Multivariate Analysis

By Ursula Frei, Reka Howard, William Beavis (ISU)

Introduction

Multivariate datasets: for each unit various variables have been assessed. What information do we want to extract?

- Group the units based on the various variables into groups.
- Determine how the variables interact, and which of them have the main effects on the units.
- Eliminate variables that eventually overshadow the effects of those variables we are really interested in.
- Reduce the complexity of the dataset so we can use graphical tools to help us in interpretation.

Objectives

- Analysis, graphical display and interpretation of complex datasets

Measures that Describe Similarities/Dissimilarities Between Units or Variables

Initial Example

Fig. 1 Initial example.

If we are asked to describe a group of diverse objects as given in Figure 1, our mind would automatically start to look for attributes, these objects have in common and others that allow us to divide them into groups.

Common: these are all letters

Different: Letters: A & B

Font size: large & small

Case: upper & lower

Color: red & black

We are still able to say, which of the objects are identical = highest similarity (for example the 2 black uppercase “A” in large font - positions 1 and 4 in figure 1, or the 2 red lowercase “a” in small font – positions 2 and 9, but we have to take a closer look to find the objects pairs, that share the least of the variables = lowest similarity.

Data Sheet for Initial Example

Even in a very small dataset, it would be nice to have algorithms at hand to compute similarities/dissimilarities between the objects.

Object	Letter	Font	Case	Color
1	A	L	U	B
2	A	S	L	R
3	B	S	U	B
4	A	L	U	B
5	A	S	U	R
6	B	L	L	B
7	B	S	U	R
8	B	S	L	R
9	A	S	L	R

Object	Letter	Font	Case	Color
1	1	1	1	1
2	1	0	0	0
3	0	0	1	1
4	1	1	1	1
5	1	0	1	0
6	0	1	0	1
7	0	0	1	0
8	0	0	0	0
9	1	0	0	0

Fig. 2 Data sheet for the initial example and conversion to binary variables.

Figure 2 (above) shows a datasheet we can create based on the objects in Figure 1. We have 9 objects and 4 variables. These are categorical/ordinal variables. For this example, we can transform the variables into binary variables,since each of them has only two states.

R Output for Initial Example

There are many different approaches, how to compute the dissimilarity between objects. Without looking into detail, let's just try out one and use R to calculate the distance matrix for the example.

```
## load packages

library("cluster")

#load the binarey datasheet (Ex1.csv)

Ex1_data <- read.csv("Ex1.csv", header = T)

Ex1_data
```

	Letter	Font	Case	Color
1	1	1	1	1
2	1	0	0	0
3	0	0	1	1
4	1	1	1	1
5	1	0	1	0
6	0	1	0	1
7	0	0	1	0
8	0	0	0	0
9	1	0	0	0

#calculate the simple matching coefficient with the function daisy

daisy(Ex1_data, metric = "gower")

Dissimilarities:

```
    1    2    3    4    5    6    7    8
2  0.75
3  0.50  0.75
4  0.00  0.75  0.50
5  0.50  0.25  0.50  0.50
6  0.50  0.75  0.50  0.50  1.00
7  0.75  0.50  0.25  0.75  0.25  0.75
8  1.00  0.25  0.50  1.00  0.50  0.50  0.25
9  0.75  0.00  0.75  0.75  0.25  0.75  0.50  0.25
```

Metric : mixed ; Types = I, I, I, I

Number of objects : 9

I

R output for the initial example.

The simple matching coefficient computes the percentage of variables that are different between two objects. If you have a look into the dissimilarities matrix, you can see, that for example the object pair 1-4 has a dissimilarity value of zero ($d_{14} = 0$) – these are the 2 identical large, black, uppercase “A”, we identified earlier.

You can also see, that for cases, where the dissimilarities are expressed as numbers between 0 and 1, there is a simple relation between similarities (s_{ij}) and dissimilarities (d_{ij}):

$$S_{ij} = 1 - d_{ij}$$

Furthermore a dissimilarity d_{ij} is also called a distance or metric, if it fulfills certain properties:

[1] $d_{ij} >= 0$ and $d_{ij} = 0$ if and only if i = j

[2] $d_{ij} = d_{ij}$

[3] $d_{jk} =< d_{ij} + d_{jk}$

Calculating Similarities/Dissimilarities for Different Data Types

Calculating Similarities and Dissimilarities in Binary Data

In molecular marker data based on allelic non-informative marker systems (as for example AFLP data), only the presence or absence of a specific band can be scored.

Similarity coefficients calculated in a binary marker data set depend on the question, if absences of the marker alleles in both observed objects should be taken into account or not and also, if alleles present in both observed objects should be counted once or twice.

The first step is to determine, which bands are common or different in two objects:

$\mathbf{a_{ij}}$ - number of bands present in both objects i and j

$\mathbf{b_{ij}}$ - number of bands present in object i but not in j

$\mathbf{c_{ij}}$ - number of bands present in object j but not in i

$\mathbf{d_{ij}}$ - number of bands absent in both objects i and j

Different Coefficients

Simple matching coefficient:

$$D(SM) = 1 - \frac{aij + dij}{aij + bij + cij + dij}$$

The simple matching coefficient computes the percentage of variables that are different between two objects related to all variables observed.

Jaccard's coefficient:

$$d(J) = 1 - \frac{aij}{aij + bij + cij|}$$

The Jaccard's coefficient takes only those observations into account, where an observation was made – if for example a marker band is absent in both objects observed, then the observation is considered as non-informative and not included in the calculation.

Dice coefficient:

$$d(SOR) = 1 - \frac{2aij}{2aij + bij + cij}$$

The Dice coefficient is similar to the Jaccard's coefficient as non-informative observations are not included, but a band present in both objects is counted twice.

Second Example

CALCULATING DISSIMILARITIES FOR MARKER DATA

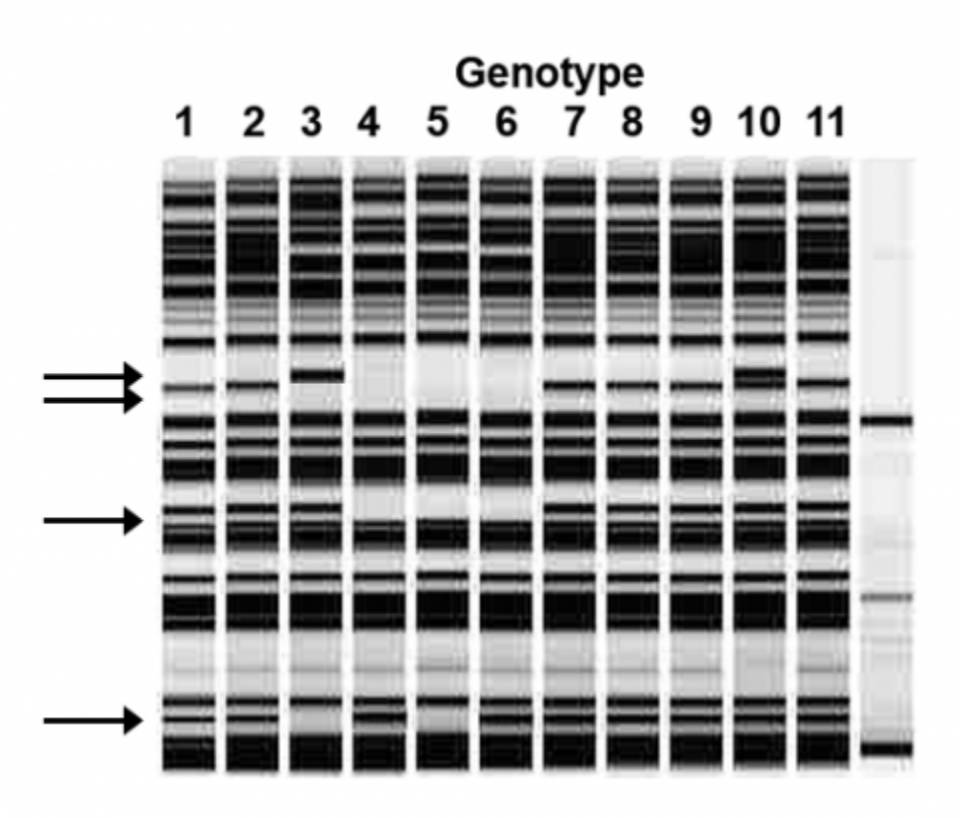


Fig. 3 Partial AFLP gel, 4 polymorphic marker bands are found within 11 genotypes analyzed.

Marker Data for Second Example

First we score the presence/ absence of the marker bands in each genotype.

Table 1 Marker data for the second example. The 0-1 matrix is saved as a .csv in Excel (Ex2.csv).

	m1	m2	m3	m4
1	0	1	1	1
2	0	1	1	1
3	1	0	1	0
4	0	0	0	1
5	0	0	0	0
6	0	0	0	1
7	0	1	1	1
8	0	1	1	1
9	0	1	1	1
10	1	1	1	1
11	0	1	1	1

Then we can calculate the dissimilarities d_{SM}, d_J and d_{SOR} using R (Ex2.csv).

For the simple matching coefficient, we can use the function daisy{cluster}; the other coefficients can be calculated with the function betadiver{vegan}. Encoding for the different diversity measures can be found in: Koleff et al. (2003) Journal of Animal Ecology, 73, 367-382.

Dissimilarity Matrices

```
> DSM
Dissimilarities :
     1    2    3    4    5    6    7    8    9   10
2  0.00
3  0.75 0.75
4  0.50 0.50 0.75
5  0.75 0.75 0.50 0.25
6  0.50 0.50 0.75 0.00 0.25
7  0.00 0.00 0.75 0.50 0.75 0.50
8  0.00 0.00 0.75 0.50 0.75 0.50 0.00
9  0.00 0.00 0.75 0.50 0.75 0.50 0.00 0.00
10 0.25 0.25 0.50 0.75 1.00 0.75 0.25 0.25 0.25
11 0.00 0.00 0.75 0.50 0.75 0.50 0.00 0.00 0.00 0.25

Metric :  mixed ;  Types = I, I, I, I
Number of objects : 11
> DJ
          1         2         3         4         5         6         7         8         9        10
2  1.0000000
3  0.2500000 0.2500000
4  0.3333333 0.3333333 0.0000000
5  0.0000000 0.0000000 0.0000000 0.0000000
6  0.3333333 0.3333333 0.0000000 1.0000000 0.0000000
7  1.0000000 1.0000000 0.2500000 0.3333333 0.0000000 0.3333333
8  1.0000000 1.0000000 0.2500000 0.3333333 0.0000000 0.3333333 1.0000000
9  1.0000000 1.0000000 0.2500000 0.3333333 0.0000000 0.3333333 1.0000000 1.0000000
10 0.7500000 0.7500000 0.5000000 0.2500000 0.0000000 0.2500000 0.7500000 0.7500000 0.7500000
11 1.0000000 1.0000000 0.2500000 0.3333333 0.0000000 0.3333333 1.0000000 1.0000000 1.0000000 0.7500000
> DSOR
          1         2         3         4         5         6         7         8         9        10
2  1.0000000
3  0.4000000 0.4000000
4  0.5000000 0.5000000 0.0000000
5  0.0000000 0.0000000 0.0000000 0.0000000
6  0.5000000 0.5000000 0.0000000 1.0000000 0.0000000
7  1.0000000 1.0000000 0.4000000 0.5000000 0.0000000 0.5000000
8  1.0000000 1.0000000 0.4000000 0.5000000 0.0000000 0.5000000 1.0000000
9  1.0000000 1.0000000 0.4000000 0.5000000 0.0000000 0.5000000 1.0000000 1.0000000
10 0.8571429 0.8571429 0.6666667 0.4000000 0.0000000 0.4000000 0.8571429 0.8571429 0.8571429
11 1.0000000 1.0000000 0.4000000 0.5000000 0.0000000 0.5000000 1.0000000 1.0000000 1.0000000 0.8571429
```

Fig. 4 Dissimilarity matrices for the three coefficients.

Calculating Similarities and Dissimilarities in Categorical Data

Categorical variables are non- numerical and it is not possible to apply any order to them.

In order to calculate distances between objects described by categorical variables, the variables can be transformed into binary variables, as we did already in example 1, for the special case when only two states for each variable are possible. If there is more than 2 states, we will have to proceed a bit differently.

Table 2
Data extracted from
http://www.fs.usda.gov/Internet/FSE_DOCUMENTS/stelprdb5186034.pdf

Common Name	Species Name	Origin	On noxious weed list	Flower	Growth form
Musk thistle	Carduus	nonnative	1	Purple	biennial
Scotch thistle	Onopordum	nonnative	1	Purple	biennial
Canada thistle	Cirsium	nonnative	1	Purple	perennial
Bull thistle	Cirsium	nonnative	0	Purple	biennial
Anderson's thistle	Cirsium	native	0	Red	perennial
Snowy thistle	Cirsium	native	0	Red	biennial
Douglas or swamp thistle	Cirsium	native	1	White	biennial
Elk or Drummond thistle	Cirsium	native	1	White	biennial
Perennial sow-thistle	Sonchus	nonnative	1	Yellow	perennial

In the example in Table 2 (Ex3.csv), only the variables “origin”, “on noxious weed list” and “Growth Form” can readily be transformed into a binary variable. “Species name” and “Flower” have 4 different stages each.

Creating Placeholder Variables

One way to handle this would be to annotate these variables the following way—we create a set of binary placeholder variables, which define the states the original variable can assume:

For example, Flower—to describe the 4 states of flower color we will need two binary variables Fl1, and Fl2:

	Purple	**Red**	**White**	**Yellow**
Fl1	1	1	0	0
Fl2	1	0	1	0

Same for species names:

	Carduus	**Onopordum**	**Cirsium**	**Sonchus**
Sn1	1	1	0	0
Sn2	1	0	1	0

How to estimate the number of placeholder variables (N) needed to represent the categorical data with X states:

$\mathrm{N} = \frac{\log \mathrm{X}}{\log 2}$, rounded up to the integer

Binary Placeholder Variables

Table 3 Binary placeholder variables for example 3 (Ex3-tr.csv)—Flower and Species name.

Common Name	Fl1	Fl2	Sn1	Sn2
Musk thistle	1	1	1	1
Scotch thistle	1	1	1	0
Canada thistle	1	1	0	1
Bull thistle	1	1	0	1
Anderson's thistle	1	0	0	1
Snowy thistle	1	0	0	1
Douglas or swamp thistle	0	1	0	1
Elk or Drummond thistle	0	1	0	1
Perennial sow-thistle	0	0	0	0

Calculating Similarities or Dissimilarities in Quantitative Data

Quantitative variables can be discrete values (for example ear row counts, pod numbers) or continuous values (for example yield (t/ha) or temperatures (°C)).

We will use the data in Table 4 (**Ex4.csv**) to calculate a few of the most commonly used similarities/dissimilarities in quantitative datasets.

Table 4

Data extracted from the Season 2014 Corn Variety Report of the University of Michigan,
http://www.varietytrials.msu.edu/wp-content/uploads/2013/01/2014-MSU-Corn-Bulletin-E-431.pdf

Brand/Hybrid	%H20	t/ha	%SL	%Sd
AGRIGOLD A6408VT3PRIB	24.7	12.93	10.7	100
AGRIGOLD A6499STXRIB	35.4	11.74	1.7	99
DIARYLAND SEED DS-9307SSX	22.7	13.25	5.1	95
DEKALB DKC57-75	24.7	12.97	1.0	99
DEKALB DKC62-08 GENSRIB	32.4	12.78	0.2	100
GOLDEN HARVEST G07F23-3111	24.7	13.28	5.9	100
GOLDEN HARVEST G12J11-3011A	29.9	12.34	5.5	99
GREAT LAKES 5755STTXRIB	25.5	14.00	10	99
NK Brand N61P-3000GT Brand	23.8	12.99	5.3	96
NK Brand N70J-3011A	29.2	12.58	9.5	97
NuTech/G2 Genetics 5Z-707^{TM}	24.2	13.68	0.9	88
RENK RK776SSTX26.0216.4	26.0	13.58	1.9	99
RUPP XRJ07-20	24.7	13.11	8.7	99
SELECT 4746 DP RIB	25.2	13.96	1.8	99
Unity Seeds 5512 SS-RIB	34.5	12.77	0.2	99
%H20 - moisture content at harvest; t/ha yield of shelled corn corrected to 15.5% moisture; %SL - percent os stalk lodging (plants broken below the ear and/or 45 degrees off vertical at harvest); %Std - percent stand of target population				

Euclidean Distance

$$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

The Euclidean distance calculates the square root of the sum over all squared differences between two objects.

In our example, the distance between the two hybrids from AGRIGOLD would be calculated as:

$$d_{12} = \sqrt{(24.7 - 35.4)^2 + (12.93 + 11.74)^2 + (10.7 - 1.7)^2 + (100 - 99)^2})(= \sqrt{197.906} = 14.07$$

You can already see that the unit the respective variable is measured in, has a great influence on the results of the Euclidean Distance. Some standardization of the data set is advisable and we will come to that later.

Manhattan Distance

$$d_{ij} = \sum_{k=1}^{n} |x_{ik} + x_{jk}|$$

The Manhattan distance describes the distance between two points in a grid, allowing only strictly horizontal or vertical paths. The distance is the sum of the horizontal and vertical components of the path between two points.

In our example, comparing the two first hybrids of **table** 3:

d_{12} = (24.7-35.4) + (12.93-11.74) + (10.7-1.7) + (100-99) = 21.89

Euclidean and Manhattan Distance Results

Below are the results for Euclidean and Manhattan distances generated in R:

```
> DEUC<-daisy(Ex4_matrix, metric = "euclidean")
> DEUC
Dissimilarities :
          1         2         3         4         5         6         7         8         9
2  14.0679103
3   7.7757572 13.8249810
4   9.7514922 10.7931877  6.0735821
5  13.0216166  3.6512464 11.9716707  7.8080791
6   4.8127435 11.6405155  5.4443457  5.0105988  9.5932268
7   7.4450050  6.7119297  8.2962702  6.9055702  5.9610066  5.3929213
8   1.8096685 13.1151668  6.9579092  9.0940035 12.0887716  4.3552727  6.5088862
9   6.7803835 12.5730863  1.5223666  5.3198120 10.7709842  4.1538055  6.8317275  5.9160882
10  5.5509008 10.1973330  8.1276626  9.8311800 10.2844543  6.5345237  4.5329461  4.4672587  6.9259007
11 15.5194233 15.8380428  8.3111311 11.0346772 14.5787517 13.0157597 13.2832827 14.3388424  9.1649386
12  8.9751045  9.5804802  6.1023684  1.6947271  6.7446275  4.3335897  5.4504679  8.1262784  5.0742586
13  2.2433011 12.8595062  5.7427868  7.7012726 11.5173304  2.9780698  6.1540962  1.7669465  4.6243270
14  9.0288925 10.4392720  5.8003534  1.3675160  7.5360732  4.3037658  6.1971284  8.2055835  4.9143565
15 14.3984583  2.0300000 13.3970295  9.8346327  2.3259622 11.3925458  7.0309957 13.3623688 12.2289983
          10        11        12        13        14
2
3
4
5
6
7
8
9
10
11 13.4599406
12  8.5440037 11.1915146
13  5.0170609 13.5061060  6.9390850
14  9.0107935 11.0855040  0.8912912  6.9701148
15 10.8911019 15.1131764  8.7060956 12.9771183  9.5113669
```

Fig. 5 R output: Euclidean distances, Manhattan distances for the example 4 data.

```
Metric :  euclidean
Number of objects : 15
> DMAN<-daisy(Ex4_matrix, metric = "manhattan")
> DMAN
Dissimilarities :
      1     2     3     4     5     6     7     8     9    10    11    12    13    14
2  21.89
3  12.92 21.61
4  10.74 12.63 10.38
5  18.35  6.54 20.07  9.69
6   5.15 17.44  7.83  6.21 13.90
7  11.99  9.90 12.51 10.33  9.24  7.54
8   3.57 20.46 12.45 10.83 18.92  6.62 10.56
9  10.36 19.45  2.56  8.22 17.91  5.79  9.95 10.41
10  9.05 16.84 13.57 15.39 15.70 11.80  6.94  7.62 11.01
11 23.05 24.94 13.13 12.31 21.80 17.90 22.64 21.72 13.49 23.70
12 11.75 11.44 10.83  2.81  9.90  6.60  8.74  9.02  9.19 13.80 13.90
13  3.18 19.07  9.74  7.84 17.53  3.97  9.17  2.99  7.42  7.83 19.87  8.57
14 11.43 12.52 10.51  2.29 10.98  6.28 10.02  8.54  8.87 15.08 13.18  1.28  8.25
15 21.46  3.43 21.18 10.80  3.11 17.01 10.33 20.03 19.02 16.79 22.91 11.01 18.64 12.09

Metric :  manhattan
Number of objects : 15
```

Fig. 6 R output: Euclidean distances, Manhattan distances for the example 4 data.

Correlation

Correlation (linear correlation coefficient, Pearson correalation coefficient):

The correlation coefficient can obtain values between -1 and 1; it measures similarity.

$$s_{ij} = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)^2 \sum_{k=1}^{n}(x_{jk} - \bar{x}_j)^2}}$$

The function cor() in R calculates similarities between columns/variables (which is the more common application for the function)—if you want to compare the rows/objects, you will have to transpose your data matrix first.

Calculating the Correlation

```
> Ex4_dft <-t(Ex4_df)
> cor(Ex4_dft)
          1         2         3         4         5         6         7         8         9        10
1  1.0000000 0.9826690 0.9981800 0.9948651 0.9863283 0.9986156 0.9950108 0.9998018 0.9982792 0.9976970
2  0.9826690 1.0000000 0.9871757 0.9922141 0.9991313 0.9884615 0.9962344 0.9853336 0.9888368 0.9925406
3  0.9981800 0.9871757 1.0000000 0.9988980 0.9916960 0.9998900 0.9968445 0.9991332 0.9999210 0.9972193
4  0.9948651 0.9922141 0.9988980 1.0000000 0.9961106 0.9987994 0.9980771 0.9966828 0.9990677 0.9965968
5  0.9863283 0.9991313 0.9916960 0.9961106 1.0000000 0.9923954 0.9976532 0.9890464 0.9928390 0.9940142
6  0.9986156 0.9884615 0.9998900 0.9987994 0.9923954 1.0000000 0.9976652 0.9994629 0.9999813 0.9982070
7  0.9950108 0.9962344 0.9968445 0.9980771 0.9976532 0.9976652 1.0000000 0.9964164 0.9977348 0.9991482
8  0.9998018 0.9853336 0.9991332 0.9966828 0.9890464 0.9994629 0.9964164 1.0000000 0.9992452 0.9982855
9  0.9982792 0.9888368 0.9999210 0.9990677 0.9928390 0.9999813 0.9977348 0.9992452 1.0000000 0.9980360
10 0.9976970 0.9925406 0.9972193 0.9965968 0.9940142 0.9982070 0.9991482 0.9982855 0.9980360 1.0000000
11 0.9914272 0.9935427 0.9970930 0.9995541 0.9973337 0.9968943 0.9972666 0.9938303 0.9973335 0.9945567
12 0.9946078 0.9931737 0.9986356 0.9999681 0.9967572 0.9986315 0.9984489 0.9964690 0.9989030 0.9968495
13 0.9997199 0.9857536 0.9992743 0.9969813 0.9894901 0.9995793 0.9966177 0.9999929 0.9993845 0.9983338
14 0.9947282 0.9918507 0.9988967 0.9999929 0.9958968 0.9987422 0.9978366 0.9965717 0.9990228 0.9963163
15 0.9821653 0.9996967 0.9879341 0.9933595 0.9996333 0.9888779 0.9959653 0.9851556 0.9893745 0.9916689
          11        12        13        14        15
1  0.9914272 0.9946078 0.9997199 0.9947282 0.9821653
2  0.9935427 0.9931737 0.9857536 0.9918507 0.9996967
3  0.9970930 0.9986356 0.9992743 0.9988967 0.9879341
4  0.9995541 0.9999681 0.9969813 0.9999929 0.9933595
5  0.9973337 0.9967572 0.9894901 0.9958968 0.9996333
6  0.9968943 0.9986315 0.9995793 0.9987422 0.9888779
7  0.9972666 0.9984489 0.9966177 0.9978366 0.9959653
8  0.9938303 0.9964690 0.9999929 0.9965717 0.9851556
9  0.9973335 0.9989030 0.9993845 0.9990228 0.9893745
10 0.9945567 0.9968495 0.9983338 0.9963163 0.9916689
11 1.0000000 0.9996191 0.9942380 0.9995707 0.9950933
12 0.9996191 1.0000000 0.9967732 0.9999367 0.9942136
13 0.9942380 0.9967732 1.0000000 0.9968763 0.9856432
14 0.9995707 0.9999367 0.9968763 1.0000000 0.9930824
15 0.9950933 0.9942136 0.9856432 0.9930824 1.0000000
```

Fig. 7 Calculating the correlation between objects after transposing the data matrix.

Preparing Data for Statistical Analysis

Preparing Data for Statistical Analysis

If we have a closer look at the data from Example 4 (**Table 4**) and the distances we calculated, we realize, that the values depend a lot on, in what unit for example the yield is measured. Changing the data from t/ha into kg/ha, would result in completely different results.

The Euclidean distance between the first two hybrids would change from 14.0679 to a value of 1190.08256! As the numerical value of yield is then much larger than the value of the other variables, yield would have a much larger weight than the other variables.

The example shows, that raw data cannot just be used for statistical analysis. It has to be prepared before any statistical analysis is applied.

Real data are never perfect, there are missing values, outliers, or other inconsistencies, which have to be dealt with.

As an example how to prepare a raw dataset for statistical analysis we will use the data collected in the file "**RawDataEarPhenotypes.xlsx**":

Four inbred lines, their respective F1 (6), F2 (6), and the 2 possible BC1 (2 x 6) were grown in a field trial with three replications. From each of the 3 x 28 plots, 10 randomly sampled ears were evaluated for 14 different ear phenotypes: row number (RowNo.), Kernels per row (K/Row), ear length (EarL), cob length (CobL), ear diameter at the base (EarDB), middle (EarDM), and tip (EarDT), cob diameter at the base (CobDB), middle (CobDM), and tip (CobDT), ear weight (EWt), grain weight (GrWt), cob weight (CobWt), and the 300 kernel weight (300KWt). So we expect a datasheet with 10 x 28 x 3 = 840 rows with data for the 14 variables measured…

Looking for Obvious Inconsistencies

Generating consistent data - looking for obvious inconsistencies:

The following inconsistencies can be found in the dataset (Fig. 6):

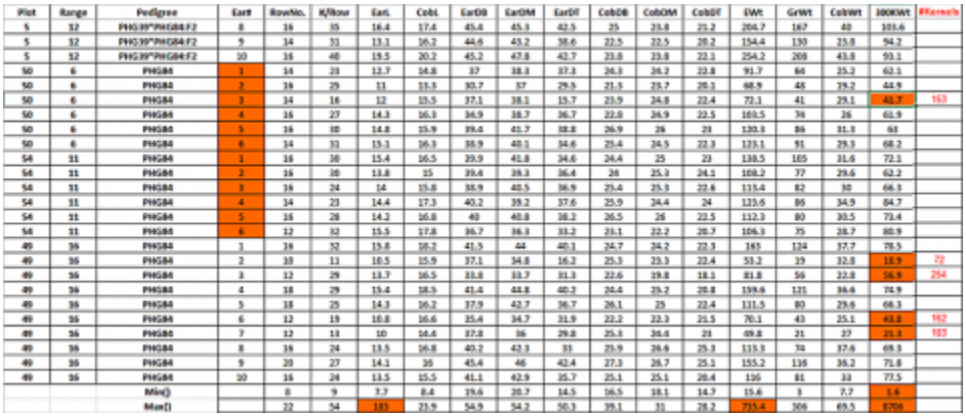
1. There are 832 instead of the expected 840 observations: in two replication of inbred line PHG84, 6 instead of 10 cobs were measured!
2. Using the excel function Min() and Max(), we observe in the 300KWt variable an unexpected high value for the 300 kernel weight of 8706—obvious a typing mistake, if we correct this value to 87.6, another obvious too high value, 998.2 shows up, we correct it to 99.82 In CobL the value of 183 as a maximum value is too high, we change it to 18.3 In EWt the value of 735.4 is too high, we change it to 73.54
3. The Min() value in 300KWt column also seems to be very low—what happened?—obviously not all cobs had the required number of 300 seed—so the seed were counted and weighed, and in an additional column #Kernels, the number of seed weighed is given. There are different ways to handle this problem
    a. Eliminate the value generated with less than 300 seed completely form the data set = missing values.
    b. Replace the value generated with less than 300 seed completely form the data set with the mean 300KWt calculated based on the available correct data.
    c. Calculate an estimate for the weight of 300 seed based on the available information—problem, if only a few seeds could be weighed this estimate can be very insecure—we might consider to do this estimate only for cases, where a certain number of seed (for example more than 150) are available.

The column “ca300KWt” is the result of applying first solution 3c and then 3a to the 300 kernel weight data.

4. Once you start reading the data into a program like R, you will realize, that there is another small typo in the EarL—one value has 2 decimal points: 21..9

Typical Data Clean-up - Example

This is an example for a typical data clean-up.



Plot	Range	Pedigree	EarN	RowNo.	K/Row	EarL	CobL	EarDB	EarDM	EarDT	CobDB	CobDM	CobDT	EWt	GrWt	CobWt	100KWt	#Kernels
5	32	PHG39*PHG84:F2	8	16	35	16.4	17.4	45.4	45.3	42.5	25	23.8	25.2	204.7	167	40	101.6	
5	32	PHG39*PHG84:F2	9	14	31	13.1	16.2	44.6	43.2	38.6	22.5	22.5	28.2	154.4	130	23.8	94.2	
5	32	PHG39*PHG84:F2	10	16	40	19.5	20.2	45.2	47.8	42.7	23.8	23.8	22.1	254.2	208	43.8	93.1	
50	6	PHG84	1	14	23	12.7	14.8	37	38.3	37.3	24.3	26.2	22.8	91.7	66	25.2	62.1	
50	6	PHG84	2	16	25	11	13.3	30.7	37	29.3	21.3	25.7	28.1	68.9	48	19.2	44.9	
50	6	PHG84	3	14	16	12	15.5	37.1	38.1	15.7	23.9	24.8	22.4	72.1	41	29.1	43.7	163
50	6	PHG84	4	16	27	14.3	16.3	34.9	38.7	36.7	22.8	24.9	22.5	103.5	74	26	61.9	
50	6	PHG84	5	16	30	14.8	15.9	39.4	41.7	38.8	26.9	26	23	120.1	86	31.8	63	
50	6	PHG84	6	14	33	15.1	16.3	38.9	40.1	34.6	25.4	24.5	22.3	123.1	91	29.3	68.2	
54	31	PHG84	1	16	30	15.4	16.5	39.9	41.8	34.6	24.4	25	23	138.5	105	31.6	72.1	
54	31	PHG84	2	16	30	13.8	15	39.4	39.3	36.4	26	25.3	24.1	108.2	77	29.6	62.2	
54	31	PHG84	3	16	24	14	15.8	38.9	40.5	36.9	25.4	25.3	22.6	113.4	82	30	66.3	
54	31	PHG84	4	14	23	14.4	17.3	40.2	39.2	37.6	25.9	24.4	24	123.6	86	34.9	84.7	
54	31	PHG84	5	16	28	14.2	16.8	40	40.8	38.2	26.5	26	22.5	112.3	80	30.5	73.4	
54	31	PHG84	6	12	32	15.5	17.8	36.7	36.3	33.2	23.1	22.2	28.7	106.3	75	28.7	80.9	
49	36	PHG84	1	16	32	15.8	16.2	41.5	44	40.1	24.7	24.2	22.3	183	134	37.7	78.5	
49	36	PHG84	2	10	11	16.5	15.9	37.1	34.8	16.2	25.3	23.3	22.4	53.2	39	32.8	18.9	72
49	36	PHG84	3	12	29	13.7	16.5	33.8	33.7	31.3	22.6	19.8	18.1	81.8	56	22.8	56.9	254
49	36	PHG84	4	18	29	15.4	18.5	41.4	44.8	40.2	24.4	25.2	20.8	159.6	121	36.6	74.9	
49	36	PHG84	5	18	25	14.3	16.2	37.9	42.7	36.7	26.1	25	22.4	131.5	80	29.6	66.3	
49	36	PHG84	6	12	19	16.8	16.6	35.4	34.7	31.9	22.2	22.3	21.5	70.1	43	25.1	43.8	162
49	36	PHG84	7	12	13	10	14.4	37.8	36	29.8	25.3	24.4	23	69.8	21	27	23.3	103
49	36	PHG84	8	16	24	13.5	16.8	40.2	42.3	33	25.9	26.6	25.3	115.5	74	37.6	69.3	
49	36	PHG84	9	20	27	14.1	16	45.4	46	42.4	27.3	26.7	25.1	155.2	116	36.2	71.8	
49	36	PHG84	10	16	24	13.5	15.5	41.1	42.9	35.7	25.1	25.1	20.4	136	81	33	77.5	
		Min()		8	9	7.7	8.4	19.6	20.7	14.5	16.5	18.1	14.7	15.6	1	7.7	1.4	
		Max()		22	54	183	23.9	54.9	54.2	50.3	39.1	33	28.2	715.4	506	69.5	8706	

Fig. 8 Extract from the raw data file - "RawDataEarPhenotypes.xlsx" - colored are inconsistencies in the data set that have to be dealt with before any statistical analysis.

Missing Values

Generating consistent data - missing values:

Some computer programs tolerate missing values, but eventually it can become necessary to replace them by estimates for the real value.

One way to go is to replace the missing value with the mean over all values or better the mean over the values within the group of your missing value.

In order to make our data set a bit more manageable we continue with the means over the 10 cobs for all repetitions and variables: "**MeanEarPhenotypes.csv**"

In this dataset only the variable ca300KWt still has missing values—4 in total. We replace these with the mean calculated with the two remaining values within each pedigree (Fig. 9).

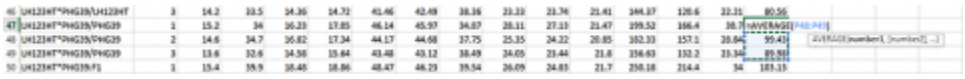


Fig. 9 The final cleaned dataset is saved as a csv file: "MeanEarPhenotypes.csv".

Cluster Analysis

Explanation of Cluster Analysis

Cluster analysis is an exploratory technique, which allows us to subdivide our sample units into groups, such that similarities of sample units within a group are larger than between groups. Cluster analysis is applied, if we have no idea how many groups there are. This is in contrast to another technique called the **Discriminant Analysis**, where the groups are given and the sample units are distributed to the groups so they fit best.

Besides the grouping of sample units, cluster analysis may also reveal a natural structure in the data and eventually allow to define prototypes for each cluster in order to reduce complexity of datasets.

There are several algorithm to perform the task of grouping, we will have a closer look at 2 of them:

1. Agglomerative hierarchical clustering
2. K-means clustering

Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering methods produce a hierarchical classification of the data.

There are two ways to do so: (1) starting from a single “cluster”, containing all units of the datasets, in a series of partitions the units are divided in n clusters, containing each one individual unit, (2) starting out from n individual “cluster”, a series of fusions are performed until only one cluster containing all units is formed (=agglomerative techniques).

Hierarchical classifications can be represented in 2 dimensional diagrams called **dendrogram**, which will show the stage at which a fusion between units has been made.

Hierarchical Clustering Example

Dataset: **MeanMeanEarPhenotypes.csv**

The dataset contains the means over all repetitions done in the 4 inbred lines, the 6 F1, the 6 F2 and the 2 x 6 BC1 families. We will perform the cluster analysis based on the Euclidean distance matrix calculated with the raw (non-standardized) data (Fig. 10).

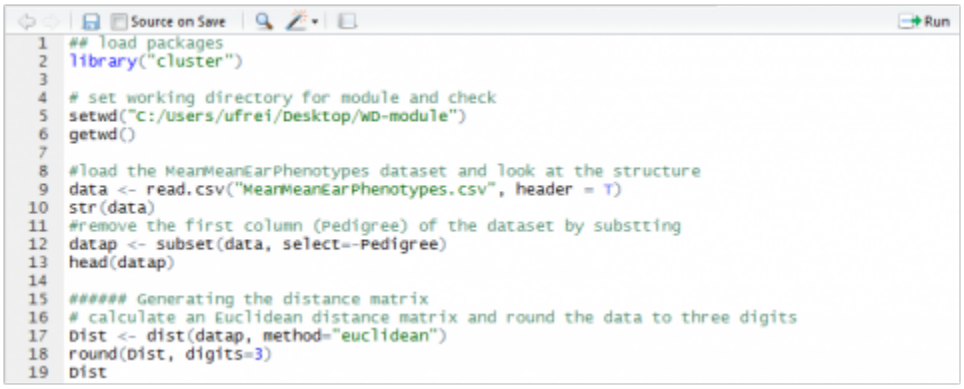


Fig. 10 R-code to prepare the data for the cluster analysis. See file: Example-HierarchicalClusterAnalysis.txt

Different Agglomeration Methods

We will be using the hclust() function of the stats package of R:

The function allows to choose between different agglomeration methods, which basically differ in how to calculate the distance between a group of units. Given the distance matrix, one can calculate the distance between two groups of units (cluster) based on the minimal distance between two individuals of the group (Fig 11a) or the maximal distance between the two individuals of a group (Fig 11b).

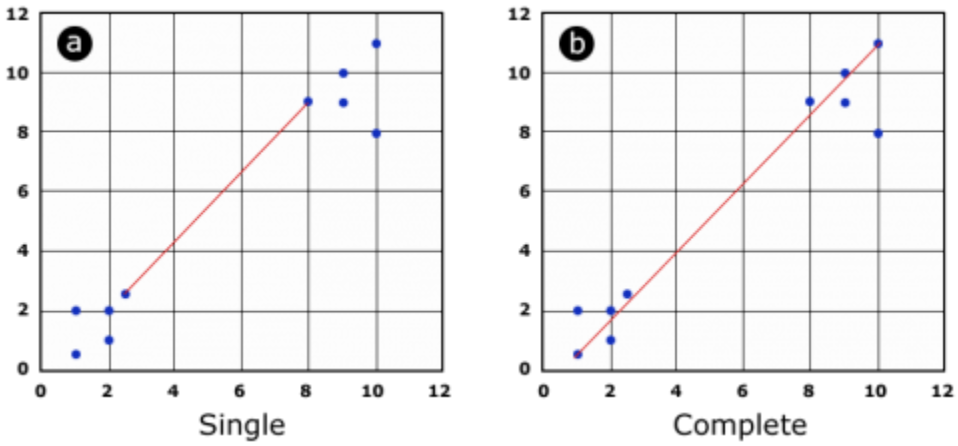


Fig. 11 Illustration of the single and complete agglomeration method.

Cluster Analysis Results

Alternatively the distance between two cluster can be calculated as the average distance of the units within these clusters, this method is also called Unweighted Pair Group Method with Arithmetic Mean (UPGMA) and widely applied.

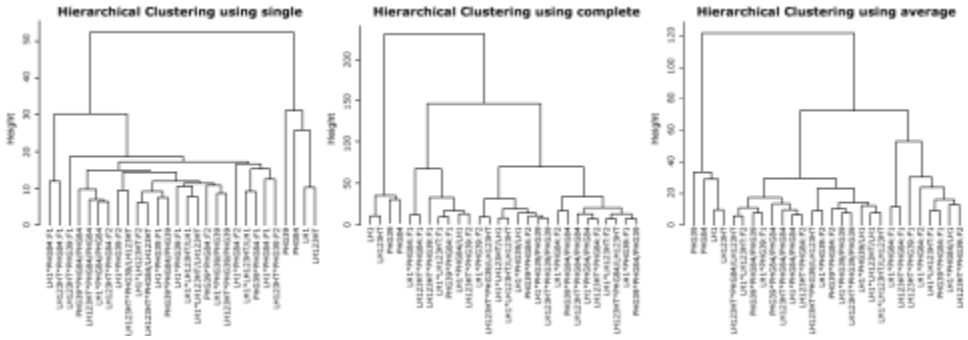


Fig. 12 Results of the cluster analysis using different agglomeration methods.

All clustering methods partition the inbred lines clearly apart from the other families (F1, F2 and BC2) (Fig. 13). The methods “complete” and “average” return a similar partition of the units, although a clear partitioning in F1 versus F2 versus BC1 is not achieved. By choosing a height at which to cut off the researcher decides, how many groups or clusters he wants to form within the dataset.

```
#### Hierarchical clustering using different methods: single, complete, average=UPGMA

x<- hclust(Dist, method="single")
plot(x, labels = data$Pedigree, hang = -0.1, main = "Hierarchical Clustering using single")

x<- hclust(Dist, method="complete")
plot(x, labels = data$Pedigree, hang = -0.1, main = "Hierarchical Clustering using complete")

x<- hclust(Dist, method="average")
plot(x, labels = data$Pedigree, hang = -0.1, main = "Hierarchical Clustering using average")
```

Fig. 13 R-code for generating the cluster and dendrogram (Example-HierarchicalClusterAnalysis.R).

Deciding a Cut-off Height

Deciding on a cut-off height, we can divide our dataset in 3, 4 or more groups. In Figure 14, the most obvious cut-off height will be at the level of 3 cluster, dividing inbred lines against mainly F1 and the rest (F2, BC1).

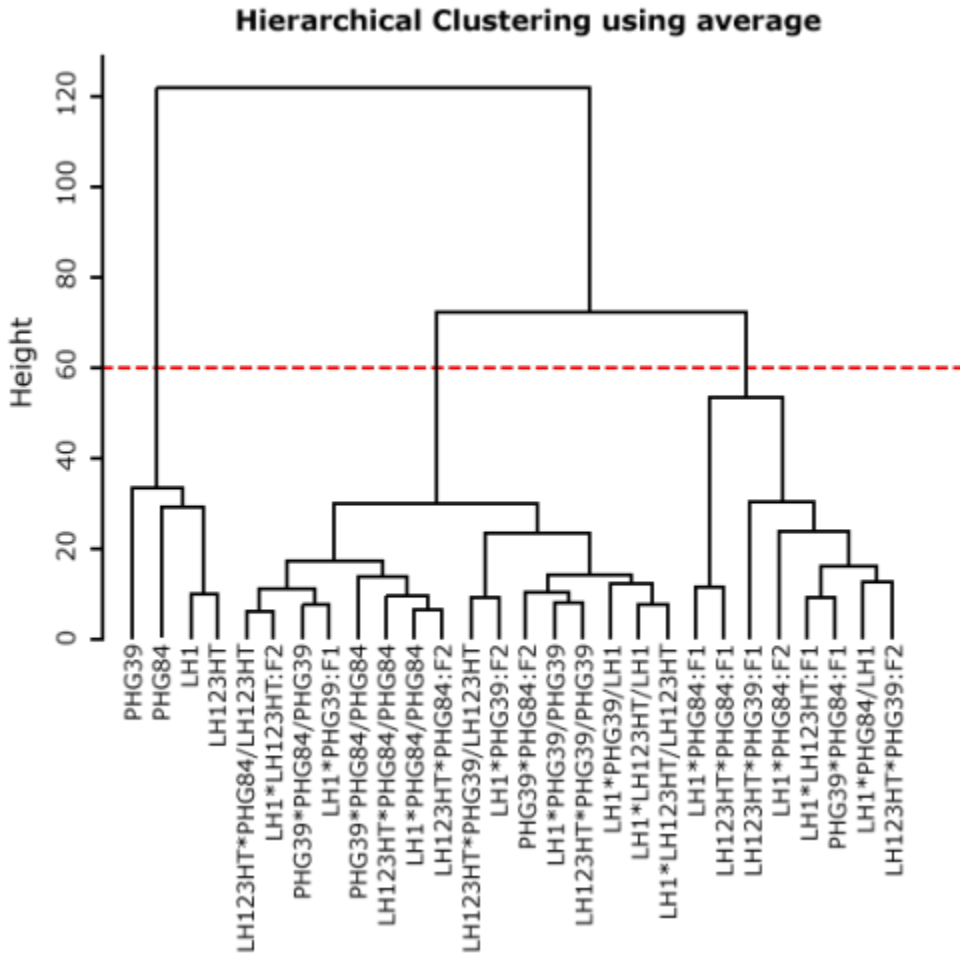


Fig. 14 Dendrogram after UPGMA.

K-means Clustering

The k-means clustering algorithm tries to partition your dataset in a given number of groups with the goal to minimize the “within group sum of squares” over all variables. Checking each possible partition of your n sample units into k groups for the lowest within-group sum of squares is not practical, as the numbers of calculations necessary rise exponentially. In our example dataset, it would be more than 2,375,000 possible partitions to check, if we assume *k=3* groups. The k-means clustering algorithm therefore starts out making an initial partition in the number of groups requested and rearranges these so that the sum of squares is minimized. The technique is called an unsupervised learning technique and can result each time it is initialized in slightly different results. For k-means clustering it is recommended to use standardized datasets.

K-means Clustering Example

Dataset: **MeanMeanEarPhenotypes.csv**

To prepare the dataset for the k-means cluster analysis, we remove the first two columns (Pedigree and Type) and standardize the data (Fig. 16)—see file **Example-kmeansClusterAnalysis.txt**

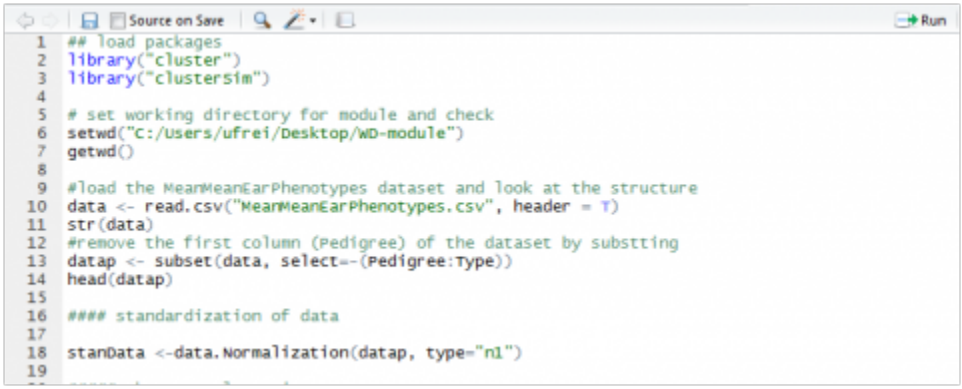


Fig. 15 Preparing the data for analysis with k-means - data standardization with the function data.Normalization{clusterSim}.

K-means Cluster Analysis

If we compare the clustering results, depending on the number of groups we allow, we see, that all k-means clustering analyses clearly put the inbred lines into a single group (Fig. 16).

F1, F2 and BC1 are distributed over the remaining groups. It looks like the “k=3 cluster analysis” is able to assign the F1 and BC1 at least to some extent mainly to their individual group, but overall, k-means gives us here the same result as the hierarchical cluster analysis.

```
>
> #####  k-means clustering
>
> cl3 <-kmeans(stanData, 3)
> table(data$Type,cl3$cluster)

         1 2 3
  BC1    8 4 0
  F1     1 5 0
  F2     2 4 0
  Inbred 0 0 4
>
> cl4 <-kmeans(stanData, 4)
> table(data$Type,cl4$cluster)

         1 2 3 4
  BC1    0 1 4 7
  F1     0 3 2 1
  F2     0 1 3 2
  Inbred 4 0 0 0
>
> cl5 <-kmeans(stanData, 5)
> table(data$Type,cl5$cluster)

         1 2 3 4 5
  BC1    1 3 6 2 0
  F1     3 2 0 1 0
  F2     1 2 1 2 0
  Inbred 0 0 0 0 4
>
```

Fig. 16 R-code for k-means cluster analysis with k=3, k=4 and k=5 groups, and assignment of the Types into these groups.

Distribution of Types

The graphics below show the data for the variables GrWt and ca300KWt, as an example how the types are distributed compared to the group assignment though k-means cluster analysis.

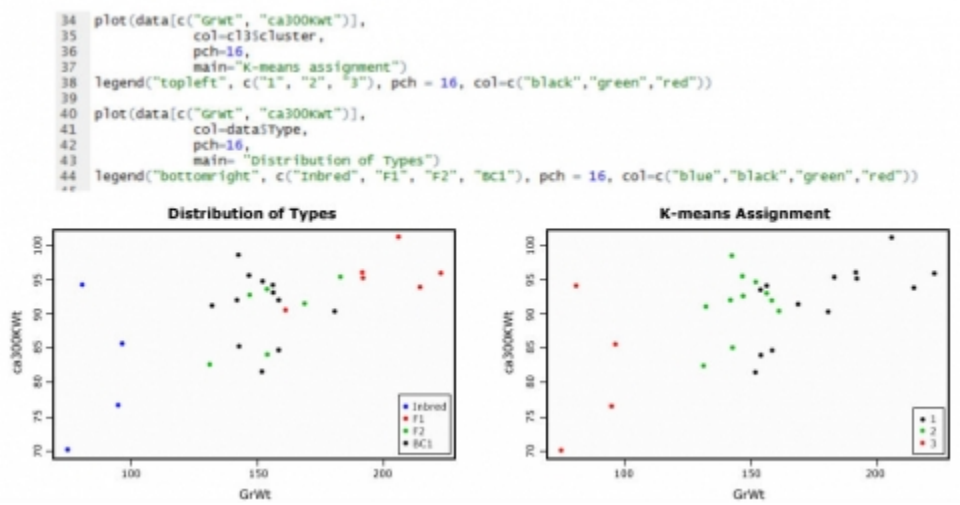


Fig. 17 Graphical display based on the Grain Weight and 300 Kernel Weight, how the types are distributed compared to the k-means assigned groups, R code to generate plots is given above.

Principal Components Analysis

Principal Components Analysis

The more variables there are in a multivariate dataset, the more difficult it becomes to describe and extract useful information from it. Also displaying the data in a graphic becomes harder when more than 3 variables are included. In this case a Principal Component Analysis (PCA) helps to reduce the dimensionality of our dataset, by changing the set of correlated variables we want to describe into a new set of uncorrelated variables. The new variables are sorted in order of importance, the first one accounting for the largest portion of variation found in the original dataset, the later for a less and less large portions of the variation. The hope is, that a few of these new variables will be sufficient to describe most of the variability found in your original dataset, so we can replace our data with only a few variables and end up with less complexity and dimensionality.

In our first example we will look at a dataset that has only two variables for an easy demonstration of PCA. We will use the following dataset: MeanMeanEarPhenotypes.csv, and create a subset consisting of the columns GrWt and ca300KWt only.

PCA Step by Step....

Dataset: MeanMeanEarPhenotypes.csv, R-code: Example-PCA1.R

At first we will have to standardize our data if the scales in your dataset are similar, we can use mean centering for standardization and do the subsequent calculation based on covariance. If the scales in a dataset are very different (weights, temperatures…) it is recommended to divide the mean center values by the standard deviation and use the correlation for subsequent calculations.

Head of the standardized data for the variables GrWt and ca300KWt

```
head(PCAdata)

       GrWt     ca300KWt

1  -0.3157250   0.23311021

2  -0.2982722   1.18856570

3  -0.2955165  -0.76213572

4  -0.1779398   0.75378250

5   0.7562440  -0.02213083

6   0.1371291  -0.83589358
```

```
#display data in a plot

plot(PCAdata[c("GrWt", "ca300KWt")]
+       col-data$Type,
+       pch-16
+       main-"Distribution of data for PCA example")

legend("bottomright", c("Inbred", "F1", "BC1"), pch = 16, col=c("blue","red","green","black"))
```

Distribution of Data

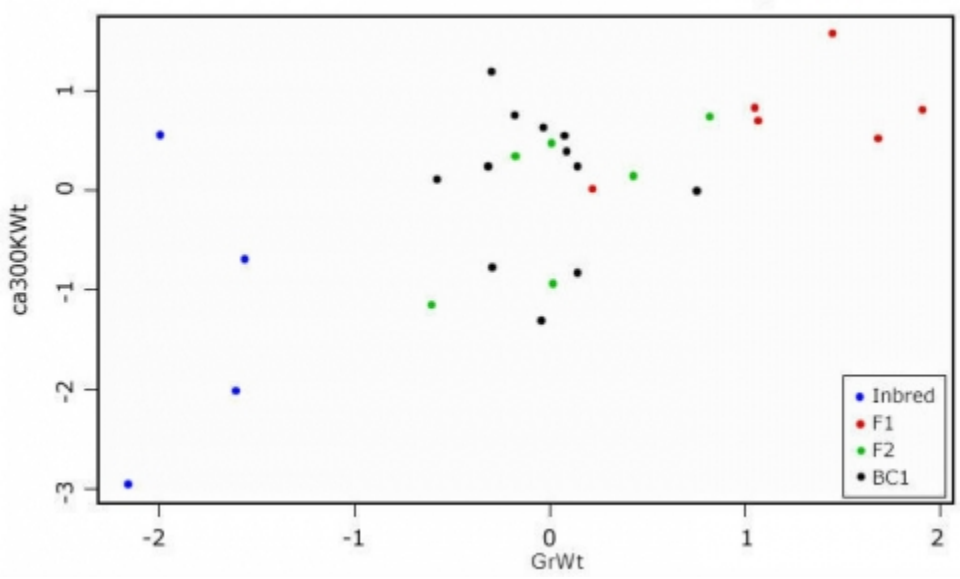


Fig. 18 Scatterplot to display the standardized data for the PCA example.

Looking at the scatterplot (Fig. 18), we see, that there is some correlation between the two measures. If we want to know which of the variables contributes the most to the overall variance of the dataset, we have to calculate the Eigenvalues for the covariance (or correlation) matrix.

Eigenvectors Output Matrix

```
my.eigen$values

[1] 1.6208572  0.3791428

rownames(my.eigen$vectors) <- c("GrWt", "ca300KWt")

colnames(my.eigen$vectors) <-c("PC1", "PC2")

my.eigen$vectors

                PC1        PC2

GrWt        -0.7071068   0.7071068

ca300KWt    -0.7071068  -0.7071068



sum(my.eigen$values)

[1] 2

var(PCAdata$GrWt) + var(PCAdata$ca300KWt)

[1] 2
```

|

In the output matrix of the Eigenvectors, the first column is also our first Principal Component, the second column is the second Principal Component. The values are a measure of the strength of association with the Principal Component While values in the first Principal Component trend together, in the second they trend apart. We will try to visualize this in a graphic.

Display of Principal Components

```
PC1.slope <- my.eigen$vectors[1,1]/my.eigen$vectors[2,1]

PC2.slope <- my.eigen$vectors[1,2]/my.eigen$vectors[2,2]



abline(0, PC1.slope, col="green")

abline(0, PC2.slope, col="red")
```

Graphical display of the Principal Components. PC1 in green, PC2 in red—the two components are orthogonal to each other.

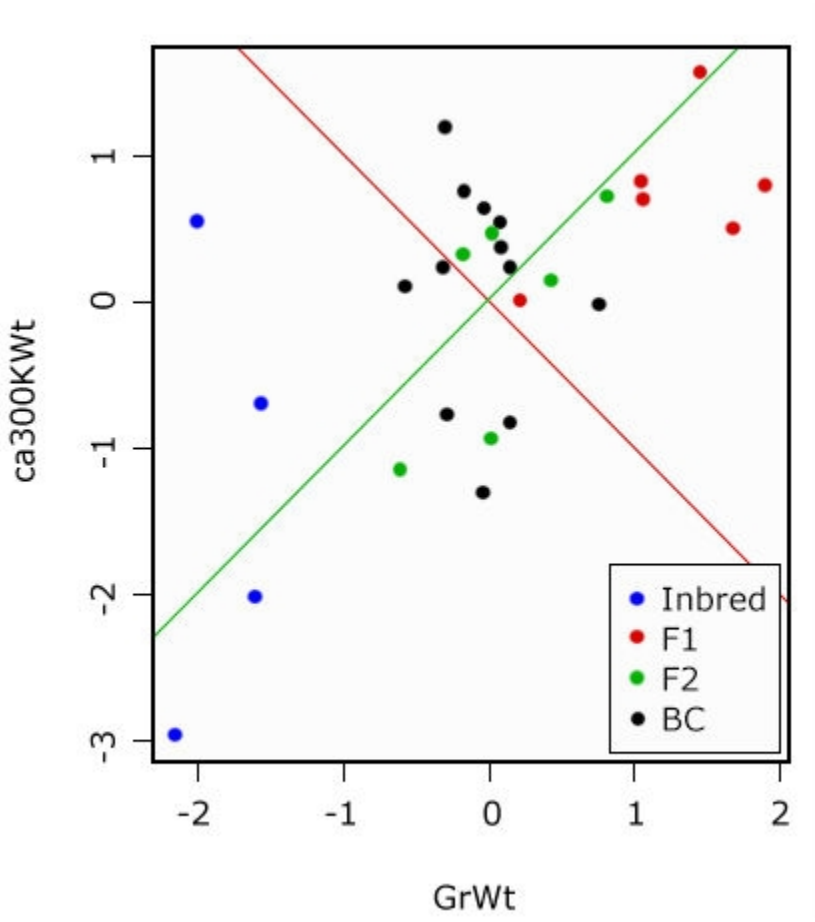


Fig. 19 Distribution of data for PCA example

Percentage of Overall Variance

Finally we want to ask how much (or what percentage) of the overall variance in our data is represented by the first Principal Component:

```
PC1.slope <- my.eigen$vectors[1,1]/my.eigen$vectors[2,1]

PC2.slope <- my.eigen$vectors[1,2]/my.eigen$vectors[2,2]



abline(0, PC1.slope, col="green")

abline(0, PC2.slope, col="red")



PC2.var <-100 * (my.eigen$values[1]/sum(my.eigen$values))

PC2.var <-100 * (my.eigen$values[2]/sum(my.eigen$values))

PC1.var

[1] 81.04286

PC2.var

[1] 18.95714
```

|

PC1 explains ca. 81% of the variance, PC2 explains about 18%.

Calculate the PCA Scores

By multiplying the original variable with the respective Eigenvectors, we calculate the PCA scores for each sample unit in our dataset. Plotting the scores shows, that the plot is rotated such, that the Principal Components form the x and y axis (Fig. 20). The relations between the sample units is unchanged, although one should be aware, that a Principal Component per se has no biological meaning.

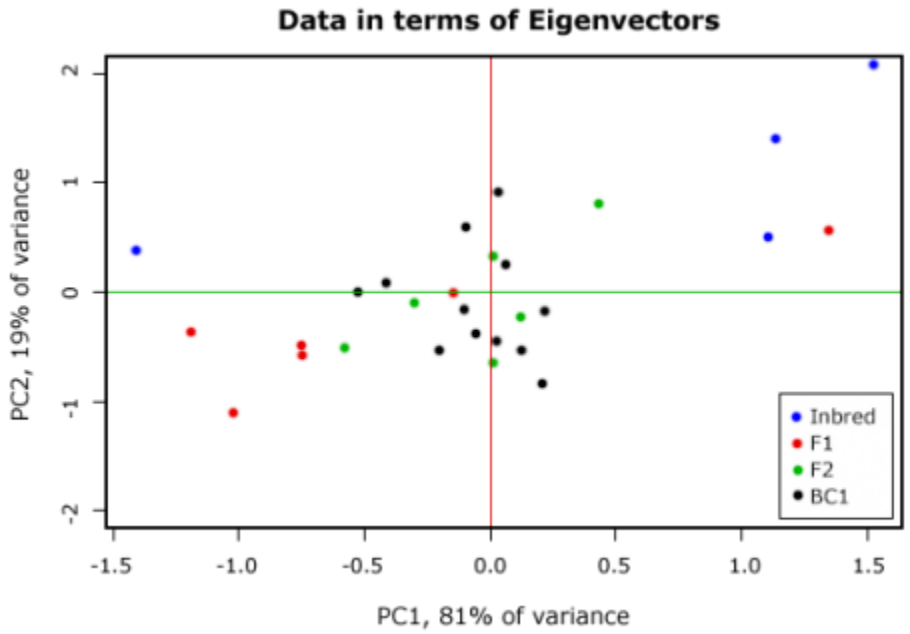


Fig. 20 Plotting the PCI scores.

It is possible to plot the variables as vectors into the graphic (Fig. 20)—how to do this and how to interpret the arrows will be shown in the next example.

PERFORMING A PRINCIPAL COMPONENT ANALYSIS USING INBUILT R FUNCTIONS

Dataset: MeanMeanEarPhenotypes.csv, and create a subset consisting of the columns: RowNo., EarDM, EarL, CobWt, ca300KWt (Fig. 21), we will use the standardized data.

R-code: Example-PCA2.R

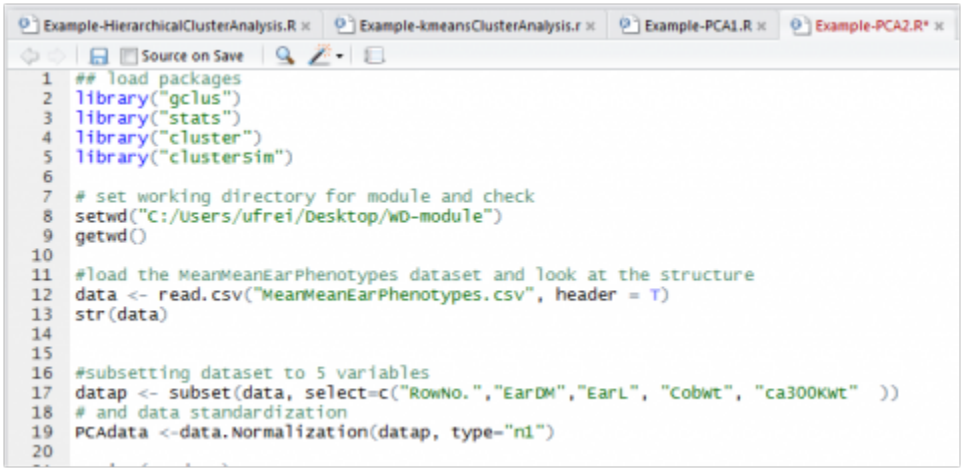


Fig. 21 Preparing an example dataset with 5 variables.

Generate a Scatterplot Matrix

With the cpair() function we can generate a scatterplot matrix of our dataset (Fig. 22).

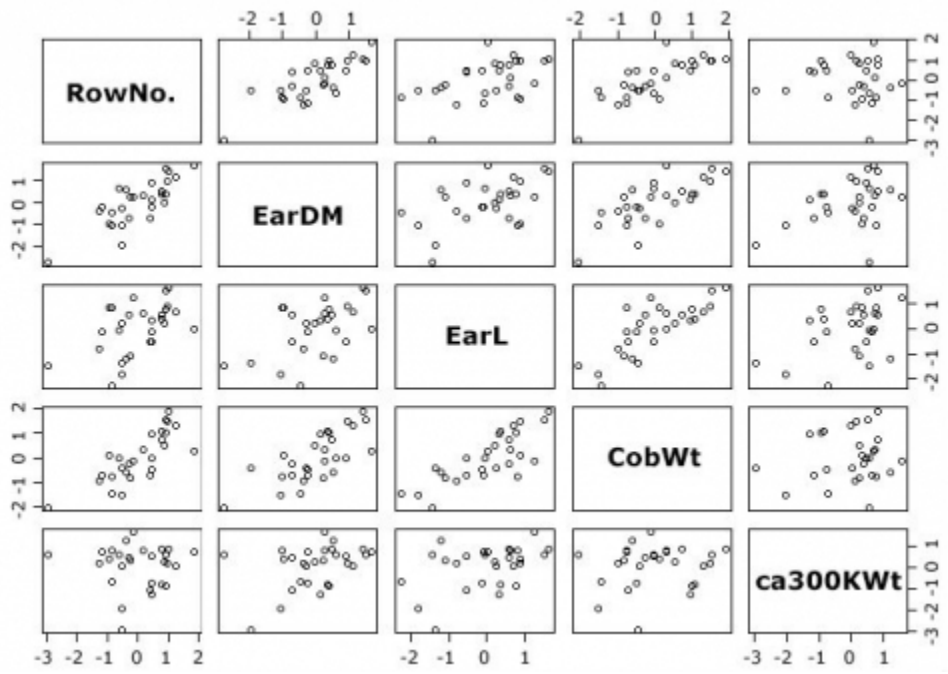


Fig. 22 Scatterplot matrix for the 5 variables dataset.

Not unsurprising, for example, there is a strong correlation between the RowNo. and ear diameter.

Calculate the Principal Components

To calculate the Principal Components we will use the function prcomp() {stats}.

```
PCA2 <- prcomp(PCAdata, center=T, scale=T)

class(PCA2)

[1] "prcomp"

1s(PCA2)

[1] "center"  "rotation"  "scale"   "sdev"    "x"

summary(PCA2)

Importance of components:

                          PC1     PC2     PC3      PC4      PC5

Standard deviation      1.7858  1.0466  0.7404  0.33744  0.23121

Proportion of Variance  0.6378  0.2191  0.1096  0.02277  0.01069

Cumulative Proportion   0.6378  0.8569  0.9665  0.98931  1.00000
```

Looking at the results of prcomp 5 vectors are listed note that “x” stands for the Principal component scores. In the summary each Principal component is assigned its standard deviation, the proportion of the overall variance, which is explained with this component as well as a cumulative proportion. The first three components explain already more than 96% of the overall variance in the dataset. As the main goal of the PCA is to reduce the complexity of the dataset, we could ask ourselves, how many Principal components we have to keep.

Loadings of the Principal Components

In order to see which variable contributes mainly to the Principal Components, you will have to look at the rotation data, or loadings:

```
PCA2$rotation
```

```
                PC1         PC2         PC3         PC4         PC5
RowNo.    0.4766670 -0.40713953   0.2610523   0.6767104 -0.2845007
EarDM     0.4964644  0.01002647   0.5805497 -0.3360285  0.5508806
EarL      0.4595622  0.22644933 -0.6767696   0.2657527  0.4570357
CobWT     0.5197882 -0.18893398 -0.2894435 -0.5855696 -0.5171605
ca300KWt  0.2119776  0.86438506   0.2302587   0.1250267 -0.3731665
```

While all variables contribute almost equally to the first component, the ca300KWt has a strong influence on the second component, in the third components variables that measure the ear dimensions are prevalent. This can also be shown in a simple graphic (Fig. 23).

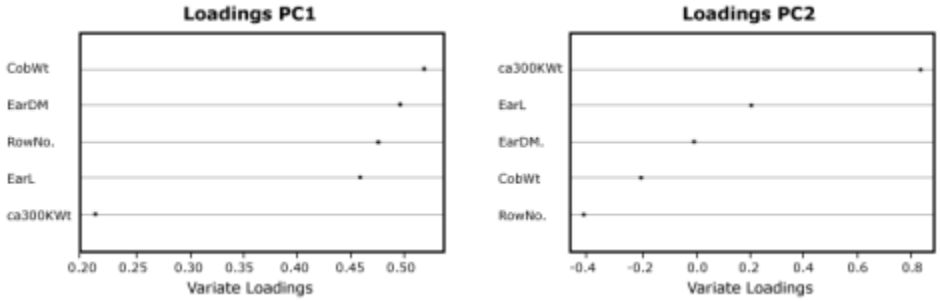


Fig. 23 Loadings of the Principal components 1 and 2 in a dot plot.

Scree Plot

Coming back to the question, how many Principal Components should be considered - there are two ways to answer for this question:

1. The Kaiser criterion: as long as the Eigen Value of a component is larger than 1 keep it (Fig. 24).
2. Make a decision based on a Scree plot: keep components, as long as the rate of change is still larger than 0.

So based on the Kaiser criterion we would keep the first two components, based on the Scree plot maybe the first three components would be kept, this is up to the researcher.

```
>
> #calculate Eigen values for each component: square of standard deviation
> PCA2$sdev^2
[1] 3.18910550 1.09535795 0.54821611 0.11386400 0.05345645
> # Graphical display
> screeplot(PCA2, type="line", main="Scree Plot")
>
```

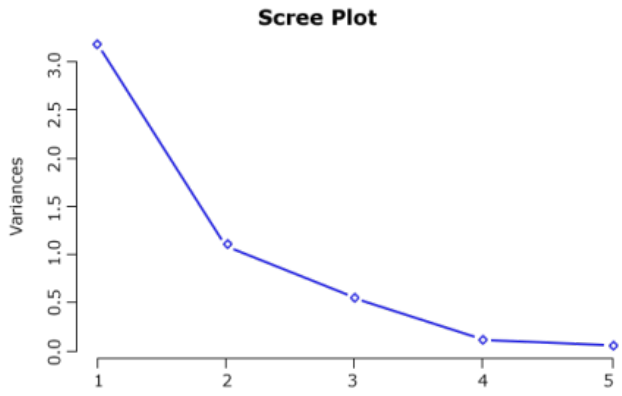


Fig. 24 R-output for Eigen Values of Principal Components and Scree plot.

Create a Biplot

Finally we want to create a biplot to visualize the results of our PCA (Fig. 25) the observations are plotted as points, the variables as vectors:

```
#graphical display:
biplot(PCA2,cex =c(1, 0.7) )
biplot(PCA2,cex =c(1, 0.7),xlabs=data$Type )
```

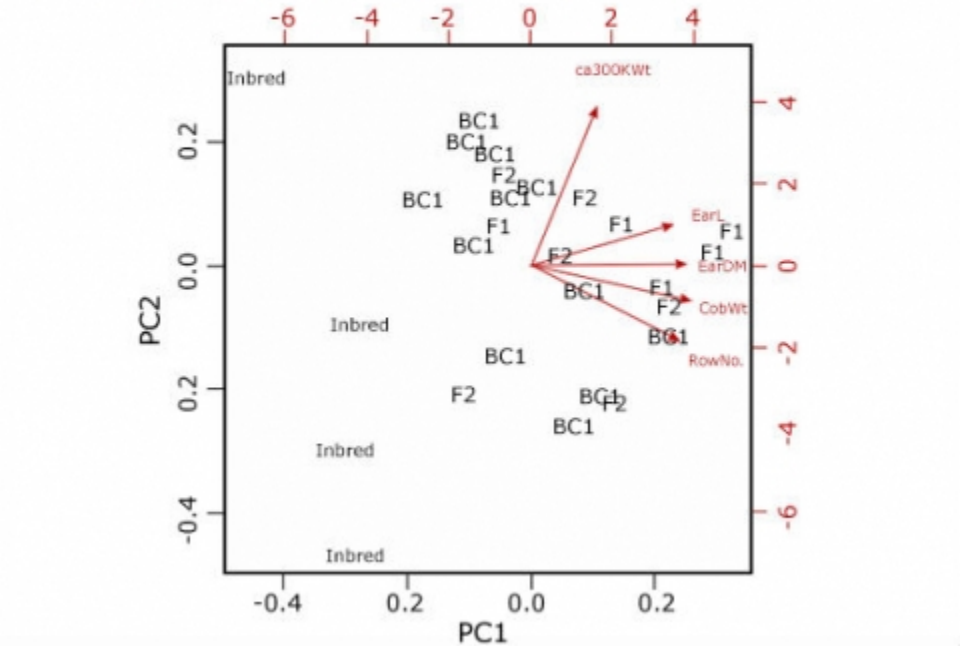


Fig. 25 R-output for Eigen Values of Principal Components and Scree plot.

The biplot reveals, what we saw looking at the loadings of the principal components: although trending in the same direction as the other 4 variables, the ca300KWt vector is set apart from the other 4. The cosines of the angles between the vectors reflect actually the correlation between these variables.

Reflection

The **Module Reflection** appears as the last "task" in each module. The purpose of the Reflection is to enhance your learning and information retention. The questions are designed to help you reflect on the module and obtain instructor feedback on your learning. Submit your answers to the following questions to your instructor.

1. In your own words, write a short summary (< 150 words) for this module.
2. What is the most valuable concept that you learned from the module? Why is this concept valuable to you?
3. What concepts in the module are still unclear/the least clear to you?

Acknowledgements

This module was developed as part of the Bill & Melinda Gates Foundation Contract No. 24576 for Plant Breeding E-Learning in Africa.

Quantitative Methods Multivariate Analysis Author: Ursula Frei, Reka Howard, and William Beavis (ISU)

Multimedia Developers: Gretchen Anderson, Todd Hartnell, and Andy Rohrback (ISU)

How to cite this module: Frei, U., R. Howard, and W. Beavis. 2016. Multivariate Analysis. *In* Quantitative Methods, interactive e-learning courseware. Plant Breeding E-Learning in Africa. Retrieved from https://pbea.agron.iastate.edu.

Source URL: https://pbea.agron.iastate.edu/course-materials/quantitative-methods/multivariate-analysis-0?cover=1