 = as.matrix(daisy(sort.r1.d['X1408']))
r1.9 = as.matrix(daisy(sort.r1.d['X1409']))
r1.10 = as.matrix(daisy(sort.r1.d['X1412']))
r1.11 = as.matrix(daisy(sort.r1.d['X1413']))
r1.12 = as.matrix(daisy(sort.r1.d['X1414']))
r1.13 = as.matrix(daisy(sort.r1.d['X1415']))
r1.14 = as.matrix(daisy(sort.r1.d['X1416']))
r1.15 = as.matrix(daisy(sort.r1.d['X1417']))

4. Then I had to combine the individual distance matrices rowwise (in other words I had to make a long and skinny matrix)

r1.dist = rbind(r1.1, r1.2, r1.3, r1.4, r1.5, r1.6, r1.7, r1.8, r1.9, r1.10,
r1.11, r1.12, r1.13, r1.14, r1.15)

5. Then I wanted to see what I had

head(r1.dist)
dim(r1.dist)
rownames(r1.dist)

```
> head(r1.dist)
            I_REFOSCO I_MERLOT I_SYRAH I_PRIMITIVO C_SYRAH C_REFOSCO C_MERLOT
I_REFOSCO           0        1       1           1       1         1        1
I_MERLOT            1        0       1           1       1         1        0
I_SYRAH             1        1       0           1       1         1        1
I_PRIMITIVO         1        1       1           0       1         1        1
C_SYRAH             1        1       1           1       0         1        1
C_REFOSCO           1        1       1           1       1         0        1
            C_ZINFANDEL
I_REFOSCO             1
I_MERLOT              0
I_SYRAH               1
I_PRIMITIVO           1
C_SYRAH               1
C_REFOSCO             1

> dim(r1.dist)
[1] 120   8
```

8 wines * 15 matrices = 120 rows.

```
> rownames(r1.dist)
  [1] "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"
  [6] "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"
 [11] "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"
 [16] "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO"
 [21] "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"
 [26] "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"
 [31] "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"
 [36] "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL"
 [41] "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"
 [46] "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"
 [51] "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"
 [56] "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO"
 [61] "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"
 [66] "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"
 [71] "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"
 [76] "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL"
 [81] "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"
 [86] "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"
 [91] "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"
 [96] "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO"
[101] "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"
[106] "I_MERLOT"     "I_SYRAH"      "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"
[111] "C_MERLOT"     "C_ZINFANDEL" "I_REFOSCO"    "I_MERLOT"     "I_SYRAH"
[116] "I_PRIMITIVO" "C_SYRAH"      "C_REFOSCO"    "C_MERLOT"     "C_ZINFANDEL"
```

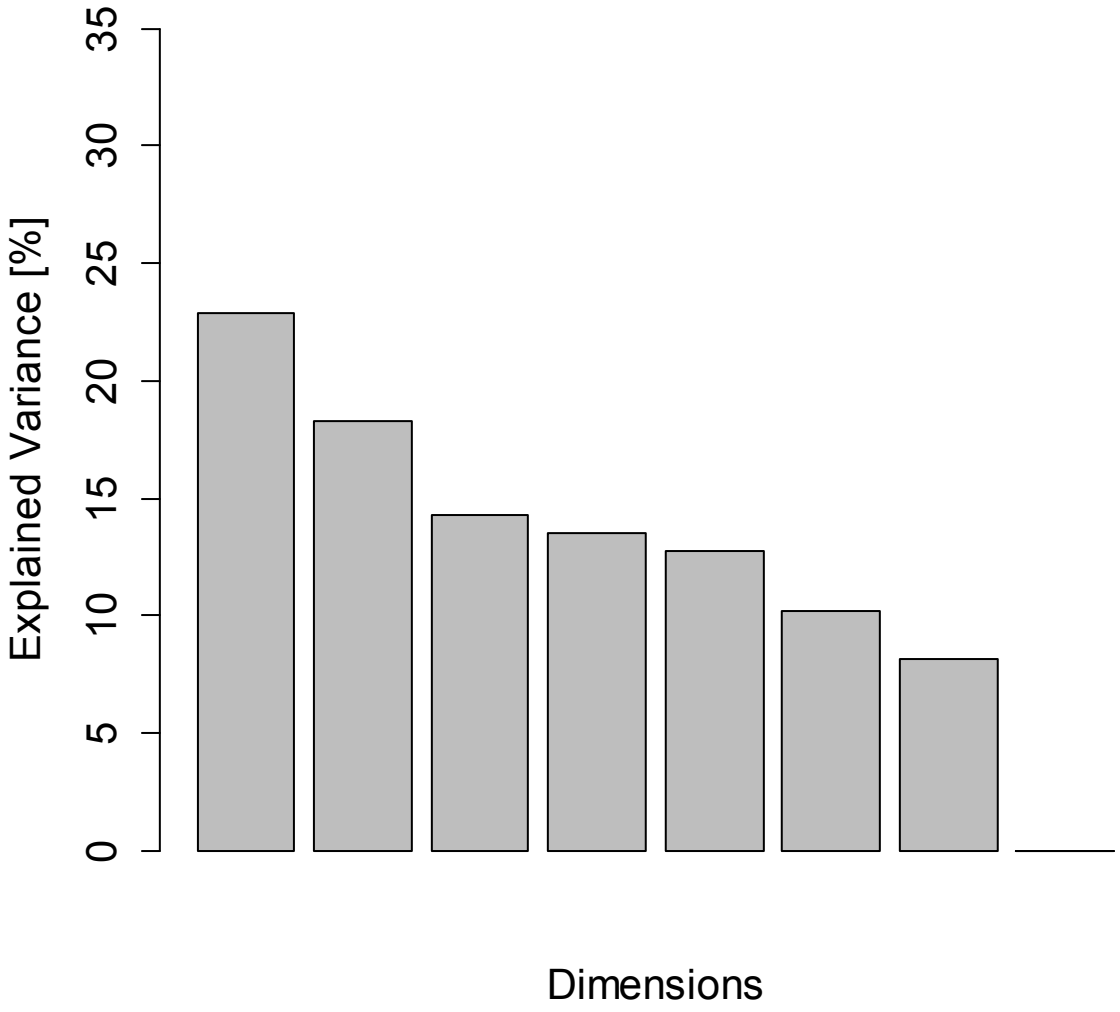
6. Next I had to run the distatis
   r1.distatis = distatis(r1.dist, s=15, p=8, coord=c(1,2), col='black')

7. And then I needed create various data sets form the output to allow me to plot the results.

   G1 = r1.distatis$sujets[,1:2] # RV matrix G
   F1 = r1.distatis$produits[,1:2] # compromise factor scores a.k.a. Consensus prod position
   a1 = rownames(r1.dist[1:8,]) # product names
   p1 = r1.distatis$p # explained variances
   f1 = r1.distatis$f # projected supplementary distance matrices
   ps = r1.distatis$pourcent_sub # explained variances for RV matrix of judges

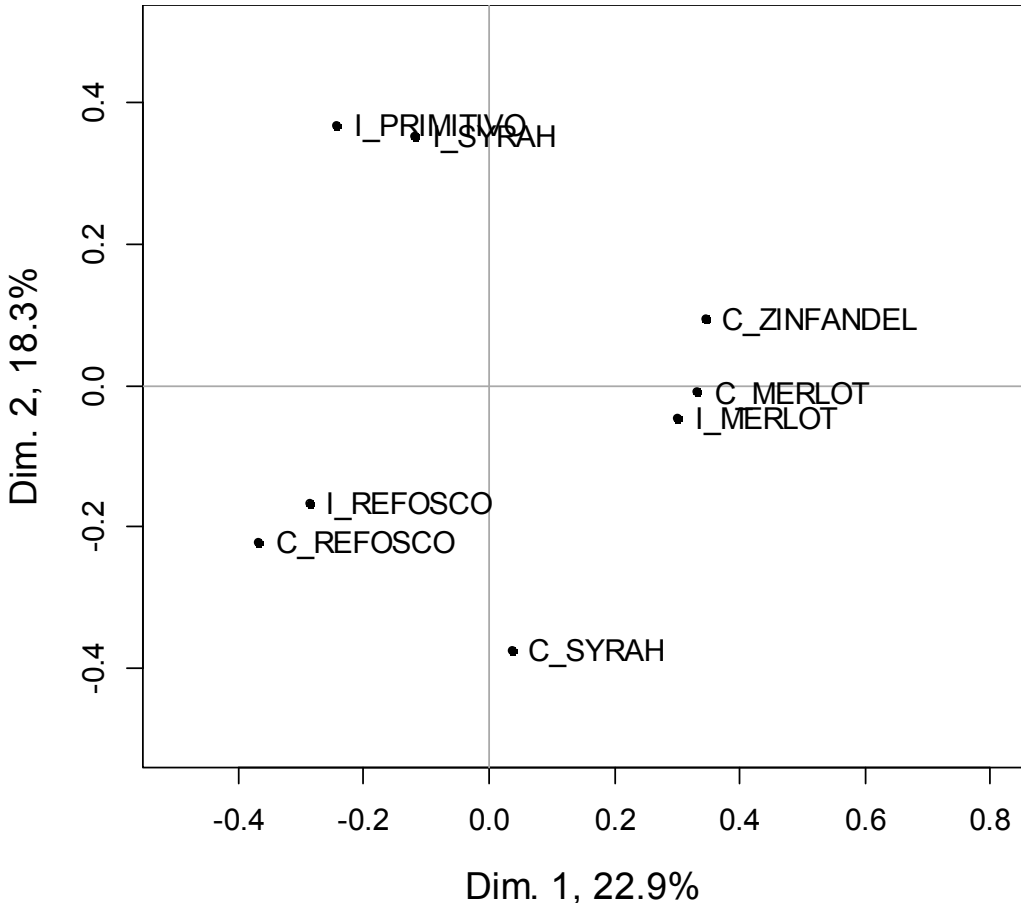
8. I asked for the Eigenvalue plot

```
par(mar = c(4,4,1,0))
barplot(p1, xlab = "Dimensions", ylab = "Explained Variance [%]", ylim = c(0,35))
```



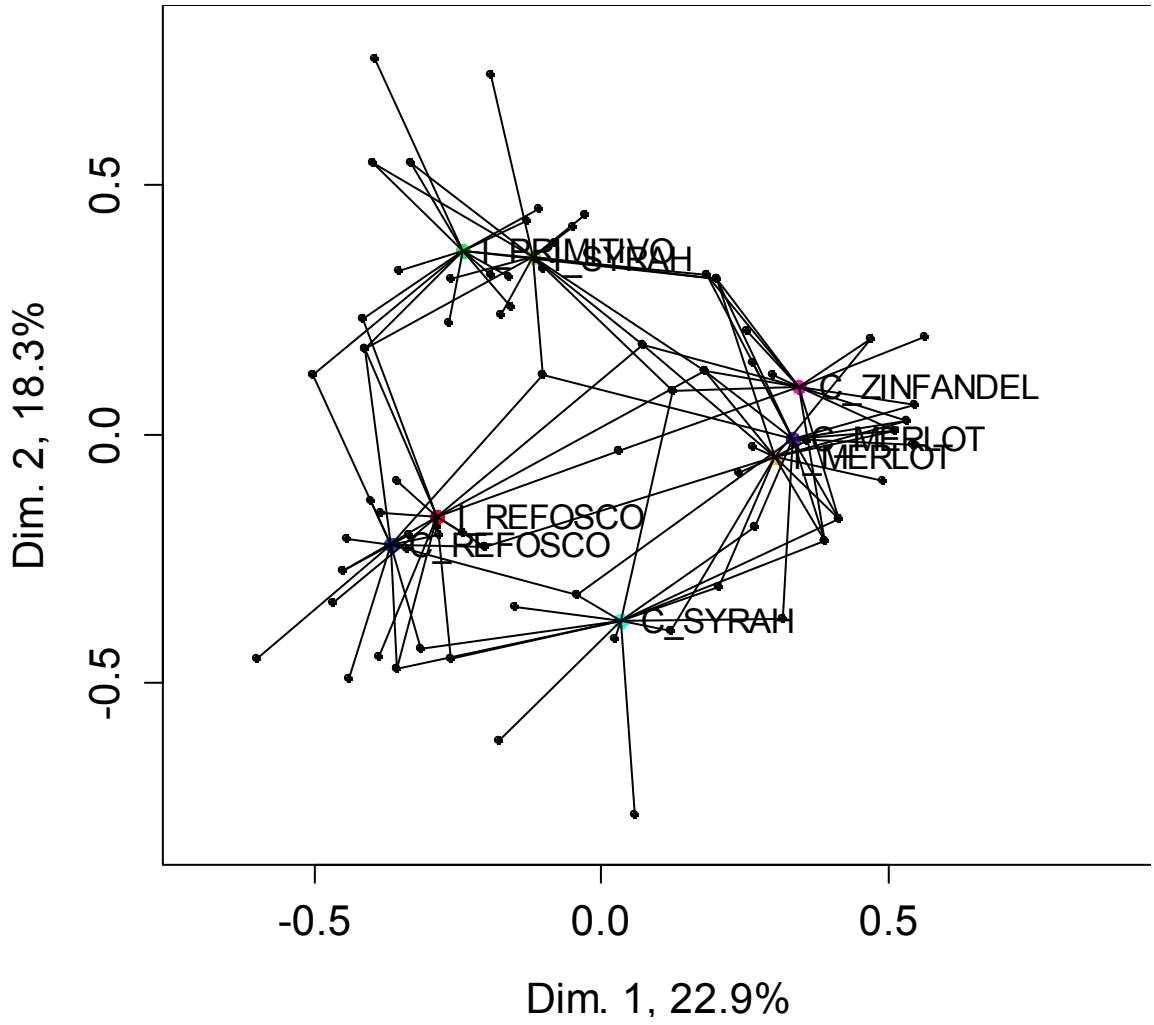
9. Then I plotted the DISTATIS product or score plot

```
par(mar = c(4,4,0,0))
plot(F1, pch = 20, xlab = paste("Dim. 1, ", round(p1[1], 1), "%", sep=""),
    ylab = paste("Dim. 2, ", round(p1[2], 1), "%", sep=""), cex.axis = .8,
    xlim = c(-.5, .8), ylim = c(-.5, .5))
abline (h=0, v=0, col='darkgray')
text(F1[,1], F1[,2], labels = a1, col='black', cex=.8, pos = 4)
```



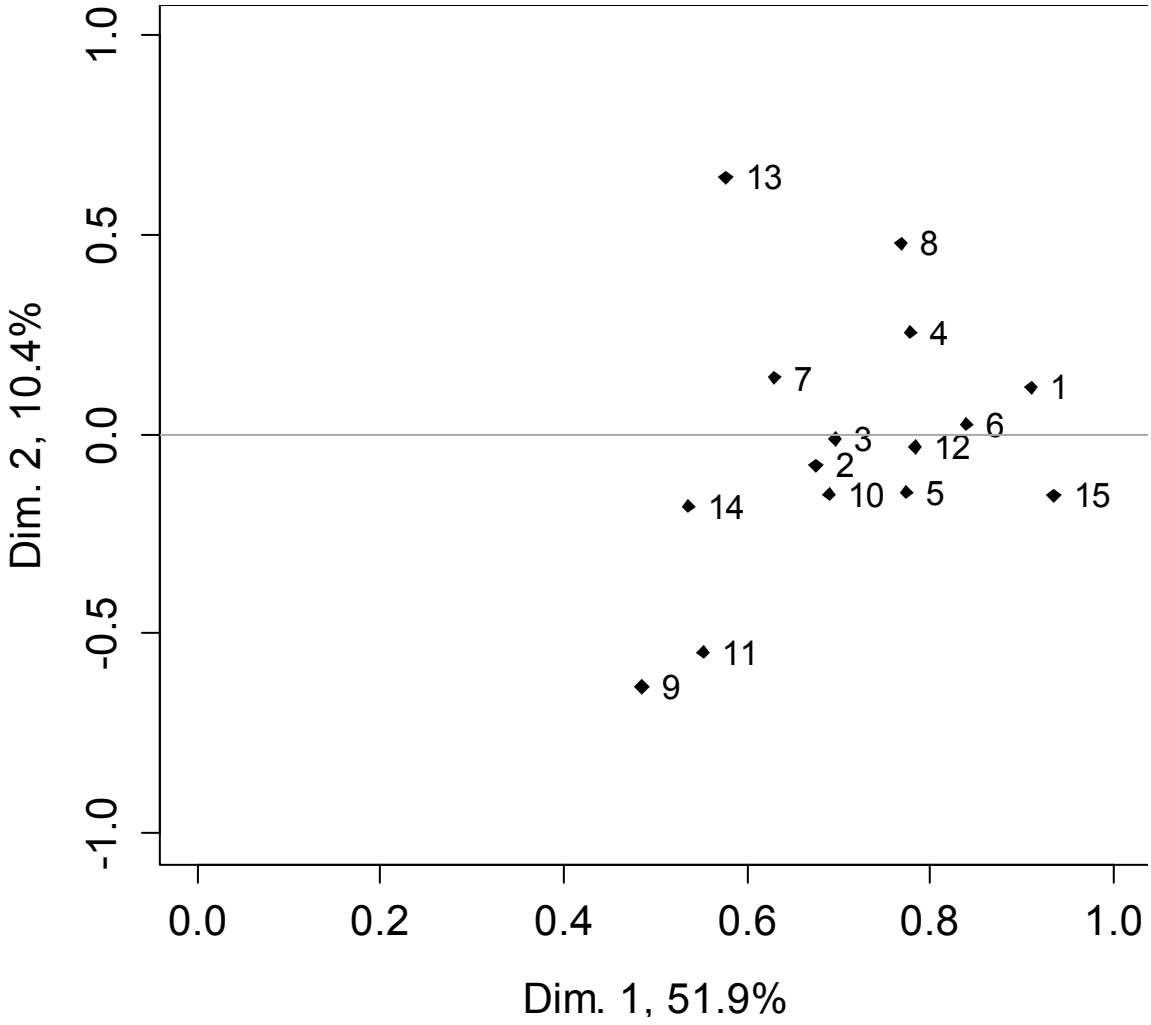
10. Then plotted the DISTATIS consensus plot with the individual judge positions shown.

```
plot(F1, xlim = c(-.7, .9), ylim = c(-.8, .8), pch = 19, col = rainbow(8),
    xlab = paste("Dim. 1, ", round(p1[1], 1), "%", sep=""),
    ylab = paste("Dim. 2, ", round(p1[2], 1), "%", sep=""))
for (i in (1:8)) # Loop for the rows a.k.a. number of products
  {
  points(t(f1[i,c(1,2),]), cex=0.8,  pch = 20)
  arrows(F1[i,1], F1[i,2], t(f1[i,1,]), t(f1[i,2,]), length = 0)
  }
text(F1[,1], F1[,2], labels = a1,  cex = .8, pos = 4)
```



11. Then I checked to see how well the judges agreed based on a PCA of the RV matrix.

```
plot(c(0,1), c(-1,1), type="n",
    xlab = paste("Dim. 1, ", round(ps[1], 1), "%", sep=""),
    ylab = paste("Dim. 2, ", round(ps[2], 1), "%", sep=""))
points(-G1[,1], G1[,2], pch = 18)
text(-G1[,1] , G1[,2], cex=0.8, pos=4)
abline(h=0, col = 'darkgray')
```



I could also have printed out the RV matrix and the other numerical output.

12. I then decided to do an MDS on the same data. So I read the data set in and checked it as before (see DISTATIS for the output)..
sort.r1.d = read.table('sorting_r1.csv', sep = ',', header=TRUE, row.names=1)
head(sort.r1.d)
str(sort.r1.d)
dim(sort.r1.d) # 8 wines, 15 panelists

13. Since MDS does not accept individual data I asked R to create a global distance matrix for all the judges combined.
library(cluster)
sort.r1.dist = daisy(sort.r1.d)
sort.r1.dist

```
> library(cluster)
> sort.r1.dist = daisy(sort.r1.d)
> sort.r1.dist
```

```
Dissimilarities :
            I_REFOSCO  I_MERLOT   I_SYRAH I_PRIMITIVO   C_SYRAH C_REFOSCO
I_MERLOT    0.8000000
I_SYRAH     0.8000000 0.8000000
I_PRIMITIVO 0.8666667 0.9333333 0.7333333
C_SYRAH     0.8666667 0.7333333 0.9333333   1.0000000
C_REFOSCO   0.7333333 0.9333333 0.8666667   0.8666667 0.8000000
C_MERLOT    0.9333333 0.5333333 0.8000000   0.8666667 0.6666667 0.8666667
C_ZINFANDEL 0.8000000 0.4666667 0.7333333   0.8666667 0.8000000 1.0000000
             C_MERLOT
I_MERLOT
I_SYRAH
I_PRIMITIVO
C_SYRAH
C_REFOSCO
C_MERLOT
C_ZINFANDEL 0.4666667

Metric :  mixed ;  Types = N, N, N, N, N, N, N, N, N, N, N, N, N, N, N
Number of objects : 8
```

14. Then I ran the MDS using the package MASS

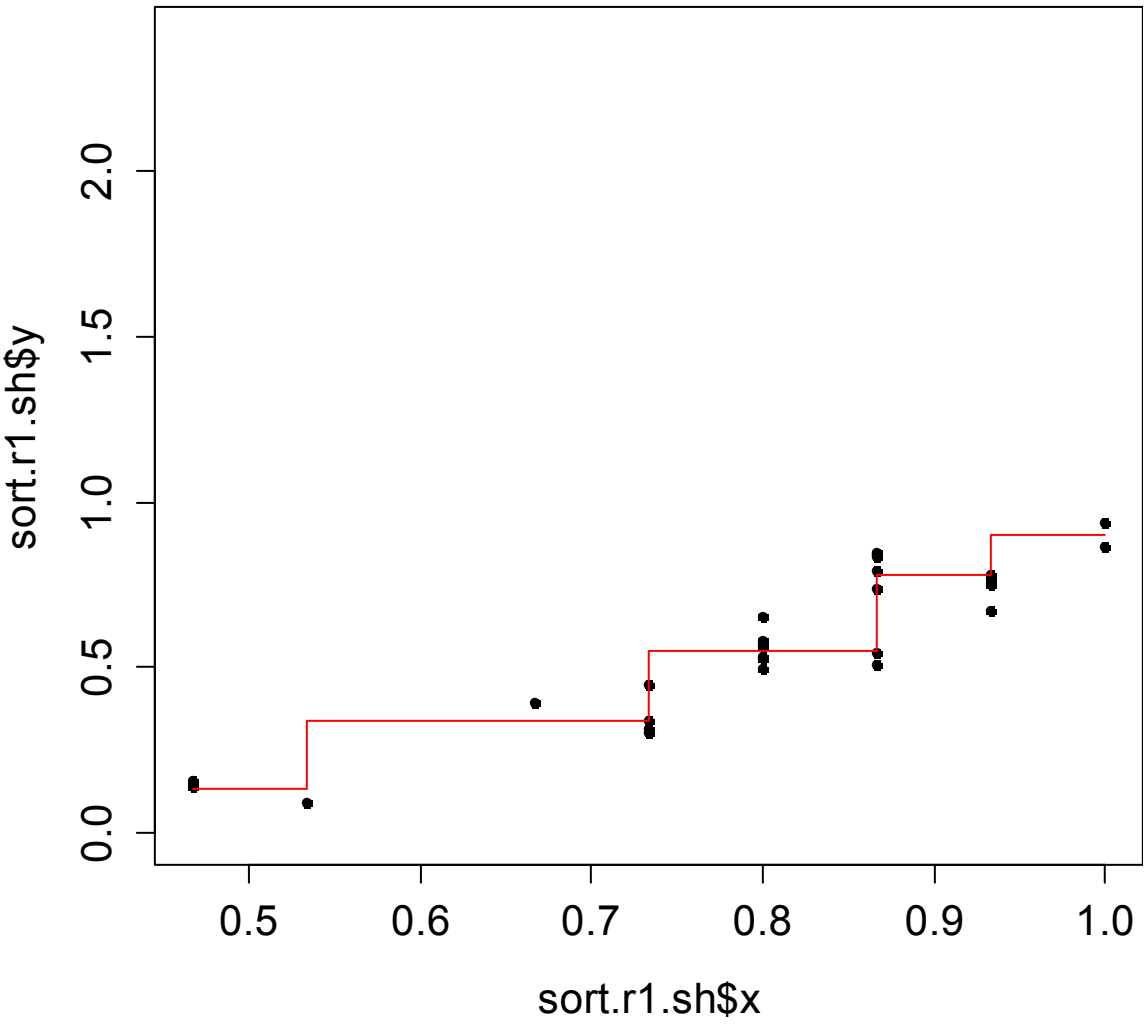
library(MASS)
sort.r1.mds1 = cmdscale(sort.r1.dist)
sort.r1.mds = isoMDS(sort.r1.dist, y = sort.r1.mds1)
sort.r1.sh = Shepard(sort.r1.dist, sort.r1.mds$points)
sort.r1.mds$stress # Kruskal's Stress

```
> library(MASS)
> sort.r1.mds1 = cmdscale(sort.r1.dist)
> sort.r1.mds = isoMDS(sort.r1.dist, y = sort.r1.mds1)
initial  value 11.632994
iter   5 value 7.321194
iter   5 value 7.320825
final  value 7.020712
converged
> sort.r1.sh = Shepard(sort.r1.dist, sort.r1.mds$points)
> sort.r1.mds$stress  # Kruskal's Stress
[1] 7.020712
```

15. I asked for a Shepard's plot

par(mar = c(4,4,0.1,0.1))
plot(sort.r1.sh, pch=20, ylim = c(0,2.4))
lines(sort.r1.sh$x, sort.r1.sh$yf, type = "S", col='red')

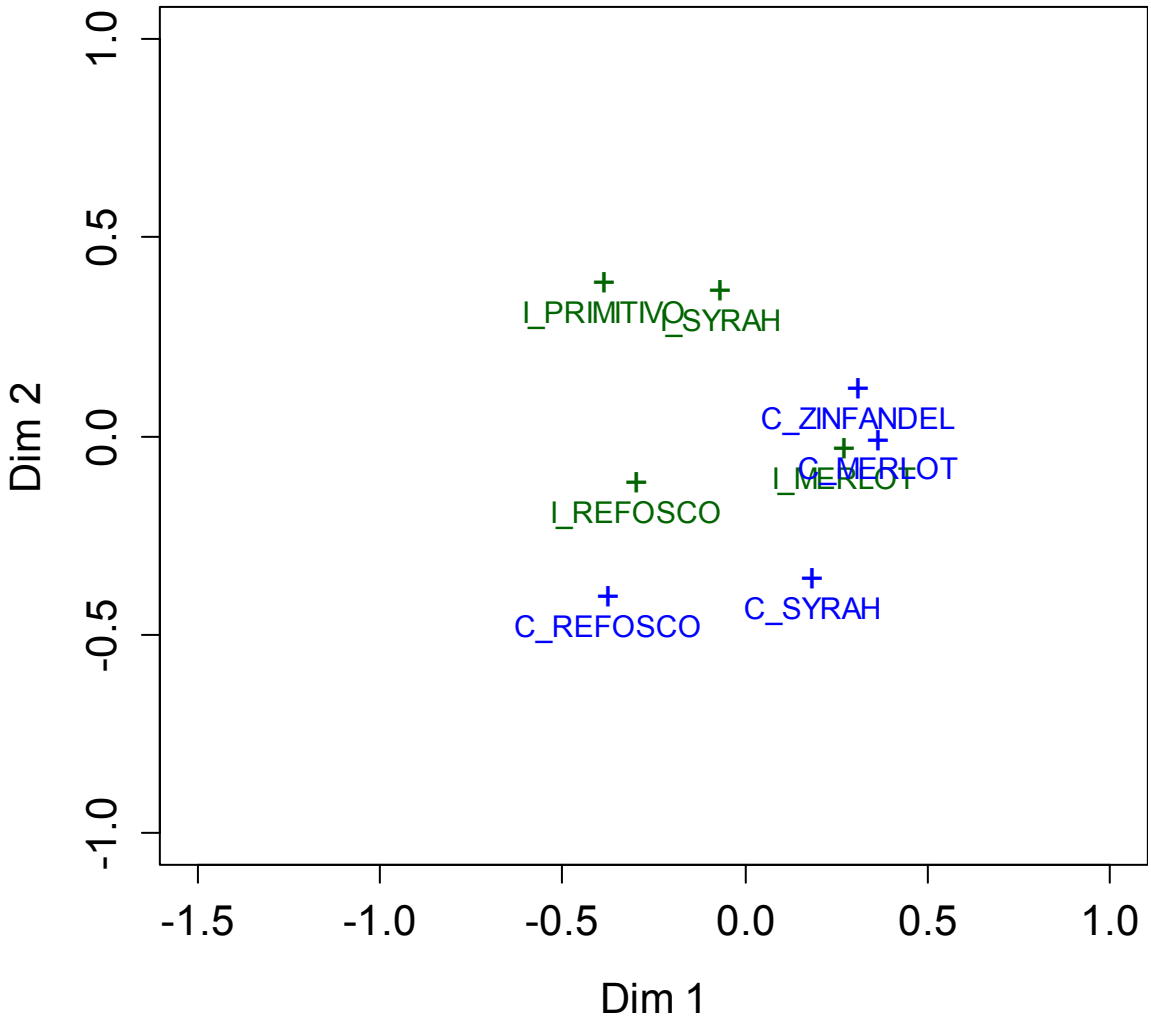
```
> # Shepard's plot
> par(mar = c(4,4,0.1,0.1))
> plot(sort.r1.sh, pch=20, ylim = c(0,2.4))
> lines(sort.r1.sh$x, sort.r1.sh$yf, type = "S", col='red')
>
```



16. And lastly I asked for an MDS product plot

plot(sort.r1.mds$points, pch='+', xlab = 'Dim 1', ylab = 'Dim 2',
xlim = c(-1.5,1), ylim = c(-1,1), col = c(rep('darkgreen',4), rep('blue',4)))
text(sort.r1.mds$points, labels = rownames(sort.r1.mds$points), cex=.7, pos=1,
col = c(rep('darkgreen',4), rep('blue',4)))

```
> # MDS product plot
> plot(sort.r1.mds$points, pch='+', xlab = 'Dim 1', ylab = 'Dim 2',
+      xlim = c(-1.5,1), ylim = c(-1,1), col = c(rep('darkgreen',4),
rep('blue',4)))
> text(sort.r1.mds$points, labels = rownames(sort.r1.mds$points), cex=.7,
pos=1,
+      col = c(rep('darkgreen',4), rep('blue',4))
```


I_PRIMITIVO
I_SYRAH
C_ZINFANDEL
C_MERLOT
I_MERLOT
I_REFOSCO
C_REFOSCO
C_SYRAH
Dim 1
Dim 2
-1.5
-1.0
-0.5
0.0
0.5
1.0
-1.0
-0.5
0.0
0.5
1.0

CORRESPONDENCE ANALYSIS

For this example I used the 'author' data set (I shortened a few names to fit here)

	a	b	c	d	e	f	g	h	i	j	k	l	m
three daughters (buck)	550	116	147	374	1015	131	131	493	442	2	52	302	159
drifters (michener)	515	109	172	311	827	167	136	376	432	8	61	280	146
lost world (clark)	590	112	181	265	940	137	119	419	514	6	46	335	176
east wind (buck)	557	129	128	343	996	158	129	571	555	4	76	291	247
farewelltoarms (hemingway)	589	72	129	339	866	108	159	449	472	7	59	264	158
soundandfury7 (faulkner)	541	109	136	228	763	126	129	401	520	5	72	280	209
soundandfury6 (faulkner)	517	96	127	356	771	115	189	478	558	6	80	322	163
profiles of future (clark)	592	151	251	238	985	168	152	381	544	7	39	416	236
islands (hemingway)	576	120	136	404	873	122	156	593	406	3	90	281	142
pendorric 3 (holt)	557	97	145	354	909	97	121	479	431	10	94	240	154
asia (michener)	554	108	206	243	797	164	100	328	471	4	34	293	149
pendorric 2 (holt)	541	93	149	390	887	133	154	463	518	4	65	265	194

	n	o	p	q	r	s	t	u	v	w	x	y	z
three daughters (buck)	534	516	115	4	409	467	632	174	66	155	5	150	3
drifters (michener)	470	561	140	4	368	387	632	195	60	156	14	137	5
lost world (clark)	403	505	147	8	395	464	670	224	113	146	13	162	10
east wind (buck)	479	509	92	3	413	533	632	181	68	187	10	184	4
farewelltoarms (hemingway)	504	542	95	0	416	314	691	197	64	225	1	155	2
soundandfury7 (faulkner)	471	589	84	2	324	454	672	247	71	160	11	280	1
soundandfury6 (faulkner)	483	617	82	8	294	358	685	225	37	216	12	171	5
profiles of future (clark)	526	524	107	9	418	508	655	226	89	106	15	142	20
islands (hemingway)	516	488	91	3	339	349	640	194	40	250	3	104	5
pendorric 3 (holt)	417	477	100	3	305	415	597	237	64	194	9	140	4
asia (michener)	482	532	145	8	361	402	630	196	66	149	2	80	6
pendorric 2 (holt)	484	545	70	4	299	423	644	193	66	218	2	127	2

1. Once I loaded the data set I typed the following and saw the data shown above
   list(author)

2. Next I loaded the ca package and checked that it was loaded.
   library(ca)

3. Then I ran the simple correspondence analysis and asked for the output
   ca=ca(author)
   summary(ca)
   list(ca)

```
> ca=ca(author)
> summary(ca)

Principal inertias (eigenvalues):
 dim    value      %   cum%   scree plot
 1      0.007664  40.9  40.9  ************************
 2      0.003688  19.7  60.6  ************
 3      0.002411  12.9  73.5  ********
 4      0.001383   7.4  80.8  ****
 5      0.001002   5.3  86.2  ***
 6      0.000723   3.9  90.1  **
 7      0.000659   3.5  93.6  **
 8      0.000455   2.4  96.0  *
 9      0.000374   2.0  98.0  *
 10     0.000263   1.4  99.4
 11     0.000113   0.6 100.0
        -------- -----
 Total: 0.018735 100.0
```

```
Rows:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1  | thrd |   85  251   44 |   -8   7   1 |  -48 244  54 |
2  | drft |   80  208   38 |   36 140  13 |  -25  67  13 |
3  | lstw |   85  623   75 |  101 623 114 |   -1   0   0 |
4  | estw |   89   66   67 |  -15  16   3 |   26  49  17 |
5  | frwl |   82  356   66 |  -73 351  57 |   -8   5   2 |
6  | snd7 |   82  928  131 |   26  23   8 |  164 904 603 |
7  | snd6 |   83  497   90 |  -81 326  71 |   59 171  78 |
8  | prfl |   90  815  168 |  168 808 332 |  -15   7   6 |
9  | isln |   83  875  121 | -137 686 203 |  -72 189 116 |
10 | pnd3 |   80  318   54 |  -63 314  42 |   -6   3   1 |
11 | asmc |   78  659  100 |  103 443 108 |  -72 216 110 |
12 | pnd2 |   83  439   45 |  -67 436  48 |   -6   3   1 |

Columns:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1  |    a |   80  162   10 |    2   1   0 |  -19 161   8 |
2  |    b |   16  365   18 |   86 338  15 |  -24  27   2 |
3  |    c |   23  831   60 |  185 691 102 |  -83 140  43 |
4  |    d |   46  920   89 | -169 788 170 |  -69 132  59 |
5  |    e |  127  357   34 |    8  12   1 |  -42 345  60 |
6  |    f |   19  529   28 |  112 456  32 |  -45  72  10 |
7  |    g |   20  344   26 |  -89 325  21 |   21  19   2 |
8  |    h |   65  735   83 | -131 721 146 |  -18  14   6 |
9  |    i |   70  465   28 |   23  74   5 |   54 392  55 |
10 |    j |    1   28    7 |   40   9   0 |   56  18   1 |
11 |    k |    9  724   43 | -241 661  70 |   75  64  14 |
12 |    l |   43  555   33 |   89 548  44 |  -10   7   1 |
13 |    m |   26  436   35 |   62 153  13 |   85 284  50 |
14 |    n |   69  166   21 |  -18  54   3 |  -25 112  12 |
15 |    o |   77  205   32 |   -9  12   1 |   39 193  31 |
16 |    p |   15  515   51 |  141 317  39 | -112 198  51 |
17 |    q |    1  416   12 |  357 376  11 | -116  40   2 |
18 |    r |   52  374   35 |   52 215  18 |  -45 159  28 |
19 |    s |   61  413   49 |   75 374  45 |   25  40  10 |
20 |    t |   93   90   13 |   -9  30   1 |   12  59   4 |
21 |    u |   30  283   23 |   14  14   1 |   62 268  31 |
22 |    v |   10  550   37 |  200 548  50 |   11   2   0 |
23 |    w |   26  888   75 | -219 883 161 |  -17   6   2 |
24 |    x |    1  418   22 |  292 237  13 |  256 182  21 |
25 |    y |   22  899  106 |    0   0   0 |  286 899 485 |
26 |    z |    1  576   30 |  596 511  37 | -213  65  10 |

> list(ca)
[[1]]

 Principal inertias (eigenvalues):
            1        2        3        4        5        6        7
Value       0.007664 0.003688 0.002411 0.001383 0.001002 0.000723 0.000659
Percentage 40.91%   19.69%   12.87%   7.38%    5.35%    3.86%    3.52%
            8        9        10       11
Value       0.000455 0.000374 0.000263 0.000113
Percentage 2.43%    2%       1.4%     0.6%

 Rows:
        three daughters (buck) drifters (michener) lost world (clark)
Mass                  0.085407            0.079728           0.084881
ChiDist               0.097831            0.094815           0.128432
Inertia               0.000817            0.000717           0.001400
Dim. 1               -0.095388            0.405697           1.157803
Dim. 2               -0.794999           -0.405560          -0.023114
        east wind (buck) farewell to arms (hemingway)
Mass            0.089411                     0.082215
```

ChiDist	0.118655	0.122889
Inertia	0.001259	0.001242
Dim. 1	-0.173901	-0.831886
Dim. 2	0.434443	-0.136485

	sound and fury 7 (faulkner)	sound and fury 6 (faulkner)
Mass	0.082310	0.083338
ChiDist	0.172918	0.141937
Inertia	0.002461	0.001679
Dim. 1	0.302025	-0.925572
Dim. 2	2.707599	0.966944

	profiles of future (clark)	islands (hemingway)	pendorric 3 (holt)
Mass	0.089722	0.082776	0.079501
ChiDist	0.187358	0.165529	0.113174
Inertia	0.003150	0.002268	0.001018
Dim. 1	1.924060	-1.566481	-0.724758
Dim. 2	-0.249310	-1.185338	-0.106349

	asia (michener)	pendorric 2 (holt)
Mass	0.077827	0.082884
ChiDist	0.155115	0.101369
Inertia	0.001873	0.000852
Dim. 1	1.179548	-0.764937
Dim. 2	-1.186934	-0.091188

Columns:

	a	b	c	d	e	f	g
Mass	0.079847	0.015685	0.022798	0.045967	0.127070	0.019439	0.020025
ChiDist	0.048441	0.148142	0.222783	0.189938	0.070788	0.165442	0.156640
Inertia	0.000187	0.000344	0.001132	0.001658	0.000637	0.000532	0.000491
Dim. 1	0.017623	0.984463	2.115029	-1.925632	0.086722	1.276526	-1.020713
Dim. 2	-0.320271	-0.398032	-1.373448	-1.135362	-0.684785	-0.732952	0.353017

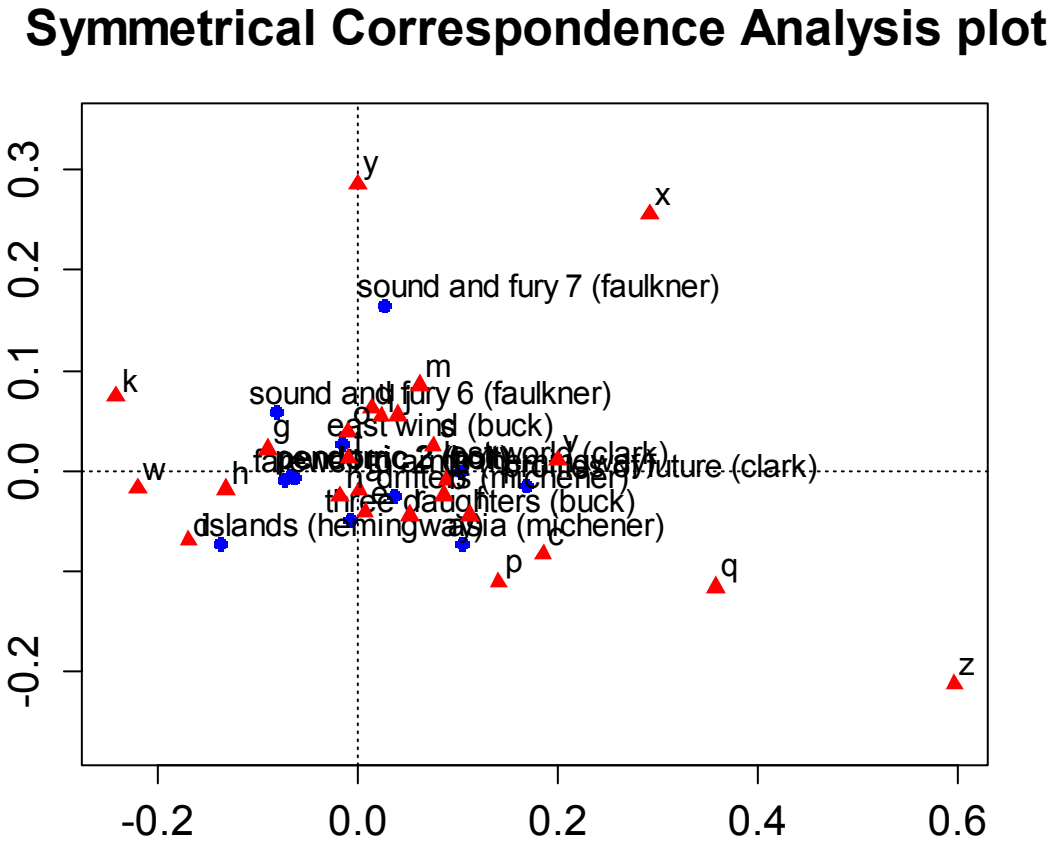
	h	i	j	k	l	m	n
Mass	0.064928	0.070092	0.000789	0.009181	0.042667	0.025500	0.068968
ChiDist	0.154745	0.086328	0.412075	0.296727	0.120397	0.159747	0.075706
Inertia	0.001555	0.000522	0.000134	0.000808	0.000618	0.000651	0.000395
Dim. 1	-1.501277	0.267473	0.453341	-2.755177	1.018257	0.712695	-0.200364
Dim. 2	-0.302413	0.889546	0.916032	1.231557	-0.165020	1.400966	-0.417258

	o	p	q	r	s	t	u
Mass	0.076572	0.015159	0.000669	0.051897	0.060660	0.093010	0.029756
ChiDist	0.088101	0.250617	0.582298	0.111725	0.123217	0.050630	0.119215
Inertia	0.000594	0.000952	0.000227	0.000648	0.000921	0.000238	0.000423
Dim. 1	-0.108491	1.610807	4.079786	0.591372	0.860202	-0.100464	0.163295
Dim. 2	0.637987	-1.837948	-1.914791	-0.734216	0.405610	0.203141	1.017140

	v	w	x	y	z
Mass	0.009612	0.025847	0.001160	0.021902	0.000801
ChiDist	0.269770	0.232868	0.600831	0.301376	0.833700
Inertia	0.000700	0.001402	0.000419	0.001989	0.000557
Dim. 1	2.281333	-2.499232	3.340505	0.001519	6.808100
Dim. 2	0.177022	-0.284722	4.215355	4.706083	-3.509223

4. Next I started to make plots. The first was the symmetrical plot.

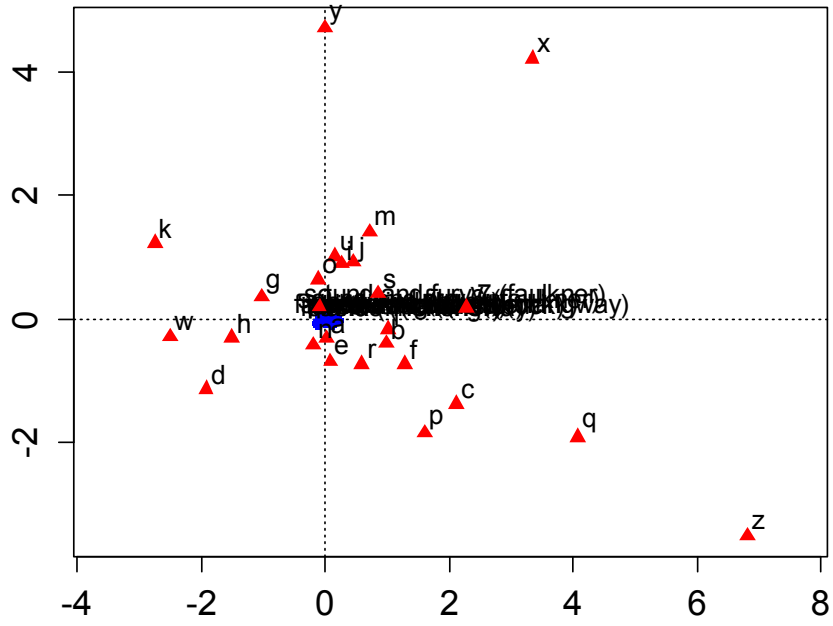
```
plot(ca, dim = c(1,2), map = "symmetric", what = c("all", "all"),
main="Symmetrical Correspondence Analysis plot",
    mass = c(FALSE, FALSE), contrib = c("none", "none"),
    labels =  2, arrows = c(FALSE, FALSE))
```



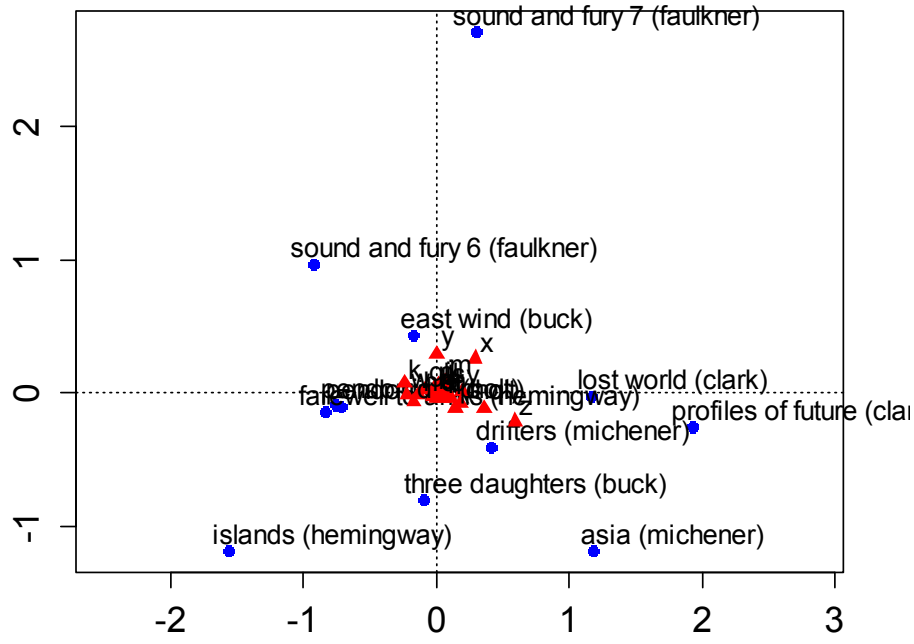
5. I also did asymmetrical plots which looked terrible.

```
plot(ca, dim = c(1,2), map = "rowprincipal", what = c("all", "all"),
main="Asymmetrical Row Correspondence Analysis plot",
    mass = c(FALSE, FALSE), contrib = c("none", "none"),
    labels =  2, arrows = c(FALSE, FALSE))
plot(ca, dim = c(1,2), map = "colprincipal", what = c("all", "all"),
main="Asymmetrical Column Correspondence Analysis plot",
    mass = c(FALSE, FALSE), contrib = c("none", "none"),
    labels =  2, arrows = c(FALSE, FALSE))
```

Asymmetrical Row Correspondence Analysis pl

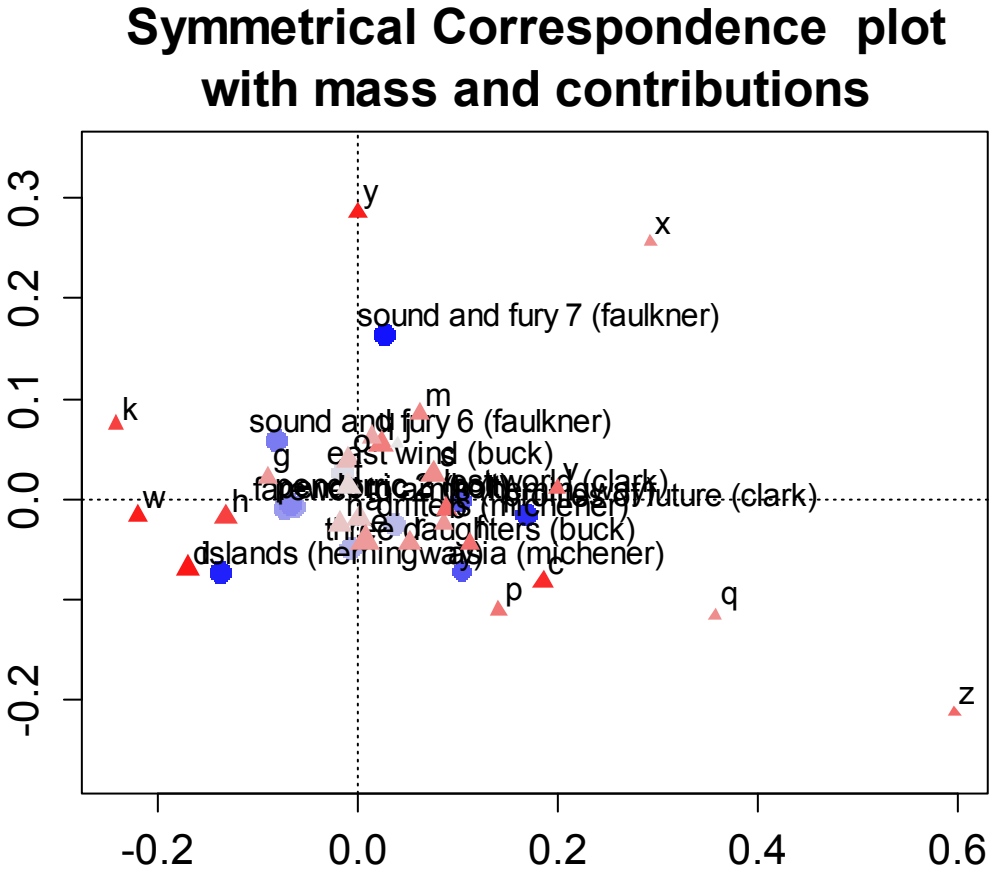


Asymmetrical Column Correspondence Analysis



6. Then I did a symmetric plot showing the mass and contributions

plot(ca, dim = c(1,2), map = "symmetric", what = c("all", "all"),
main="Symmetrical Correspondence plot\nwith mass and contributions",
mass = c(TRUE, TRUE), contrib = c("relative", "relative"),
labels = 2, arrows = c(FALSE, FALSE))

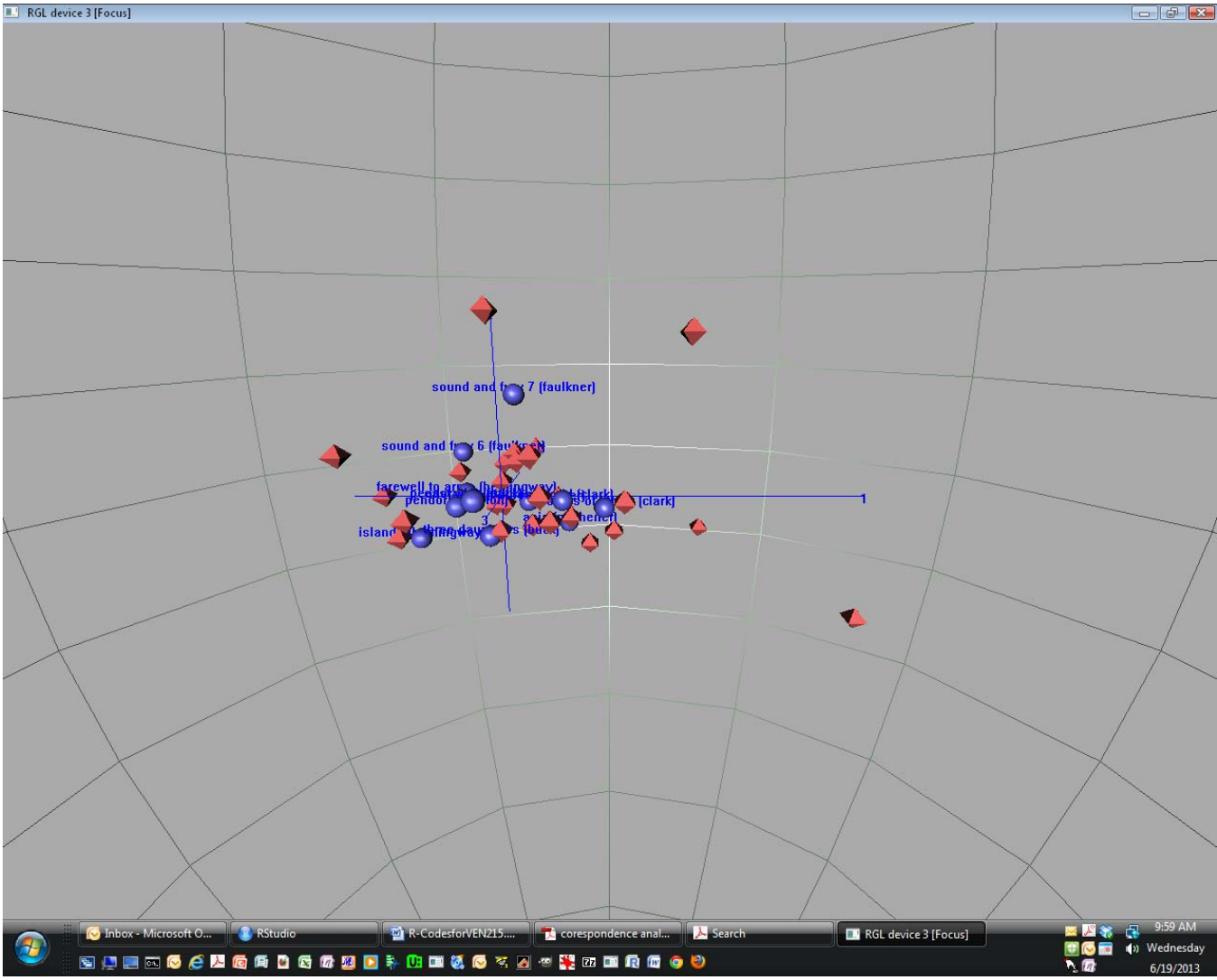


In the above code please note how I broke the title into TWO lines using \n

7. I then did a 3D plot by typing AFTER I had loaded the rgl package

plot3d.ca(ca, dim = c(1,2,3), map = "symmetric", what = c("all", "all"),
main="Symmetrical 3D Correspondence plot",
mass = c(FALSE, FALSE), contrib = c("none", "none"),
labels = 2, arrows = c(FALSE, FALSE))

I then did a screen shot of the 3 D graph since I could not figure out how to save and/or copy it.


RGL device 3 [Focus]
sound and fury 7 (faulkner)
sound and fury 6 (faulkner)
[clark]
1
Inbox - Microsoft O...
RStudio
R-CodesforVEN215....
corespondence anal...
Search
RGL device 3 [Focus]
9:59 AM
Wednesday
6/19/2013

PREFERENCE MAPPING

The Torri data set also has consumer data (both Californian and Italian consumers). For the purpose of the class we will only look at the Californian consumer data which is found in a file named: torriconsFinal.csv. I imported the data set into R, as usual.

1. I then typed
str(torriconsFinal)

```
> str(torriconsFinal)
'data.frame':  106 obs. of  13 variables:
 $ Judge         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Wine.Frequency: int  2 3 3 3 2 3 2 2 1 2 ...
 $ IT.frequency  : int  3 4 1 2 3 3 4 3 3 4 ...
 $ Gender        : int  2 2 1 2 1 1 2 1 1 2 ...
 $ Age           : int  22 21 25 24 31 60 24 25 26 40 ...
 $ C_MERLOT      : int  5 8 8 4 4 4 8 5 7 7 ...
 $ C_SYRAH       : int  4 5 6 6 6 4 6 4 5 1 ...
 $ C_ZINFANDEL   : int  3 8 6 3 6 4 7 6 6 4 ...
 $ C_REFOSCO     : int  8 5 8 4 7 4 6 6 3 6 ...
 $ I_MERLOT      : int  7 4 7 8 2 3 8 7 6 5 ...
 $ I_SYRAH       : int  2 8 5 4 6 3 6 4 6 8 ...
 $ I_PRIMITIVO   : int  4 4 7 9 5 4 4 4 6 1 ...
 $ I_REFOSCO     : int  6 6 7 2 7 1 7 3 3 2 ...
```

2. Then I looked at the data set:

	Judge	Wine.Frequency	IT.frequency	Gender	Age	C_MERLOT	C_SYRAH	C_ZINFANDEL	C_REFOSCO	I_MERLOT	I_SYRAH	I_PRIMITIVO	I_REFOSCO
1	1	2	3	2	22	5	4	3	8	7	2	4	6
2	2	3	4	2	21	8	5	8	5	4	8	4	6
3	3	3	1	1	25	8	6	6	8	7	5	7	7
4	4	3	2	2	24	4	6	3	4	8	4	9	2
5	5	2	3	1	31	4	6	6	7	2	6	5	7
6	6	3	3	1	60	4	4	4	4	3	3	4	1
7	7	2	4	2	24	8	6	7	6	8	6	4	7
8	8	2	3	1	25	5	4	6	6	7	4	4	3
9	9	1	3	1	26	7	5	6	3	6	6	6	3
10	10	2	4	2	40	7	1	4	6				

3. And found that the data set needed to be transposed. To do this I installed the package named: reshape. Then I typed the following:
library(reshape)
cons.torri <- melt(torriconsFinal, id=c("Judge", "Wine.Frequency", "IT.frequency", "Gender", "Age"))

```
> library(reshape)
> cons.torri <- melt(torriconsFinal, id=c("Judge", "Wine.Frequency",
"IT.frequency", "Gender", "Age"))
```

4. And the new data set looked like this:

Judge	Wine.Frequency	IT.frequency	Gender	Age	variable	value	
1	1	2	3	2	22	C_MERLOT	5
2	2	3	4	2	21	C_MERLOT	8
3	3	3	1	1	25	C_MERLOT	8
4	4	3	2	2	24	C_MERLOT	4
5	5	2	3	1	31	C_MERLOT	4
6	6	3	3	1	60	C_MERLOT	4
7	7	2	4	2	24	C_MERLOT	8

5. Now I needed to make the judges the columns by typing **ipm.torri<-cast(cons.torri, variable~Judge)**

```
> ipm.torri<-cast(cons.torri, variable~Judge)
```

6. And part of the new data set looked like this (I cut the rest to save space):

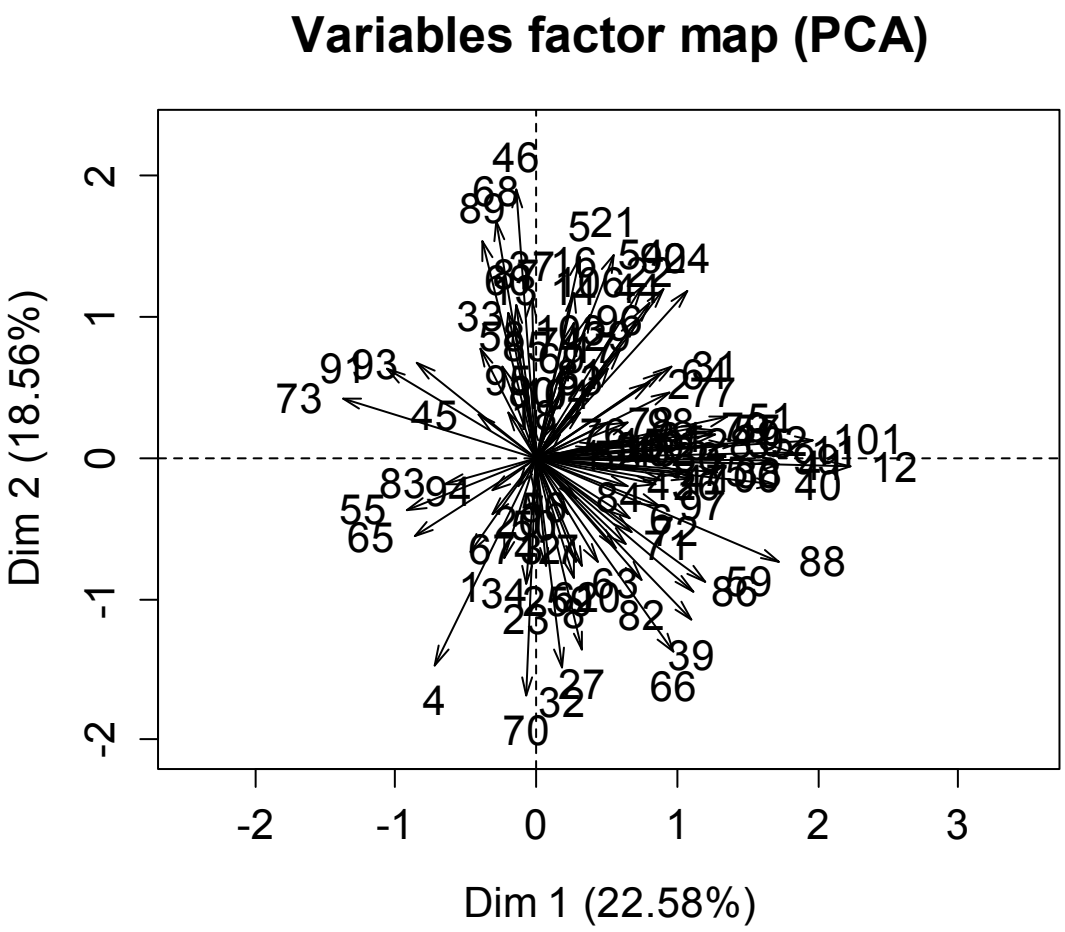
	variable	1	2	3	4	5	6	7	8
1	C_MERLOT	5	8	8	4	4	4	8	5
2	C_SYRAH	4	5	6	6	6	4	6	4
3	C_ZINFANDEL	3	8	6	3	6	4	7	6
4	C_REFOSCO	8	5	8	4	7	4	6	6
5	I_MERLOT	7	4	7	8	2	3	8	7
6	I_SYRAH	2	8	5	4	6	3	6	4
7	I_PRIMITIVO	4	4	7	9	5	4	4	4
8	I_REFOSCO	6	6	7	2	7	1	7	3

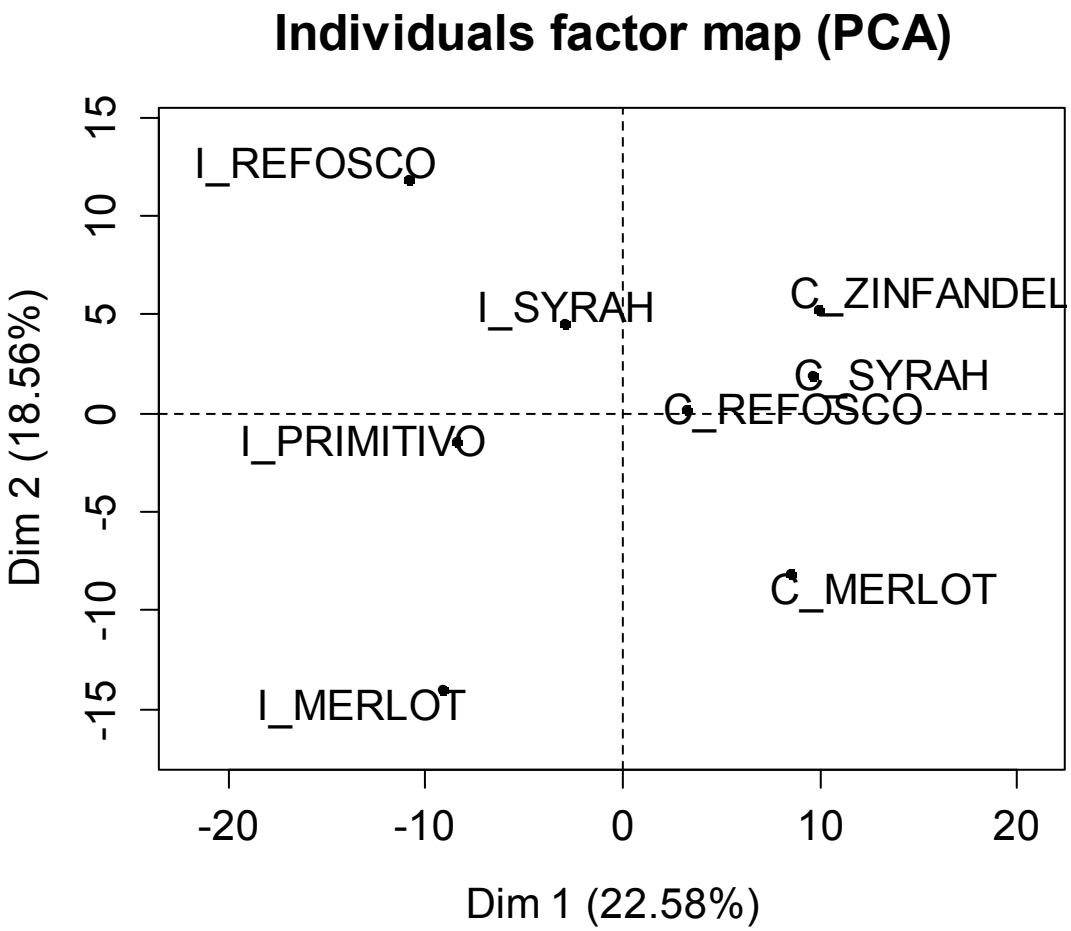
7. I wanted to rename the ‘variable’ variable name to ‘wine’ **ipm.torri <- rename(ipm.torri, c(variable="wine"))**

INTERNAL PREFERENCE MAPPING

8. Now I wanted to do the Internal Preference Map – which is a covariance PCA with the consumers as the variables. I had to tell R that wine was the row name variable.
library(SensoMineR)
row.names(ipm.torri)=ipm.torri$wine
ipm.pca = PCA(ipm.torri, scale.unit=FALSE, ncp=5, graph=TRUE)

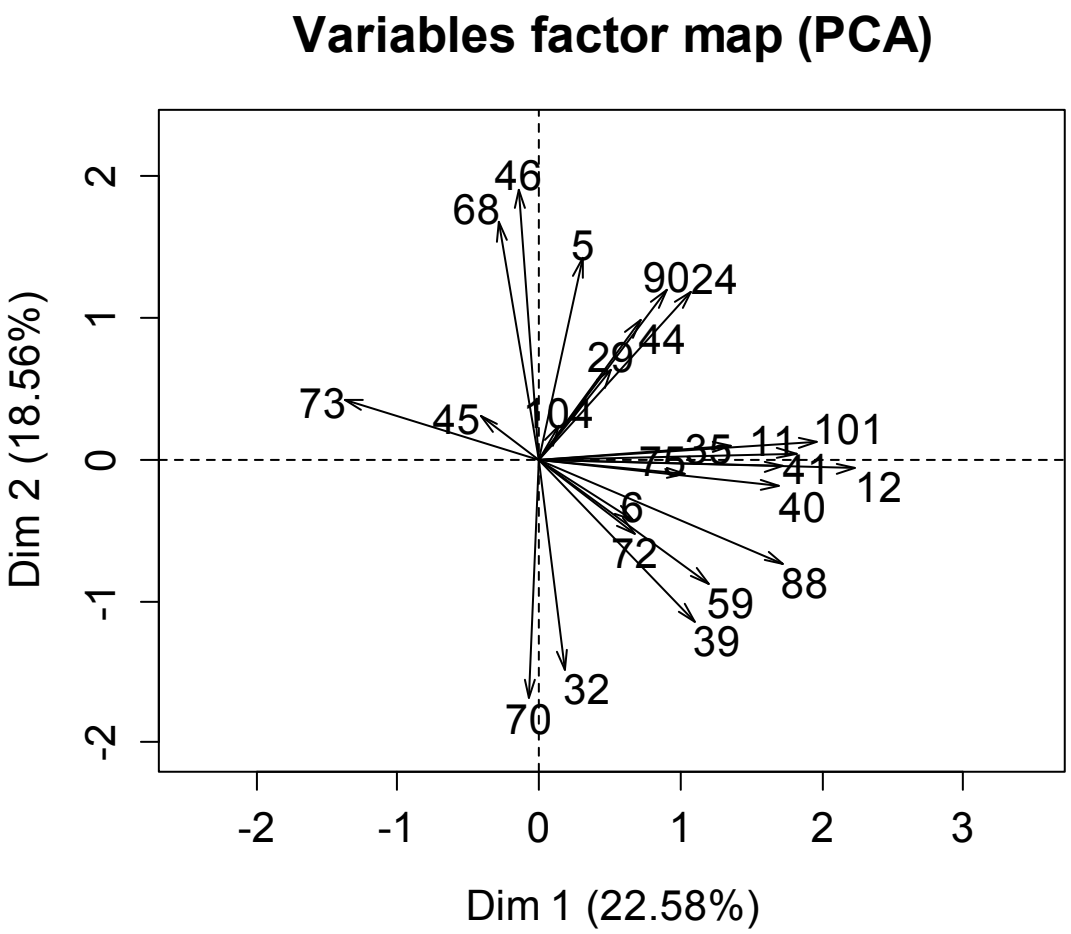
```
library(SensoMineR)
row.names(ipm.torri)=ipm.torri$wine
> ipm.pca = PCA(ipm.torri, scale.unit=FALSE, ncp=5, graph=TRUE)
```





9. I then played with the PCA plots and asked to plot the consumers with cos2 over 0.6. These are the consumers that have the most effect on the space.
plot(ipm.pca,choix="var",select="cos2 0.6")

```
> plot(ipm.pca,choix="var",select="cos2 0.6")
```


Variables factor map (PCA)
Dim 2 (18.56%)
Dim 1 (22.58%)
46
68
5
90
24
44
29
73
45
104
35
11
101
75
41
12
40
6
72
88
59
39
32
70
-2
-1
0
1
2
3

EXTERNAL PREFERENCE MAPPING

To do an external preference map I need the coordinates for the wines in the two dimensional space of interest. For this example I will use the first two dimensions from the covariance PCA that I had performed on the torri data. See in the PCA section. I had called the output da.pca

10. Here is the code as a reminder:
da.means=mtable(torriDAFinal, bycol="ProductName", firstvarcol=4)
da.pca = PCA(da.means, scale.unit=FALSE, ncp=5, graph=TRUE)

11. I then looked at the ipm.torri.data set and realized that I needed to remove column 1.

row.names	wine	1	2	3	4	5	6	7	8	9
1	C_MERLOT	C_MERLOT	5	8	8	4	4	4	8	5
2	C_SYRAH	C_SYRAH	4	5	6	6	6	4	6	4
3	C_ZINFANDEL	C_ZINFANDEL	3	8	6	3	6	4	7	6
4	C_REFOSCO	C_REFOSCO	8	5	8	4	7	4	6	6
5	I_MERLOT	I_MERLOT	7	4	7	8	2	3	8	7
6	I_SYRAH	I_SYRAH	2	8	5	4	6	3	6	4
7	I_PRIMITIVO	I_PRIMITIVO	4	4	7	9	5	4	4	4
8	I_REFOSCO	I_REFOSCO	6	6	7	2	7	1	7	3

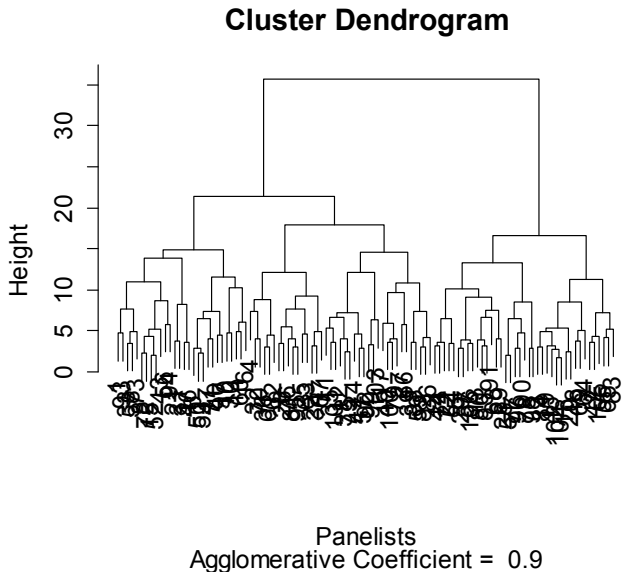
12. I removed that column by typing:
ipm.torri=ipm.torri[,-1]

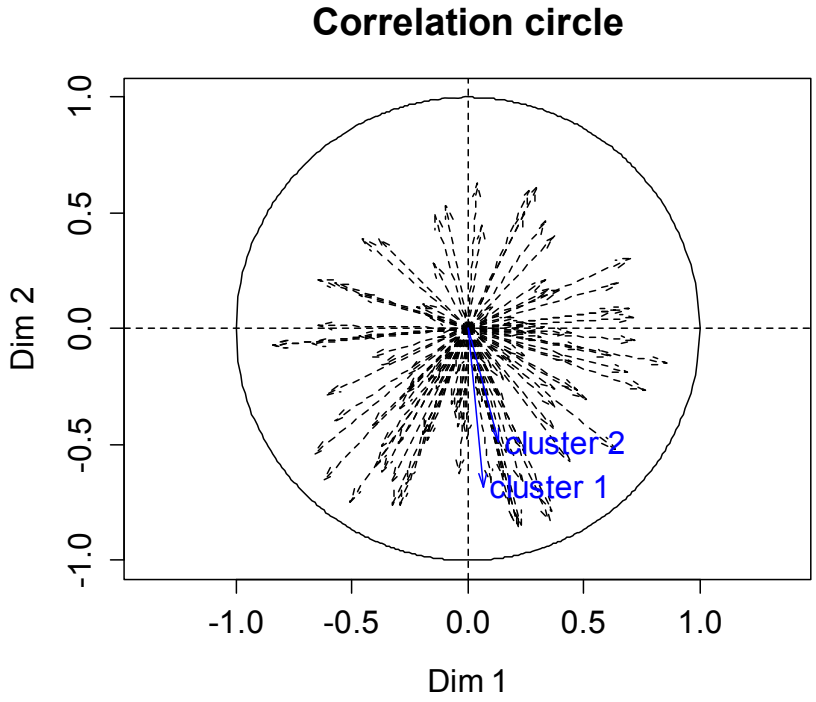
13. And then ipm.torri looked like this:

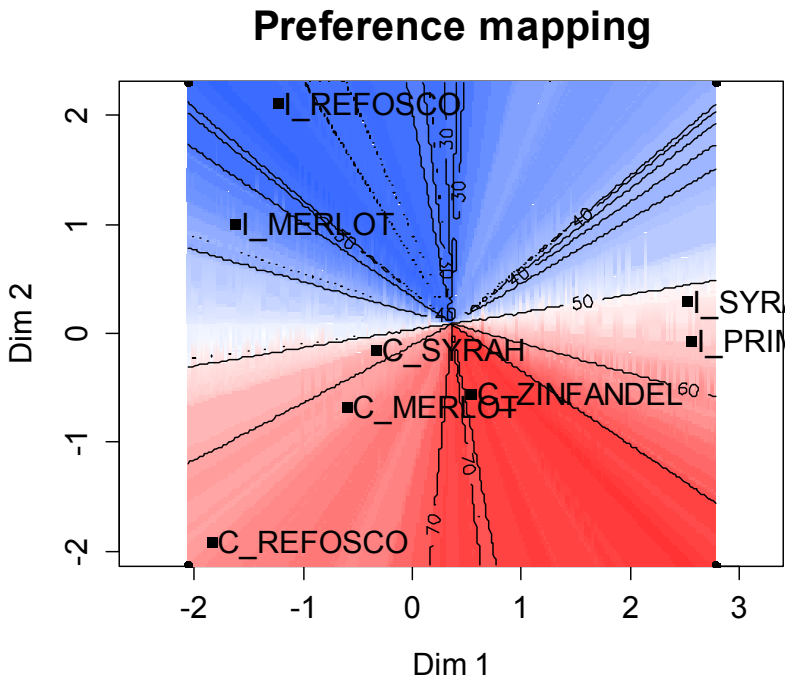
	row.names	1	2	3	4	5	6	7	8	9
1	C_MERLOT	5	8	8	4	4	4	8	5	7
2	C_SYRAH	4	5	6	6	6	4	6	4	5
3	C_ZINFANDEL	3	8	6	3	6	4	7	6	6
4	C_REFOSCO	8	5	8	4	7	4	6	6	3
5	I_MERLOT	7	4	7	8	2	3	8	7	6
6	I_SYRAH	2	8	5	4	6	3	6	4	6
7	I_PRIMITIVO	4	4	7	9	5	4	4	4	6
8	I_REFOSCO	6	6	7	2	7	1	7	3	3

14. Then I ran the External Preference Map with a VECTOR model. Since I only had 8 wines that is the ONLY model I could legally create. I specified a vector model by saying regmod=2. If is had had more wines I could have specified regmod=1 (quadratic, AND THE DEFAULT – be VERY careful), regmod=3 (circular) and regmod=4 (elliptical).
epm.torri <- carto(da.pcaindcoord[,1:2], ipm.torri,regmod = 2)

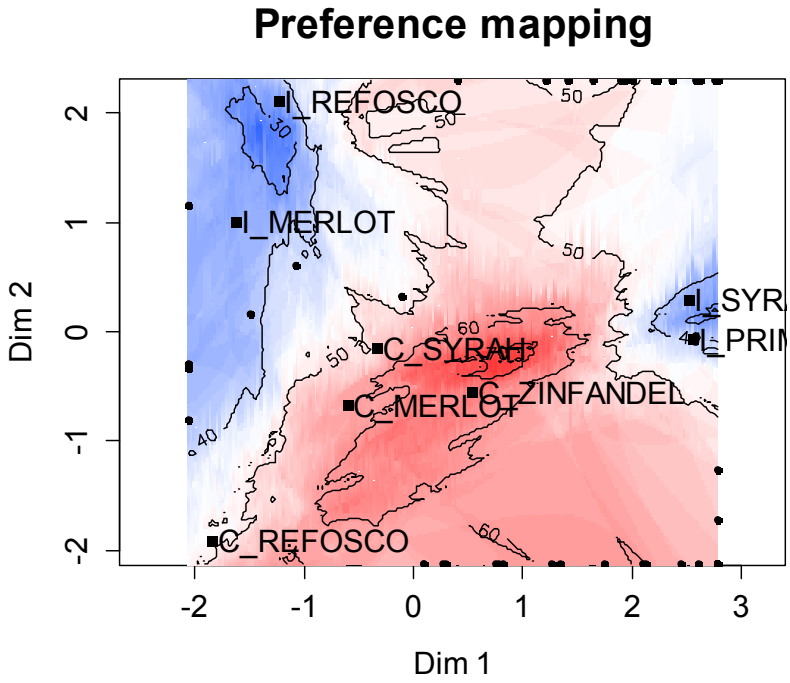
```
> epm.torri <- carto(da.pca$ind$coord[,1:2], ipm.torri,regmod = 2)
```







15. When I re-ran it with regmod=1 (illegally since I do NOT have enough wines) I got:



PARTIAL LEAST SQUARES REGRESSION (PLSR)

1. To do PLS I used the pls package.
   library(pls)

```
> library(pls)
```

Since I did not have a chemical-sensory data set handy, I decided to do PLS on the torri sensory and consumer data. First I had to combine the mean descriptive data and the short-and-fat consumer data into the same data set. I chose to do this in EXCEL and then loaded the data set. In this case I did it in the source but I could have done it the normal way too.

torrisenscons =read.csv("U:/Dropbox/MyFiles/VEN215/torrisenscons.csv", header=T, quote="")

	Wine	Red_berry	Dark_berry	Jam	Dried_fruit	Artificial_frui	Chocolate
1	C_MERLOT	2.464286	3.047619	1.3714286	1.857143	0.7761905	1.1904762
2	C_REFOSCO	2.466667	2.461905	1.0309524	1.423810	0.9238095	1.9976190
3	C_SYRAH	2.464286	2.933333	1.7452381	1.683333	0.8833333	1.4190476
4	C_ZINFANDEL	3.076190	3.059524	1.9785714	2.061905	0.8642857	0.9690476
5	I_MERLOT	2.790476	2.350000	0.8428571	1.850000	0.5738095	0.7833333
6	I_PRIMITIVO	3.850000	3.380952	3.6119048	1.435714	2.1904762	1.3809524
7	I_REFOSCO	2.478571	3.007143	1.5357143	1.873810	1.1095238	0.8095238
8	I_SYRAH	3.173810	4.483333	3.0976190	2.164286		

2. Now I had to create the subsets for the PLS regression. One for sensory (sens) and one for the consumers (cons)
   sens=as.matrix(pls.sc[,1:20])
   cons=as.matrix(pls.sc[,22:127])

row.names	Red_berry	Dark_berry	Jam	Dried_fruit	Artificial_frui
1	C_MERLOT	2.464286	3.047619	1.3714286	1.857143
2	C_REFOSCO	2.466667	2.461905	1.0309524	1.423810
3	C_SYRAH	2.464286	2.933333	1.7452381	1.683333
4	C_ZINFANDEL	3.076190	3.059524	1.9785714	2.061905
5	I_MERLOT	2.790476	2.350000	0.8428571	1.850000
6	I_PRIMITIVO	3.850000	3.380952	3.6119048	1.435714
7	I_REFOSCO	2.478571	3.007143	1.5357143	1.873810
8	I_SYRAH	3.173810	4.483333	3.0976190	2.164286

	row.names	X2	X3	X4	X5	X6	X7	X8	X9	X10
1	C_MERLOT	8	8	4	4	4	8	5	7	7
2	C_REFOSCO	5	8	4	7	4	6	6	3	6
3	C_SYRAH	5	6	6	6	4	6	4	5	1
4	C_ZINFANDEL	8	6	3	6	4	7	6	6	4
5	I_MERLOT	4	7	8	2	3	8	7	6	5
6	I_PRIMITIVO	4	7	9	5	4	4	4	6	1
7	I_REFOSCO	6	7	2	7	1	7	3	3	2
8	I_SYRAH	8	5	4	6	3	6			

I had lost consumer 1 (X1) when I did the data manipulation in EXCEL – so my first consumer is X2. The PLS explains 62.03% of the sensory (X) data set in the first tw dimensions and then 6.7% of consumer 2, 48.5% of consumer 3, 81,5% of consumer 6 etc..

3. And then I ran the PLS
   pls.torri=plsr(cons~sens, data=pls.sc, scale=TRUE)

```
> sens=as.matrix(pls.sc[,1:20])
> cons=as.matrix(pls.sc[,22:126])
> pls.torri=plsr(cons~sens, data=pls.sc, scale=TRUE)
```

4. To see some of the output I asked for a summary:
   summary(pls.torri)

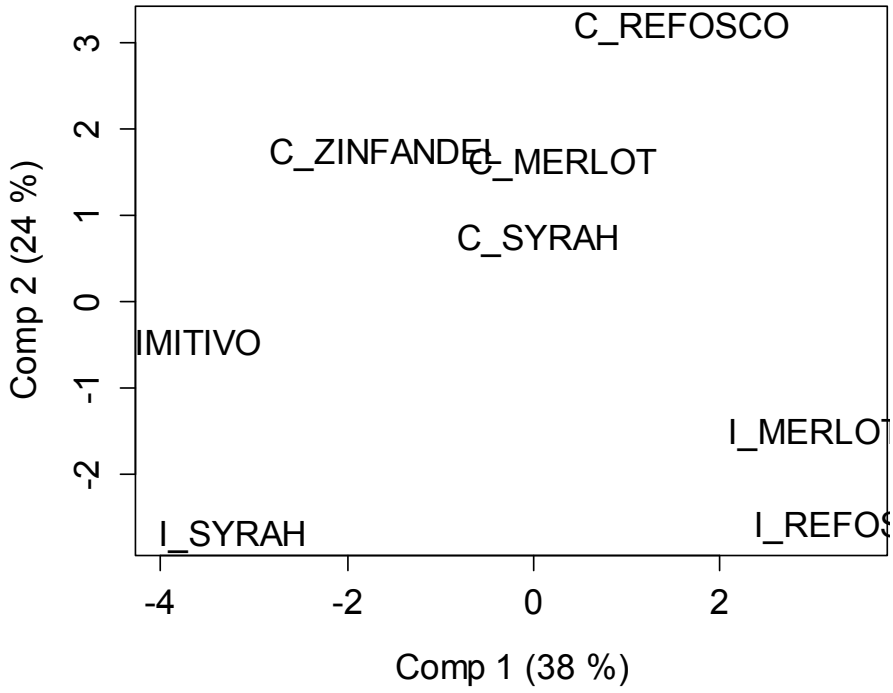
```
Data:   X dimension: 8 20
        Y dimension: 8 105
Fit method: kernelpls
Number of components considered: 7
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X     3.798e+01  62.0267   74.354    83.57    93.27    98.52      100
X2    6.396e+00   6.6706   10.671    78.69    81.87    81.87      100
X3    2.434e+01  48.5102   67.244    80.49    80.70    81.64      100
X4    7.468e+00   8.3853   39.215    68.42    74.95    81.07      100
X5    1.450e+00   3.7963   78.565    80.32    83.90    94.17      100
X6    2.599e+01  81.5187   86.565    86.61    86.82    94.87      100
X7    4.456e+01  44.5800   51.079    97.22    97.23    99.62      100
X8    5.853e+00  27.1462   59.846    65.52    74.07    74.22      100
```

5. Then I asked for the loadings
   pls.torri$loadings

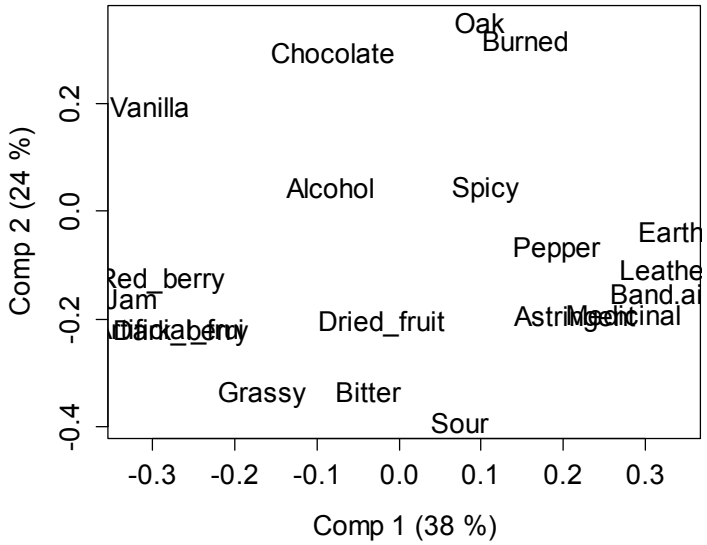
```
Loadings:
               Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7
Red_berry      -0.290 -0.124  0.202 -0.152          0.337  0.311
Dark_berry     -0.265 -0.223 -0.157  0.155 -0.158 -0.118 -0.482
Jam            -0.327 -0.159        -0.138          0.127 -0.174
Dried_fruit           -0.206         0.648
Artificial_frui -0.279 -0.220        -0.196 -0.189        -0.329
Chocolate              0.297 -0.156 -0.399 -0.267 -0.201 -0.140
Vanilla        -0.303  0.196 -0.122 -0.116          0.232
Oak                    0.351         0.332 -0.101  0.310 -0.177
Burned          0.153  0.319        -0.201 -0.335 -0.185
```

6. I then did a score plot of the wines:
   plot(pls.torri, plottype = "scores", labels=rownames(pls.sc), comps = 1:2)

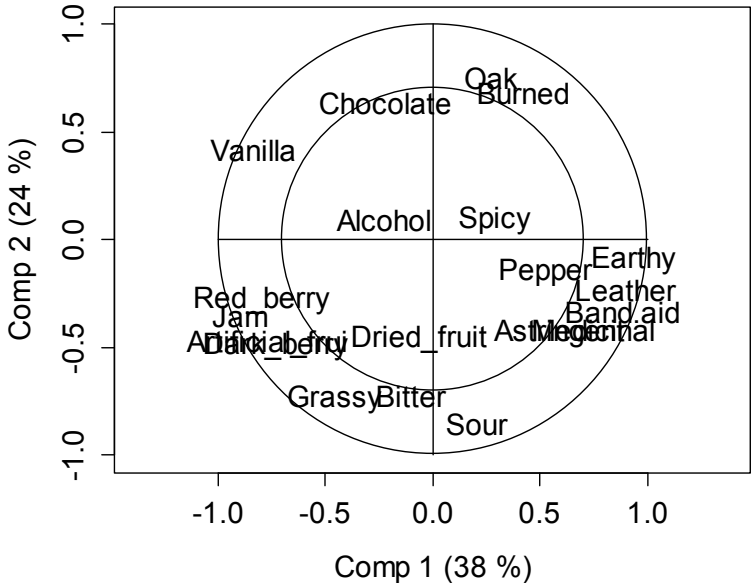
Please note that the percentage explained is for the SENSORY (X) data set



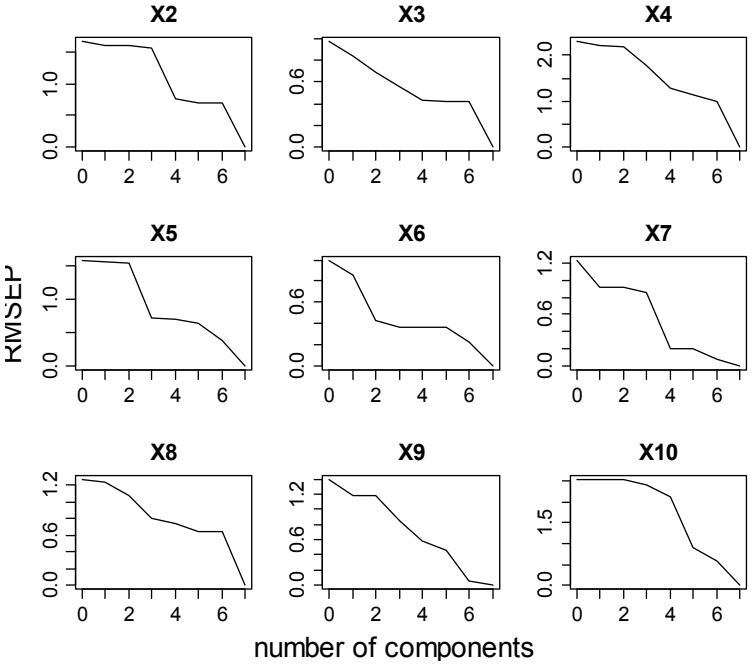
7. and a loadings plot
loadingplot(pls.torri, comps = 1:2, labels=rownames(pls.torri$loadings), scatter=TRUE)



8. and a correlation loadings plot
corrplot(pls.torri, comps = 1:2, labels=rownames(pls.torri$loadings))

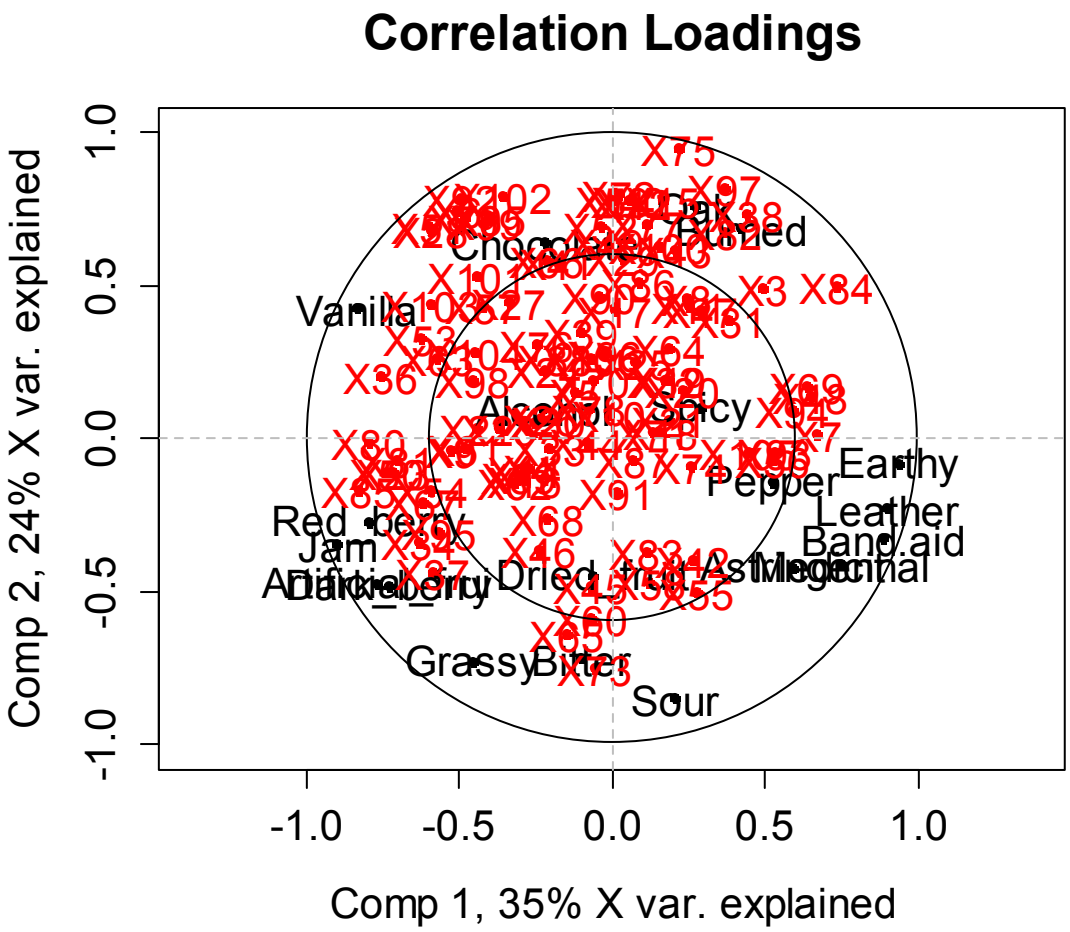


9. Then I plotted a set of validation plots for RMSEP (I could also have asked for MSEP and R2). I am only showing the first 10 consumers here.
validationplot(pls.torri, val.type="RMSEP")

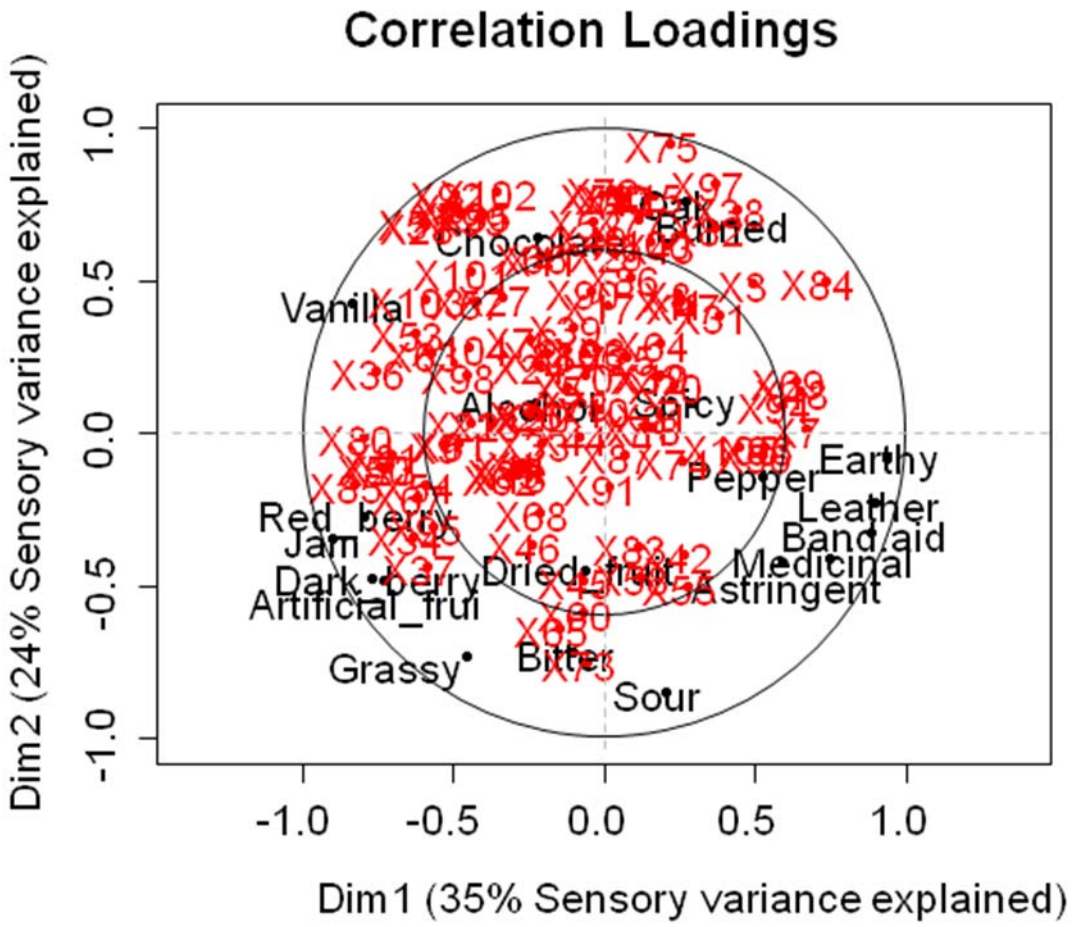


10. Next I wanted to do a plot with loadings and consumers. To do this Helene Hopfer provided the following code – with comments indicated by #.

```
# defining S as the score matrix (i.e. the one with the wines and their position on the comps) for the first 2 comps
S <- scores(pls.torri)
cl1 <- cor(pls.torri$model[,2], S)    # correlation between predicting variables and scores
cl2 <- cor(pls.torri$model[,1], S)    # correlation matrix between predicted variables and scores
plot(cl1, xlim=c(-1,1), ylim=c(-1,1), pch=20, main="Correlation Loadings",
     xlab="Comp 1, 35% X var. explained", ylab="Comp 2, 24% X var. explained", asp=1)
points(cl2, col="red", pch=20)
text(cl1, labels=row.names(cl1))
text(cl2, labels=row.names(cl2), col="red")
symbols(x=0, y=0, add=T, circles=(.6), lty=2, inches=F)
symbols(x=0, y=0, add=T, circles=(1), lty=2, inches=F)
abline(h=0, v=0, lty=2, col="grey")
```



11. I then played a little with the graph in PowerPoint to make it prettier:



PRINCIPAL COMPONENT REGRESSION

1. I then decided to do a PCR on the same data using the pls package.

library(pls)
pcr.torri=pcr(cons~sens, data=pls.sc, scale=TRUE)
summary(pcr.torri)

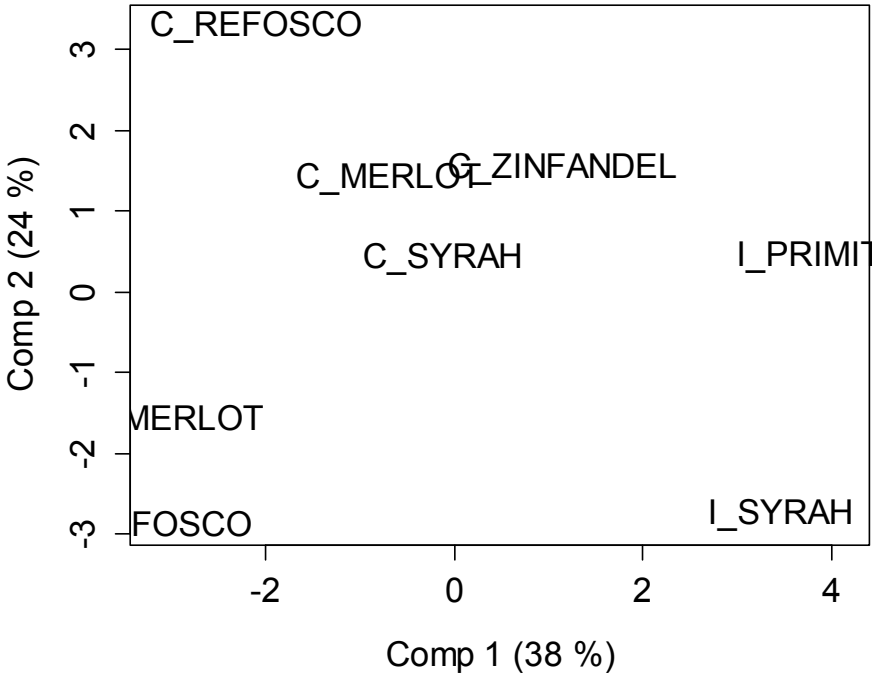
```
> summary(pcr.torri)
Data:   X dimension: 8 20
        Y dimension: 8 105
Fit method: svdpc
Number of components considered: 7
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X     3.817e+01  62.3989   74.935   84.673    93.43    98.53      100
X2    5.117e+00   5.2570    5.657   22.005    81.84    81.98      100
X3    2.932e+01  55.6561   67.093   68.234    81.64    81.97      100
X4    8.692e+00   8.7784   29.922   39.462    74.14    80.15      100
X5    1.169e+00   2.8623   78.795   78.795    79.54    92.98      100
X6    1.841e+01  81.0289   83.549   83.698    83.73    94.44      100
X7    4.635e+01  47.9212   60.747   67.467    95.53    99.80      100
X8    8.974e+00  28.8172   54.621   73.231    73.53    73.61      100
```

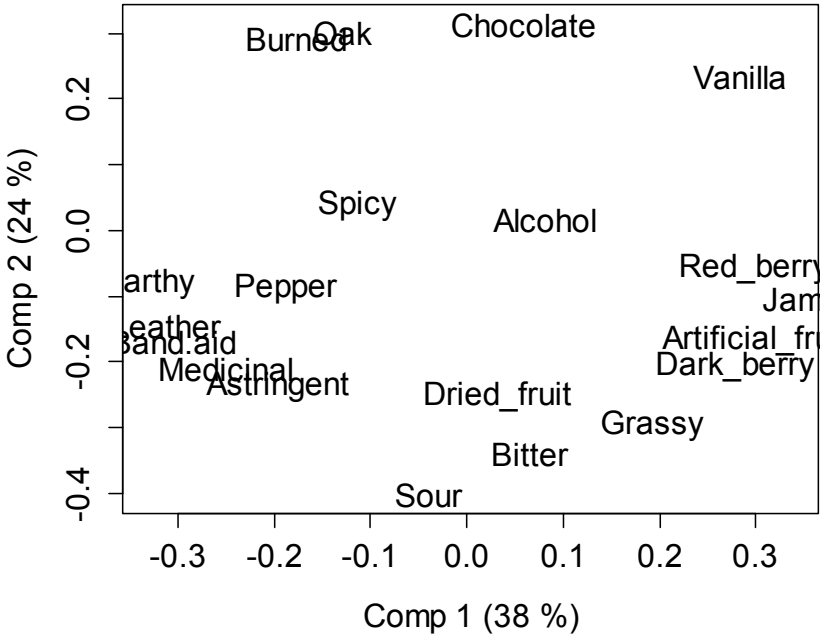
Please note differences with PLS

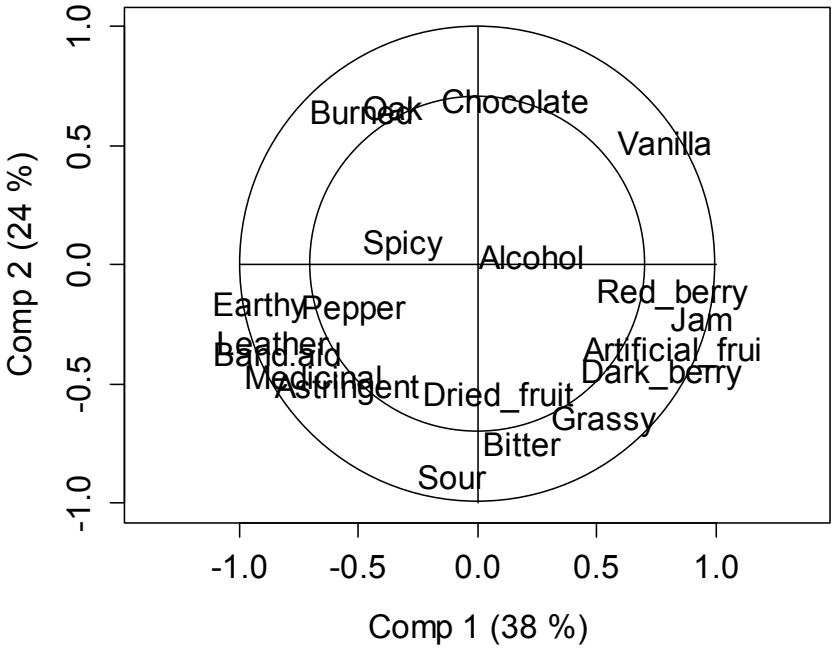
2. I then did most of the same plots as with the PLS

plot(pcr.torri, plottype = "scores", labels=rownames(pls.sc), comps = 1:2)
loadingplot(pcr.torri, comps = 1:2, labels=rownames(pcr.torri$loadings), scatter=TRUE)
corrplot(pcr.torri, comps = 1:2, labels=rownames(pcr.torri$loadings))

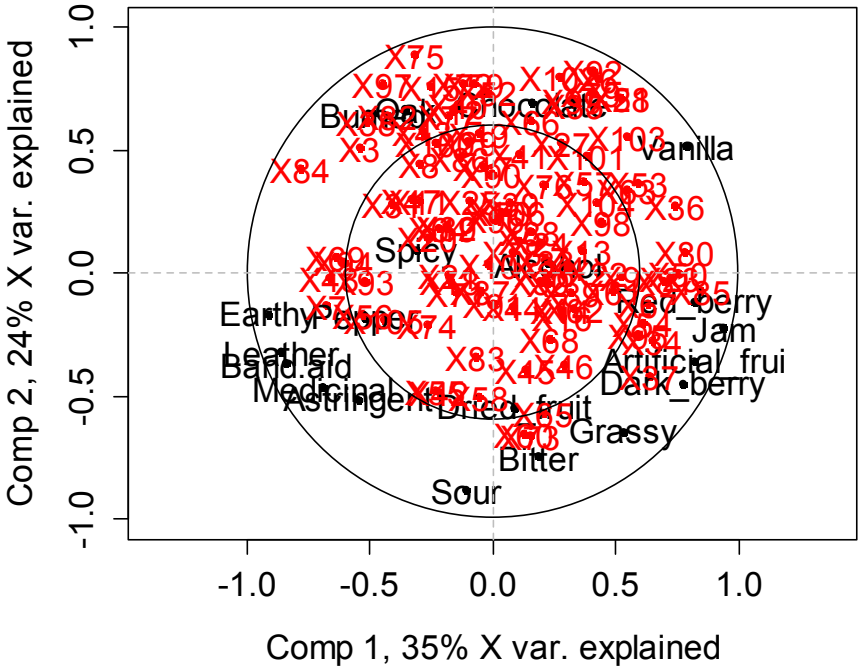
 # defining S as the score matrix (i.e. the one with the wines and their position on the comps) for the first 2 comps
S <- scores(pcr.torri)
cl1 <- cor(pcr.torri$model[,2], S) # correlation between predicting variables and scores
cl2 <- cor(pcr.torri$model[,1], S) # correlation matrix between predicted variables and scores
plot(cl1, xlim=c(-1,1), ylim=c(-1,1), pch=20, main="Correlation Loadings",
xlab="Comp 1, 35% X var. explained", ylab="Comp 2, 24% X var. explained", asp=1)
points(cl2, col="red", pch=20)
text(cl1, labels=row.names(cl1))
text(cl2, labels=row.names(cl2), col="red")
symbols(x=0, y=0, add=T, circles=(.6), lty=2, inches=F)
symbols(x=0, y=0, add=T, circles=(1), lty=2, inches=F)
abline(h=0, v=0, lty=2, col="grey")


C_REFOSCO
C_MERLOT
C_ZINFANDEL
C_SYRAH
I_PRIMIT
MERLOT
FOSCO
I_SYRAH
Comp 2 (24 %)
Comp 1 (38 %)
3
2
1
0
-1
-2
-3
-2
0
2
4


Burned
Oak
Chocolate
Vanilla
Spicy
Alcohol
arthy
Pepper
Red_berry
Jam
eather
and.aid
Artificial_fr
Dark_berry
Medicinal
Astringent
Dried_fruit
Grassy
Bitter
Sour
Comp 2 (24 %)
Comp 1 (38 %)
0.2
0.0
-0.2
-0.4
-0.3
-0.2
-0.1
0.0
0.1
0.2
0.3



Correlation Loadings



MULTIFACTOR ANALYSIS (MFA)

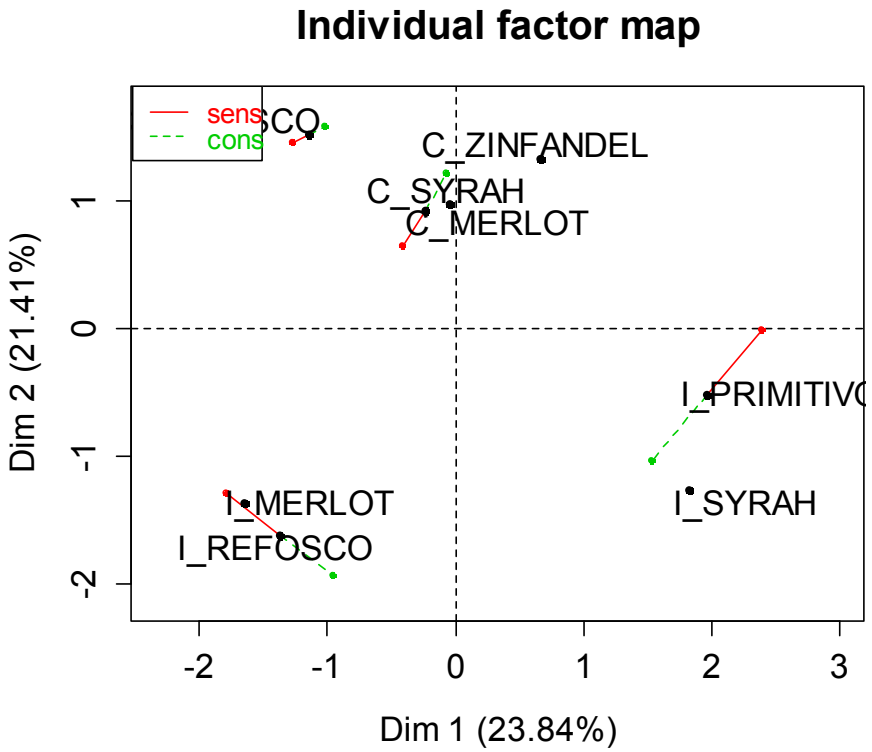
1. I did the MFA on the same data I had used for the PLS. I combined the mean descriptive data and the short-and-fat consumer data into the same data set. I chose to do this in EXCEL and then loaded the data set. As before I made wine the row name identifier and then I removed the wine column from the data.
   torrisenscons =read.csv("U:/Dropbox/MyFiles/VEN215/torrisenscons.csv", header=T, quote="")
   mfa.torri=torrisenscons
   row.names(mfa.torri)=mfa.torri$wine
   mfa.torri=mfa.torri[,-1]

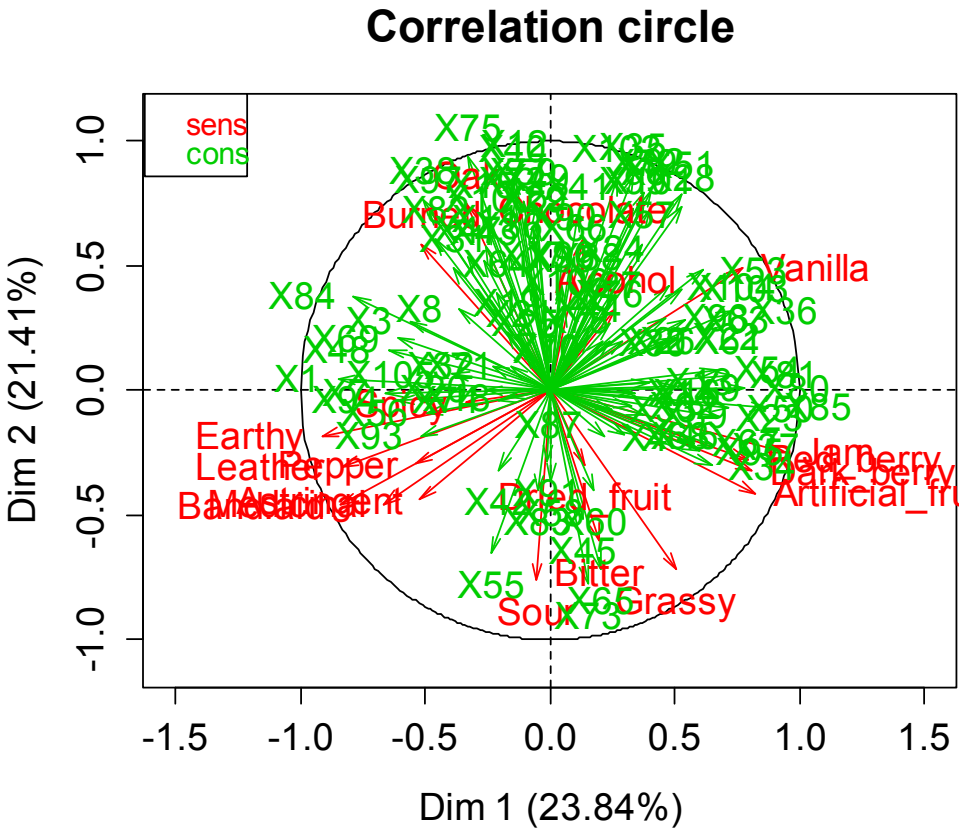
2. Then I ran the MFA. I created two groups from the data – the sensory group (20 variables) and the cosumer group (106 variables). I also asked for a summary and a bar graph of the eigenvalues. The plots were automatically plotted by my request for an MFA.
   mfa.sc <- MFA(mfa.torri, group=c(20,106), ncp=5, name.group=c("sens","cons"))
   summary(mfa.sc)
   barplot(mfa.sc$eig[,1],main="Eigenvalues",names.arg=1:nrow(mfa.sc$eig))

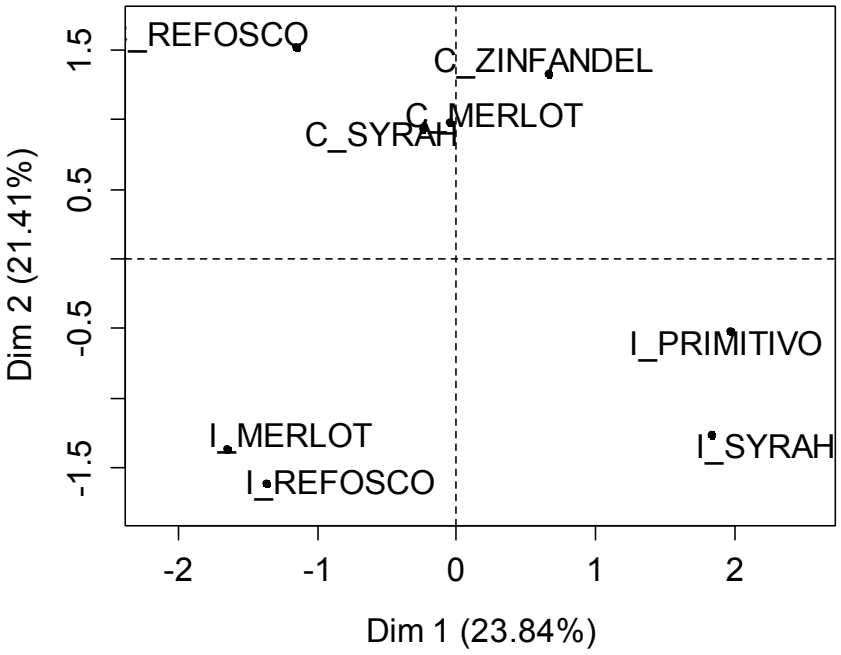
```
> mfa.sc <- MFA(mfa.torri, group=c(20,106),  ncp=5,
name.group=c("sens","cons"))
> summary(mfa.sc)
Call:
MFA(mfa.torri, group = c(20, 106), ncp = 5, name.group = c("sens",
     "cons"))


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
Variance               1.701   1.527   1.122   0.860   0.828   0.693   0.403
% of var.             23.841  21.405  15.733  12.053  11.609   9.716   5.643
Cumulative % of var.  23.841  45.246  60.979  73.032  84.641  94.357 100.000

Groups
                  Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
sens            |  0.993 58.365  0.598 |  0.578 37.820  0.202 |  0.324 28.900
cons            |  0.708 41.635  0.156 |  0.950 62.180  0.281 |  0.798 71.100
                  cos2
sens             0.064 |
cons             0.199 |

Individuals
                  Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
C_MERLOT        | -0.239  0.420  0.012 |  0.931  7.097  0.184 | -1.213 16.386
C_REFOSCO       | -1.149  9.701  0.176 |  1.521 18.942  0.308 |  0.440  2.157
C_SYRAH         | -0.046  0.016  0.000 |  0.977  7.811  0.158 |  0.988 10.875
C_ZINFANDEL     |  0.661  3.214  0.070 |  1.327 14.425  0.281 |  0.038  0.016
I_MERLOT        | -1.649 19.990  0.331 | -1.367 15.294  0.227 | -1.728 33.246
I_PRIMITIVO     |  1.962 28.295  0.480 | -0.520  2.212  0.034 | -0.659  4.839
I_REFOSCO       | -1.369 13.778  0.230 | -1.611 21.245  0.318 |  1.633 29.689
I_SYRAH         |  1.829 24.587  0.412 | -1.259 12.973  0.195 |  0.501  2.791
                  cos2
C_MERLOT         0.312 |
C_REFOSCO        0.026 |
C_SYRAH          0.161 |
```

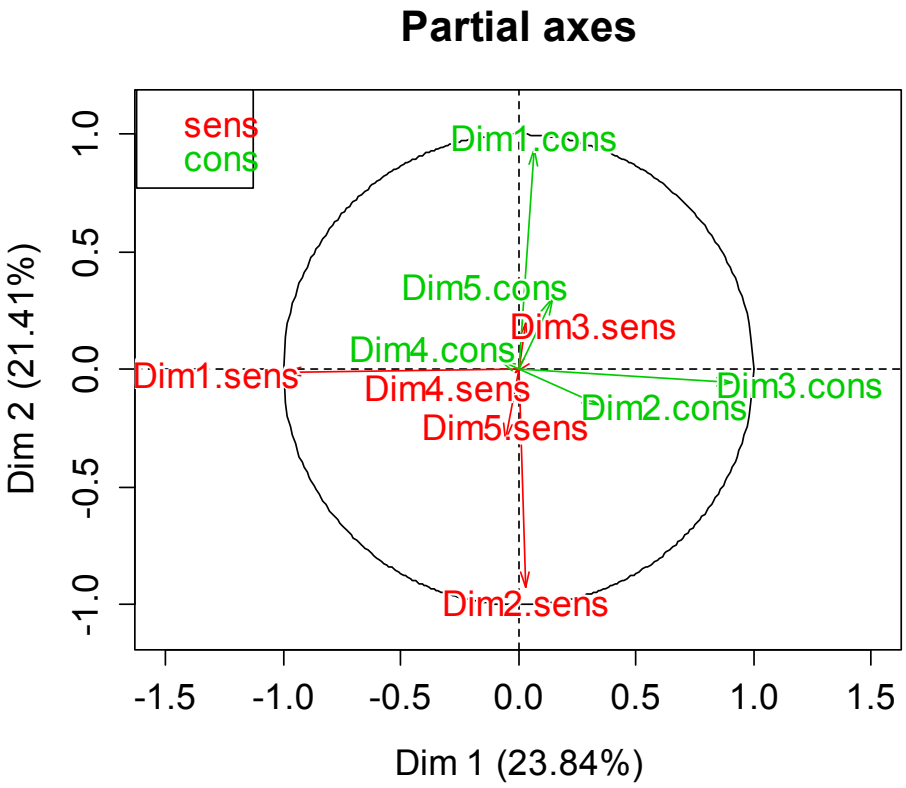
```
C_ZINFANDEL       0.000 |
I_MERLOT          0.363 |
I_PRIMITIVO       0.054 |
I_REFOSCO         0.327 |
I_SYRAH           0.031 |

Continuous variables (the 10 first)
                  Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
Red_berry       |  0.793  4.845  0.629 | -0.275  0.647  0.075 | -0.272  0.861
Dark_berry      |  0.809  5.044  0.655 | -0.322  0.888  0.104 |  0.244  0.696
Jam             |  0.941  6.827  0.886 | -0.247  0.521  0.061 |  0.105  0.128
Dried_fruit     |  0.140  0.151  0.020 | -0.300  0.773  0.090 |  0.055  0.035
Artificial_frui |  0.822  5.208  0.676 | -0.411  1.448  0.169 |  0.176  0.360
Chocolate       |  0.128  0.127  0.016 |  0.607  3.162  0.369 |  0.170  0.336
Vanilla         |  0.772  4.595  0.597 |  0.493  2.083  0.243 |  0.112  0.145
Oak             | -0.339  0.885  0.115 |  0.747  4.792  0.559 |  0.074  0.065
Burned          | -0.517  2.060  0.267 |  0.588  2.969  0.346 |  0.082  0.078
Leather         | -0.833  5.349  0.695 | -0.304  0.795  0.093 |  0.389  1.765
                 cos2
Red_berry       0.074 |
Dark_berry      0.060 |
Jam             0.011 |
Dried_fruit     0.003 |
Artificial_frui 0.031 |
Chocolate       0.029 |
Vanilla         0.012 |
Oak             0.006 |
Burned          0.007 |
Leather         0.151 |
```

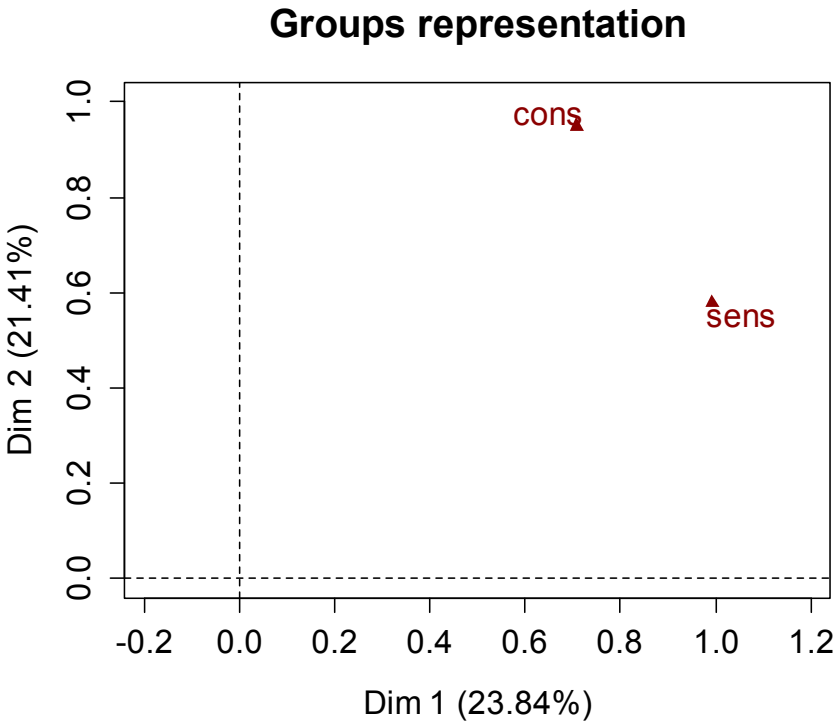
```
> barplot(mfa.sc$eig[,1],main="Eigenvalues",names.arg=1:nrow(mfa.sc$eig))
```

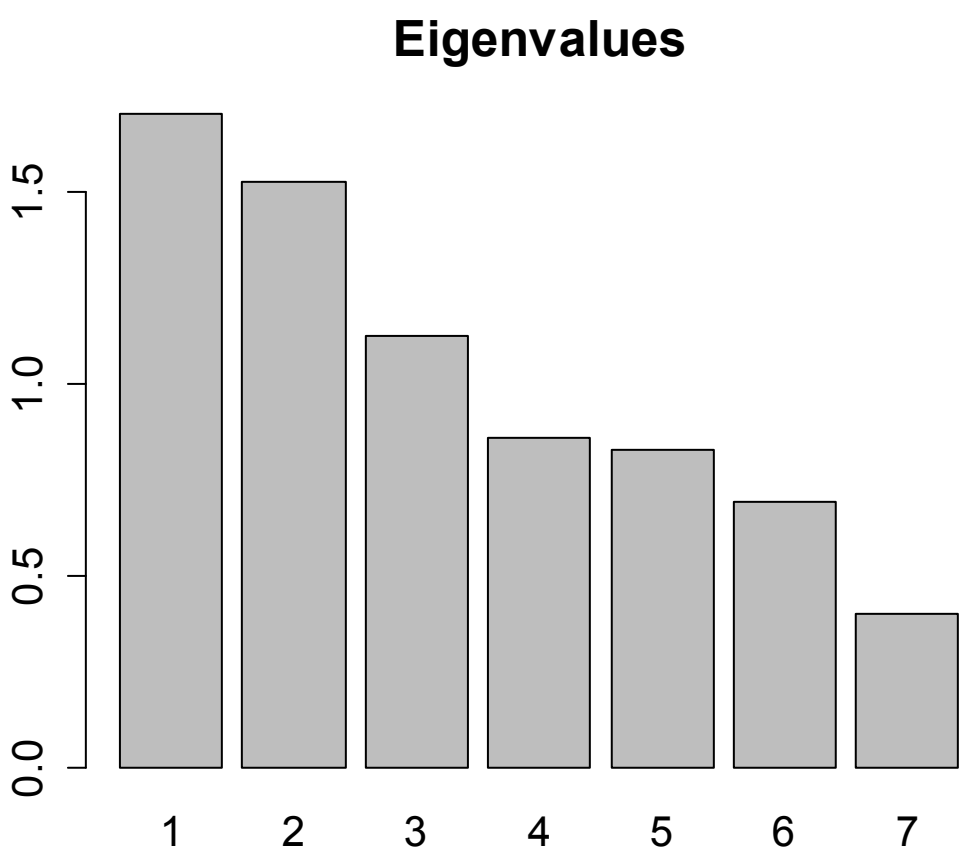



Correlation circle
sens
cons
X75
Vanilla
X84
X8
X3
Earthy
X93
Leather
Pepper
Artificial_fr
Bitter
X55
X65
Grassy
Soux73
Dim 2 (21.41%)
Dim 1 (23.84%)
1.0
0.5
0.0
-0.5
-1.0
-1.5


Individual factor map
_REFOSCO
C_ZINFANDEL
C_MERLOT
C_SYRAH
I_PRIMITIVO
I_MERLOT
I_SYRAH
I_REFOSCO
Dim 2 (21.41%)
Dim 1 (23.84%)
1.5
0.5
-0.5
-1.5
-2
-1
0
1
2


Partial axes
sens
cons
Dim1.cons
Dim5.cons
Dim3.sens
Dim4.cons
Dim1.sens
Dim4.sens
Dim2.cons
Dim3.cons
Dim5.sens
Dim2.sens
Dim 2 (21.41%)
Dim 1 (23.84%)
1.0
0.5
0.0
-0.5
-1.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


Groups representation
cons
sens
Dim 2 (21.41%)
Dim 1 (23.84%)
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Eigenvalues
0.0
0.5
1.0
1.5
1
2
3
4
5
6
7

GENERALIZED PROCUSTES ANALYSIS (GPA)

For the GPA I used the exact same data set as for the MFA but renamed it. I then asked to do the GPA on the 20 sensory and 106 conumer variables.Then I asked for a summary and then for all the possible output.

torrisenscons =read.csv("U:/Dropbox/MyFiles/VEN215/torrisenscons.csv", header=T, quote="")
gpa.torri=torrisenscons
row.names(gpa.torri)=gpa.torri$wine
gpa.torri=gpa.torri[,-1]
gpa.sc <- GPA(gpa.torri, group=c(20,106), name.group=c("sens","cons"))

summary(gpa.sc)
list(gpa.sc$RV)

```
> summary(gpa.sc)
             Length Class  Mode
RV             4    -none- numeric
RVs            4    -none- numeric
simi           4    -none- numeric
scaling        2    -none- numeric
dep          112    -none- numeric
consensus     56    -none- numeric
Xfin         112    -none- numeric
correlations   2    -none- list
PANOVA         6    -none- list
```

```
> list(gpa.sc$RV)
[[1]]
          sens      cons
sens 1.0000000 0.6674211
cons 0.6674211 1.0000000
```

These are the RV coefficients

list(gpa.sc$RVs)

These are the standardized RV coefficients

```
> list(gpa.sc$RVs)
[[1]]
          sens      cons
sens 5.1916706 0.9285894
cons 0.9285894 5.0635496
```

list(gpa.sc$simi)

These are the Procrustes similarity indexes between partial configuration

```
> list(gpa.sc$simi)
[[1]]
          sens      cons
sens 1.0000000 0.8712266
cons 0.8712266 1.0000000
```

list(gpa.sc$scaling)

```
> list(gpa.sc$scaling)
[[1]]
         [,1]
[1,] 5.451126
[2,] 0.713132
```

These are the isotropic scaling factors

list(gpa.sc$consensus)

This is the consensus configuration

```
> list(gpa.sc$consensus)
[[1]]
                  [,1]        [,2]        [,3]        [,4]         [,5]
C_MERLOT      0.06879595 -0.14047923 -0.12968827  0.04226215  0.054151844
C_REFOSCO     0.22594124 -0.20577485  0.11610262 -0.18513842 -0.019218050
C_SYRAH       0.06258607 -0.05653560  0.12171919  0.11130269 -0.057291316
C_ZINFANDEL  -0.06053748 -0.12965726 -0.02886161  0.20357923 -0.007927214
I_MERLOT      0.19664200  0.16253099 -0.21420822 -0.04930469  0.057856607
I_PRIMITIVO  -0.30077968  0.02375829 -0.08633903 -0.09980224 -0.190152195
I_REFOSCO     0.11377428  0.31676285  0.13632078  0.04031760 -0.035769533
I_SYRAH      -0.30642238  0.02939481  0.08495454 -0.06321633  0.198349858
                  [,6]         [,7]
C_MERLOT      0.038086730 -0.142874698
C_REFOSCO    -0.066757340  0.018654986
C_SYRAH       0.191900105  0.041749235
C_ZINFANDEL  -0.145251321  0.056952267
I_MERLOT      0.029989682  0.080879620
I_PRIMITIVO   0.004789097 -0.006540405
I_REFOSCO    -0.068927433 -0.062785611
I_SYRAH       0.016170481  0.013964606
```

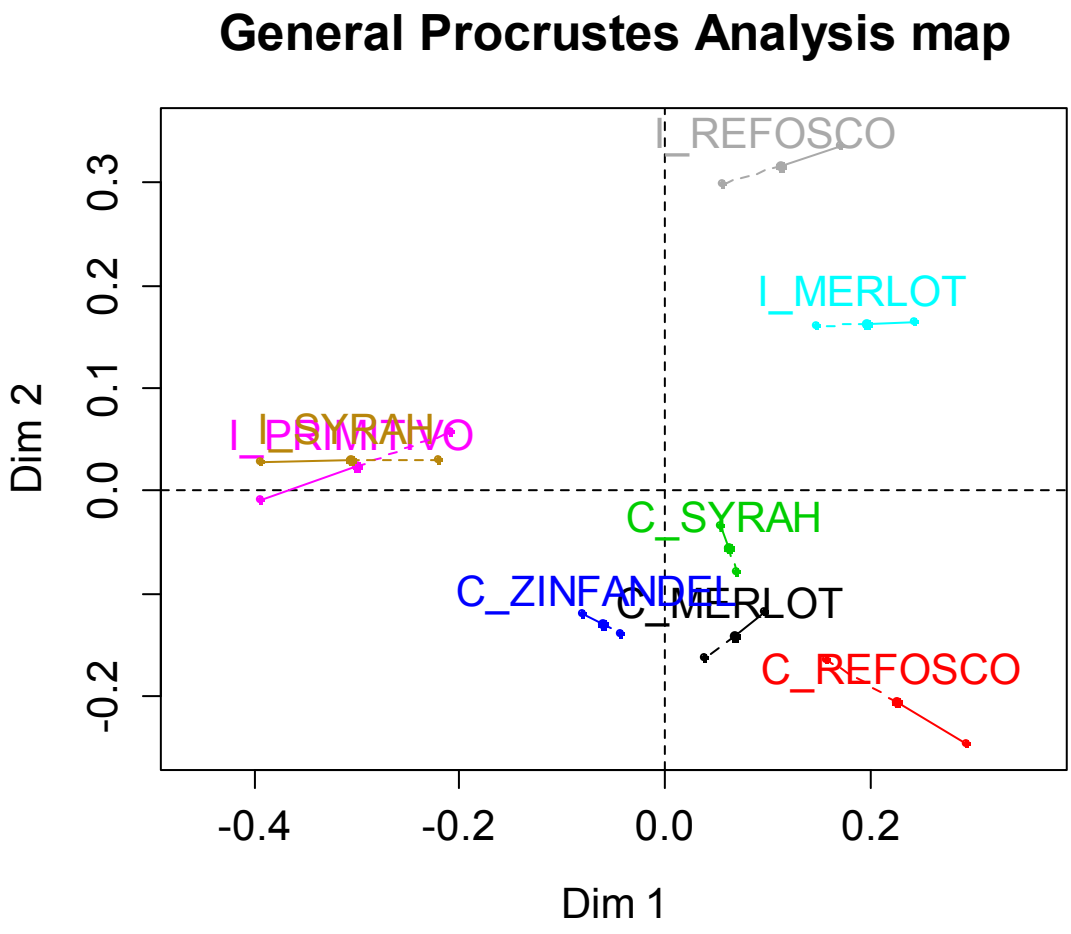
list(gpa.sc$PANOVA)

A list of "Procrustes Analysis of Variance" tables, per assesor (config), per product(object), per dimension (dimension)

```
> list(gpa.sc$PANOVA)
[[1]]
[[1]]$objet
                SSfit SSresidual    SStotal
C_MERLOT     6.786863  0.6624768   7.449340
C_REFOSCO   14.632267  0.7210783  15.353345
C_SYRAH      7.616809  0.8292166   8.446025
C_ZINFANDEL  8.715764  0.6693949   9.385158
I_MERLOT    12.418879  0.8663319  13.285211
I_PRIMITIVO 14.467136  1.1308863  15.598022
I_REFOSCO   14.346464  0.7250181  15.071482
I_SYRAH     14.577147  0.8342682  15.411415
sum         93.561329  6.4386711 100.000000

[[1]]$contribindivdim
                ssfit1      ssfit2     ssfit3    ssfit4      ssfit5
ssfit6
C_MERLOT     0.4732882  1.97344133 1.68190481 0.1786090 0.293242222
0.145059898
C_REFOSCO    5.1049444  4.23432902 1.34798190 3.4276234 0.036933346
0.445654245
C_SYRAH      0.3917016  0.31962742 1.48155620 1.2388289 0.328229492
3.682565019
C_ZINFANDEL  0.3664786  1.68110062 0.08329925 4.1444503 0.006284073
2.109794632
I_MERLOT     3.8668076  2.64163212 4.58851626 0.2430953 0.334738692
0.089938103
```

```
I_PRIMITIVO  9.0468418  0.05644565 0.74544287 0.9960486 3.615785741
0.002293545
I_REFOSCO    1.2944588 10.03387058 1.85833563 0.1625509 0.127945946
0.475099096
I_SYRAH      9.3894674  0.08640551 0.72172735 0.3996304 3.934266630
0.026148445
                SSfit7 SSresidual1  SSresidual2  SSresidual3 SSresidual4
C_MERLOT     2.041317935 0.086767415 0.0531522717 0.0171588085 0.011906092
C_REFOSCO    0.034800851 0.472842059 0.1719537829 0.0023511336 0.007589121
C_SYRAH      0.174299858 0.005015955 0.0503610646 0.0191036661 0.142685101
C_ZINFANDEL  0.324356076 0.034349060 0.0107023085 0.0430842877 0.181479185
I_MERLOT     0.654151298 0.227105876 0.0004660337 0.2581859646 0.129305510
I_PRIMITIVO  0.004277689 0.842069126 0.1115029542 0.0112328560 0.097101928
I_REFOSCO    0.394203299 0.338705055 0.0333226673 0.1383884585 0.009956540
I_SYRAH      0.019501022 0.751578990 0.0001946455 0.0004486437 0.002998110
              SSresidual5 SSresidual6  SSresidual7  SStotal 1   SStotal 2
C_MERLOT     8.361843e-03  0.10045877 0.3846716111  0.5600556  2.02659361
C_REFOSCO    9.862687e-05  0.06091814 0.0053254379  5.5777865  4.40628281
C_SYRAH      2.435238e-02  0.57720565 0.0104928013  0.3967176  0.36998848
C_ZINFANDEL  2.193047e-02  0.30561018 0.0722393787  0.4008277  1.69180293
I_MERLOT     1.276566e-02  0.02283054 0.2156722662  4.0939135  2.64209815
I_PRIMITIVO  5.975771e-02  0.00126096 0.0079607907  9.8889109  0.16794860
I_REFOSCO    3.703489e-02  0.11512060 0.0524898568  1.6331638 10.06719325
I_SYRAH      6.268229e-02  0.01546793 0.0008975906 10.1410464  0.08660016
             SStotal 3 SStotal 4  SStotal 5   SStotal 6  SStotal 7
C_MERLOT     1.6990636 0.1905150 0.30160406 0.245518665 2.42598955
C_REFOSCO    1.3503330 3.4352125 0.03703197 0.506572388 0.04012629
C_SYRAH      1.5006599 1.3815140 0.35258188 4.259770674 0.18479266
C_ZINFANDEL  0.1263835 4.3259295 0.02821454 2.415404816 0.39659545
I_MERLOT     4.8467022 0.3724008 0.34750435 0.112768648 0.86982356
I_PRIMITIVO  0.7566757 1.0931505 3.67554345 0.003554505 0.01223848
I_REFOSCO    1.9967241 0.1725074 0.16498084 0.590219700 0.44669316
I_SYRAH      0.7221760 0.4026285 3.99694892 0.041616376 0.02039861

[[1]]$contibis
             SSresidual ratio sens SSresidual ratio cons SSresidual raw sens
C_MERLOT                  0.3312384             0.3312384                  50
C_REFOSCO                 0.3605392             0.3605392                  50
C_SYRAH                   0.4146083             0.4146083                  50
C_ZINFANDEL               0.3346974             0.3346974                  50
I_MERLOT                  0.4331659             0.4331659                  50
I_PRIMITIVO               0.5654432             0.5654432                  50
I_REFOSCO                 0.3625090             0.3625090                  50
I_SYRAH                   0.4171341             0.4171341                  50
sum                       3.2193355             3.2193355                 400
             SSresidual raw cons
C_MERLOT                      50
C_REFOSCO                     50
C_SYRAH                       50
C_ZINFANDEL                   50
I_MERLOT                      50
I_PRIMITIVO                   50
I_REFOSCO                     50
I_SYRAH                       50
sum                          400

[[1]]$config
     SSfit SSresidual SStotal
sens     0   3.219336      50
cons     0   3.219336      50
sum      0   6.438671     100

[[1]]$contribconfigdim
     SSresidual 1  SSresidual 2  SSresidual 3  SSresidual 4  SSresidual 5
```

```
sens      1.379217     0.2158279     0.2449769     0.2915108     0.1134919
cons      1.379217     0.2158279     0.2449769     0.2915108     0.1134919
sum       2.758434     0.4316557     0.4899538     0.5830216     0.2269839
     SSresidual 6  SSresidual 7   SStotal1  SStotal2  SStotal3  SStotal4
sens     0.5994364     0.3748749 25.186211 11.527556  4.495785  3.709169
cons     0.5994364     0.3748749  7.506211  9.930952  8.502933  7.664689
sum      1.1988728     0.7497497 32.692422 21.458508 12.998718 11.373858
     SStotal5 SStotal6  SStotal7
sens 3.227476 1.280537 0.5732663
cons 5.676934 6.894889 3.8233914
sum  8.904410 8.175426 4.3966578

[[1]]$dimension
      Consensus   residus      Total
dim 1 29.933988 2.7584335  32.692422
dim 2 21.026852 0.4316557  21.458508
dim 3 12.508764 0.4899538  12.998718
dim 4 10.790837 0.5830216  11.373858
dim 5  8.677426 0.2269839   8.904410
dim 6  6.976553 1.1988728   8.175426
dim 7  3.646908 0.7497497   4.396658
Total 93.561329 6.4386711 100.000000
```



CONJOINT ANALYSIS

For this analysis I used the conjoint packae in R and the data set named 'tea' that is included in the package. The tea data has 5 data sets and to understand the process I asked to see each one. They are tpref (1300 observations of 1 variable), tprefm (100 observations of 13 variables), tprof (13 observations of 4 variables), tlevn (11 observations of 1 variable) and tsimp (4 observations of 4 variables)

library(conjoint)
data(tea)

tpref

```
I am only listing the last 25 or so
1275  7
1276  4
1277  3
1278  9
1279  0
1280  5
1281  4
1282  0
1283  5
1284  0
1285  6
1286 10
1287  8
1288  9
1289  7
1290  4
1291 10
1292  9
1293  3
1294  2
1295  1
1296  2
1297  3
1298  9
1299 10
1300  8
```

This is the liking data for the 100 consumers for the 13 "treatments" of tea – in the form of a long skinny vactor

tprefm

This is the same data with the "treatments" as the columns

```
> tprefm
    profil1 profil2 profil3 profil4 profil5 profil6 profil7 profil8 profil9
1         8       1       1       3       9       2       7       2       2
2         0      10       3       5       1       4       8       6       2
3         4      10       3       5       4       1       2       0       0
4         6       7       4       9       6       3       7       4       8
5         5       1       7       8       6      10       7      10       6
6        10       1       1       5       1       0       0       0       0
7         8       0       0       0       9       0       0       0       0
I CUT THE OTHER CONSUMERS
98        5       3       1       4       3       8       5       9       6
99        7       4       3       9       0       5       4       0       5
100       9       7       4      10       9       3       2       1       2
```

```
    profil10 profil11 profil12 profil13
1          2        2        3        4
2          9        7        5        2
3          1        8        9        7
4          5        2       10        9
5          6        6       10        7
6          0        0        1        1
7          0        5       10        8
I CUT THE OTHER CONSUMERS
98         8        4        3        1
99         0        6       10        8
100        3        9       10        8
```

tprof

```
> tprof
   price variety kind aroma
1      3       1    1     1
2      1       2    1     1
3      2       2    2     1
4      2       1    3     1
5      3       3    3     1
6      2       1    1     2
7      3       2    1     2
8      2       3    1     2
9      3       1    2     2
10     1       3    2     2
11     1       1    3     2
12     2       2    3     2
13     3       2    3     2
```

This shows the specifics for each of the 13 "treatments"

tlevn

```
> tlevn
        levels
1          low
2       medium
3         high
4        black
5        green
6          red
7         bags
8   granulated
9        leafy
10         yes
11          no
```

This shows the levels of each factor

tsimp

```
> tsimp
  price variety kind aroma
1     3       2    2     2
2     1       3    1     1
3     2       3    3     2
4     3       1    2     1
```

This sone is used for simulations – not of interest to us

Then I ran the conjoint analysis using the long skinny preference vector, the "treatment" profiles and the "treatment" levels.

conjoint.tea=Conjoint(y=tpref, x=tprof, z=tlevn)

```
> conjoint.tea=Conjoint(y=tpref, x=tprof, z=tlevn)
[1] "Part worths (utilities) of levels (model parameters for whole sample):"
       levnms    utls
1   intercept  3,5534
2         low  0,2402
3      medium -0,1431
4        high -0,0971
5       black  0,6149
6       green  0,0349
7         red -0,6498
8        bags  0,1369
9  granulated -0,8898
10      leafy  0,7529
11        yes  0,4108
12         no -0,4108
[1] "Average importance of factors (attributes):"
[1] 24,76 32,22 27,15 15,88
[1] Sum of average importance:  100,01
[1] "Chart of average factors importance"
```