CT5102: Programming for Data Analytics

Week 4: Factors

https://github.com/JimDuggan/CT5102

Dr. Jim Duggan,
Information Technology,
School of Engineering & Informatics

Categorical Variables

- Can take on one of a limited, and usually fixed, number of possible values
- Assign each observation to a category
- Nominal
  - Ordering not important
  - Example Diabetes(Type1, Type2), Gender(Male, Female)
- Ordinal
  - Implies order, but not amount
  - Health Status(Poor, Improved, Excellent), Satisfaction(Low, Medium, High)

Examples of Categories

Category	Set
Gender	{Male, Female}
Student	{Undergraduate, Postgraduate}
Footballer	{Goalkeeper, Defender, Midfielder, Attacker}
Cohort	{Age 0-4, Age 5-14, Age 15-24, Age 25-44, Age 45-64, Age 65+}
Staff	{Lecturer, Senior Lecturer, Professor, Dean, Registrar, President}
Travel to Work	{Car, Train, Bicycle, Walk, Bus}

Factors

- Categorical variables are factors in R
- "Might be viewed simply as a vector with a bit more information added – a record of distinct values in the vector called *levels."(* Matloff 2009)

```
> x<-c(5,12,13,12)
> x
[1]  5 12 13 12
> xf<-factor(x)
> str(xf)
 Factor w/ 3 levels "5","12","13": 1 2 3 2
> unclass(xf)
[1] 1 2 3 2
attr(,"levels")
[1] "5"  "12" "13"
```

Factor vs Vector

```
> x
[1]  5 12 13 12
> xf
[1] 5  12 13 12
Levels: 5 12 13
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00   10.25   12.00   10.50   12.25   13.00
> summary(xf)
 5 12 13
 1  2  1
> mean(x)
[1] 10.5
> mean(xf)
[1] NA
```

Generating Random Factors

```
> l<-sample(LETTERS[1:10],10000,replace=T)
> lf<-factor(l)
> summary(lf)
   A    B    C    D    E    F    G    H    I    J
1000 1018  996  951 1023  981  994 1054 1026  957
> str(lf)
 Factor w/ 10 levels "A","B","C","D",..: 5 9 3 8 8 1 10 4 2 6 ...
```

```
> pop<-factor(sample(c("Male","Female"),100,replace=T))
> summary(pop)
Female   Male
    50     50
```

Generating a medical data set

```
patientID<-c(1,2,3,4)
age<-c(25,34,28,52)
diabetes<-c("Type1","Type2","Type1","Type1")
status<-c("Poor","Improved","Excellent","Poor")

status<-factor(status,levels=c("Poor","Improved","Excellent"))
diabetes<-factor(diabetes)

pd<-data.frame(patientID,age,diabetes,status)

summary(pd)
```

Factors are processed by R

```
> pd<-data.frame(patientID,age,diabetes,status)
> pd
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor
> summary(pd)
   patientID          age          diabetes       status
 Min.   :1.00   Min.   :25.00   Type1:3   Poor      :2
 1st Qu.:1.75   1st Qu.:27.25   Type2:1   Improved  :1
 Median :2.50   Median :31.00             Excellent:1
 Mean   :2.50   Mean   :34.75
 3rd Qu.:3.25   3rd Qu.:38.50
 Max.   :4.00   Max.   :52.00
```

The table() function

- Used to gather frequency data
- More than one factor can be analysed
- Powerful function in R
- The first argument is a factor or a list of factors

Examples

```
> pd
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor
```

```
> table(pd$diabetes)

Type1 Type2
    3     1
> table(pd$status)

     Poor  Improved Excellent
        2         1         1
> table(pd$diabetes,pd$status)

       Poor Improved Excellent
Type1     2        0         1
Type2     0        1         0
```

Contingency Tables

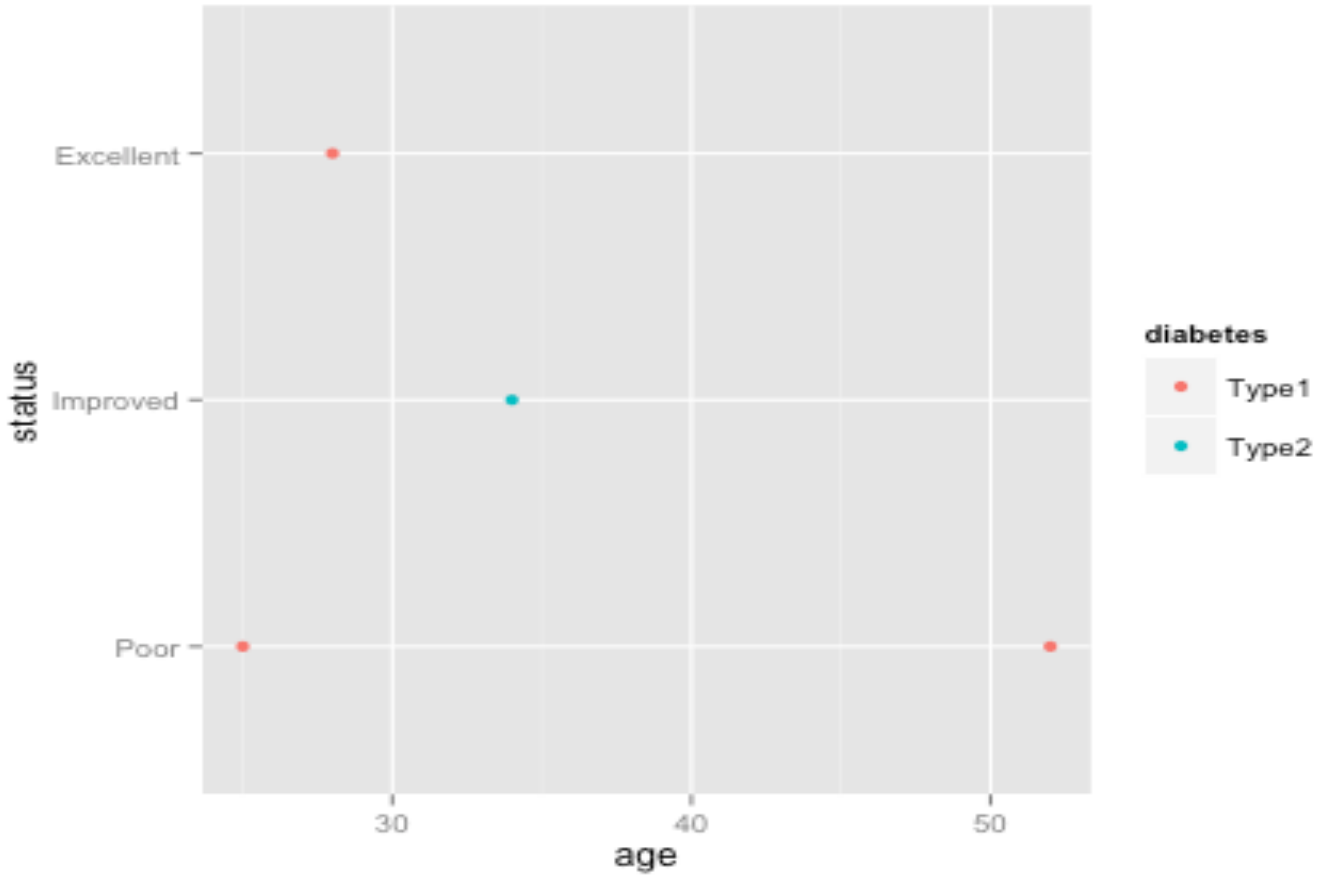
- A contingency table is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables.

	Right-handed	Left-handed	Total
Males	43	9	52
Females	44	4	48
Totals	87	13	100

https://en.wikipedia.org/wiki/Contingency_table

Visualisation (1)

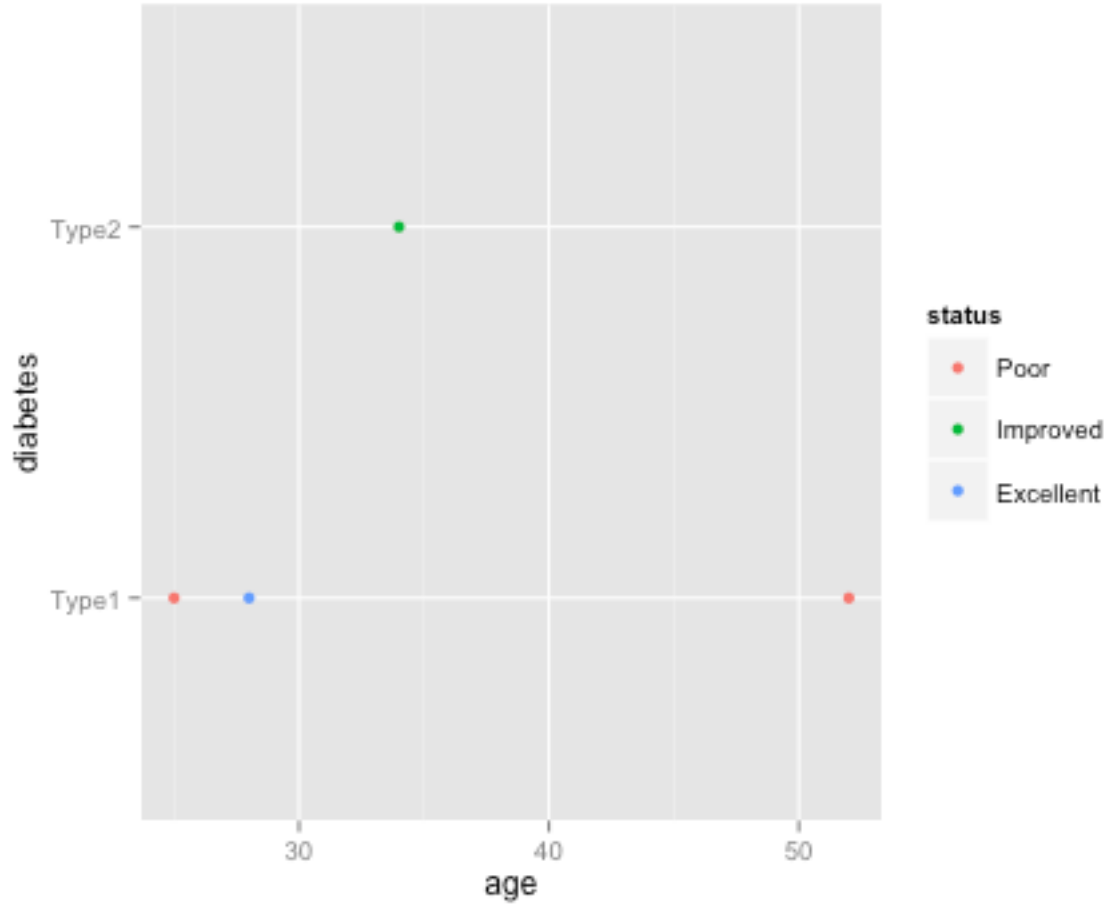
```
> qplot(pd,x=age,y=status,color=diabetes)
```



http://docs.ggplot2.org/0.9.3/qplot.html

Visualisation (2)

```
> qplot(pd,x=age,y=diabetes,color=status)
```



Challenge 4.1

- Create a random data set of 1000 patients
  - Id is a unique 4 digit identifier in the range {1000-9999}
  - Ages are uniform between 20-65
  - Gender based on 60/40 male to female split
  - Diabetes (Type I, Type 2) based on (70:30) split
  - Status (Poor, Improved, Excellent) in split (30:40:30)
- Use the table() function to explore to display contingency tables
- Plot the information using qplot()

R Script

```
ids<-sample(1000:9999,1000)

age<-sample(20:65,1000,replace=T)

diabetes<-sample(c("Type 1", "Type 2"),1000,
                 replace=T,prob=c(.7,.3))

gender<-sample(c("Male","Female"),1000,replace=T,prob=c(.6,.4))

status<-sample(c("Poor","Improved","Excellent"),
               1000,replace=T,prob=c(.3,.4,.3))

df<-data.frame(ids,ages,diabetes,status,gender)

qplot(data=df,x=age,y=status,color=gender)
```

Data Frame

```
> head(df)
   ids ages diabetes    status gender
1 7262   38   Type 1  Improved   Male
2 3982   25   Type 1 Excellent   Male
3 3382   44   Type 2 Excellent   Male
4 5737   49   Type 1      Poor Female
5 2296   34   Type 1  Improved   Male
6 5330   57   Type 2  Improved Female
```

Tables

```
> table(df$status)

Excellent  Improved      Poor
      310       370       320
> table(df$diabetes)

Type 1 Type 2
   685    315
> table(df$gender)

Female   Male
   392    608
```

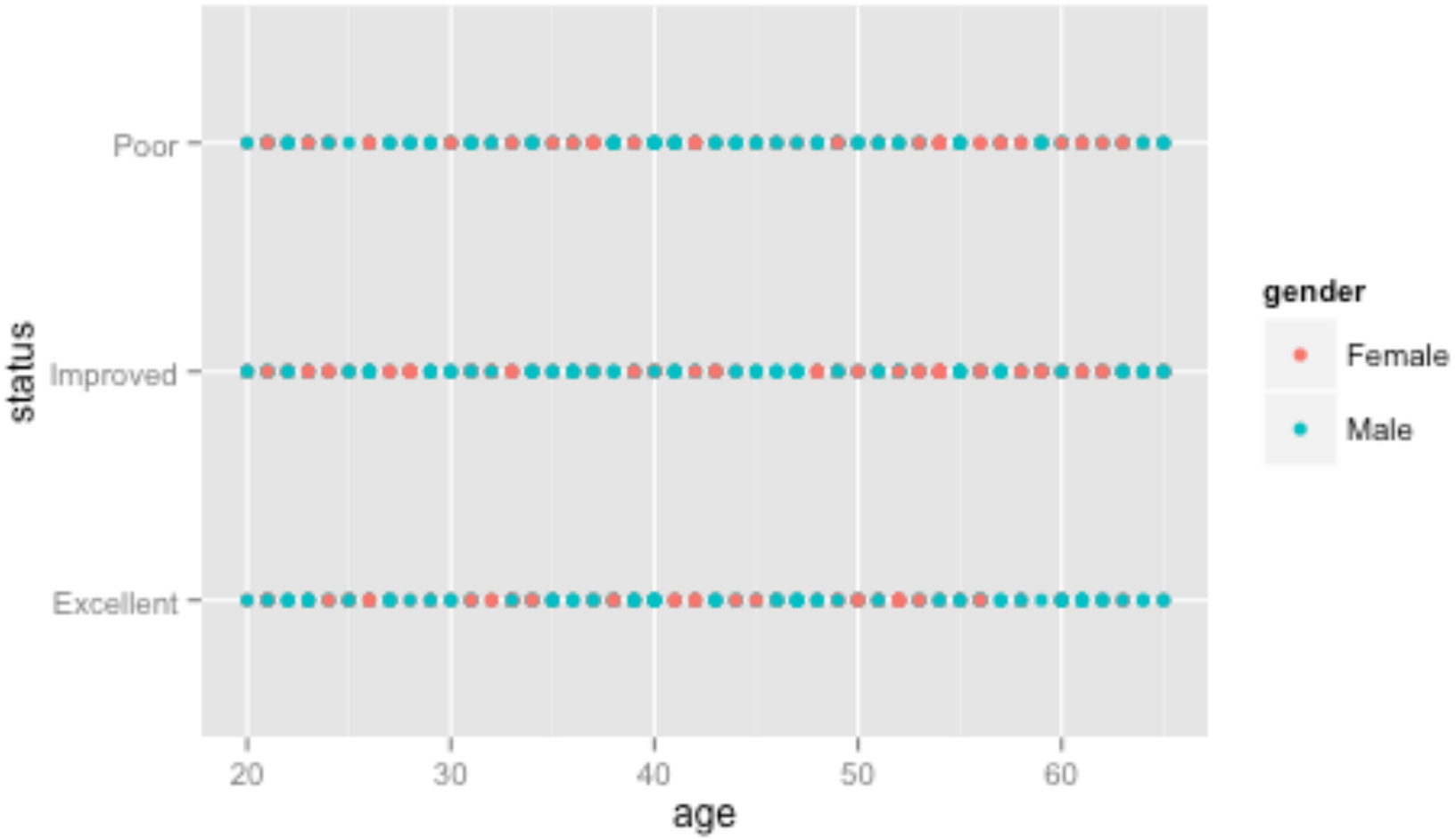
```
> table(df$gender,df$diabetes)

        Type 1 Type 2
Female     263    129
Male       422    186
> table(df$diabetes,df$status)

        Excellent Improved Poor
Type 1        208      267  210
Type 2        102      103  110
> table(df$gender,df$status)

        Excellent Improved Poor
Female        116      153  123
Male          194      217  197
```

Plot


Poor
Improved
Excellent
status
20
30
40
50
60
age
gender
Female
Male

gl() function – for generating factors

Description

Generate factors by specifying the pattern of their levels.

Usage

```
gl(n, k, length = n*k, labels = seq_len(n), ordered = FALSE)
```

Arguments

`n`	an integer giving the number of levels.
`k`	an integer giving the number of replications.
`length`	an integer giving the length of the result.
`labels`	an optional vector of labels for the resulting factor levels.
`ordered`	a logical indicating whether the result should be ordered or not.

Example 1

```
> gl(2,1,100,labels=c("Male","Female"))
  [1] Male   Female Male   Female Male   Female Male   Female Male   Female
  [9] Male   Female Male   Female Male   Female Male   Female Male   Female
 [17] Male   Female Male   Female Male   Female Male   Female Male   Female
 [25] Male   Female Male   Female Male   Female Male   Female Male   Female
 [33] Male   Female Male   Female Male   Female Male   Female Male   Female
 [41] Male   Female Male   Female Male   Female Male   Female Male   Female
 [49] Male   Female Male   Female Male   Female Male   Female Male   Female
 [57] Male   Female Male   Female Male   Female Male   Female Male   Female
 [65] Male   Female Male   Female Male   Female Male   Female Male   Female
 [73] Male   Female Male   Female Male   Female Male   Female Male   Female
 [81] Male   Female Male   Female Male   Female Male   Female Male   Female
 [89] Male   Female Male   Female Male   Female Male   Female Male   Female
 [97] Male   Female Male   Female
Levels: Male Female
```

Examples 2 and 3

```
> gl(2,50,100,labels=c("Male","Female"))
  [1] Male   Male   Male   Male   Male   Male   Male   Male
  [9] Male   Male   Male   Male   Male   Male   Male   Male
 [17] Male   Male   Male   Male   Male   Male   Male   Male
 [25] Male   Male   Male   Male   Male   Male   Male   Male
 [33] Male   Male   Male   Male   Male   Male   Male   Male
 [41] Male   Male   Male   Male   Male   Male   Male   Male
 [49] Male   Male   Female Female Female Female Female Female
 [57] Female Female Female Female Female Female Female Female
 [65] Female Female Female Female Female Female Female Female
 [73] Female Female Female Female Female Female Female Female
 [81] Female Female Female Female Female Female Female Female
 [89] Female Female Female Female Female Female Female Female
 [97] Female Female Female Female
Levels: Male Female
> gl(10,10)
  [1] 1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3
 [26] 3  3  3  3  3  4  4  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  5
 [51] 6  6  6  6  6  6  6  6  6  6  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8
 [76] 8  8  8  8  8  9  9  9  9  9  9  9  9  9  9  10 10 10 10 10 10 10 10 10 10
Levels: 1 2 3 4 5 6 7 8 9 10
```

Common Functions used with Factors

- Challenge:
  - Vector of ages of voters and a factor showing categorical trait of voters (party affiliation) (Republican, Democrat, Unaffiliated)
  - Need to find the mean ages of groups
- tapply(x,f,g)
  - x is a vector
  - f is a factor (same length as x). More than one factor can be used
  - g is a function

Operation

- Logic of tapply(x,f,g)
  - Temporarily split x into groups according to level of factor
  - Apply g to the resulting sub-vectors and return results

```
> ages<-c(25,26,55,37,21,42)
> affils<-c("R","D","D","R","U","D")
> m<-tapply(ages,affils,mean)
> m
 D  R  U
41 31 21
```

Example… Different factor

```
> ages<-c(25,26,55,37,21,42)
> gender<-c("M","M","F","F","F","M")
> g<-tapply(ages,gender,mean)
> g
       F        M
37.66667 31.00000
```

Multiple Factors

```
> ages<-c(25,26,55,37,21,42)
> affils<-c("R","D","D","R","U","D")
> gender<-c("M","M","F","F","F","M")
>
> m1<-tapply(ages,list(affils,gender),mean)
> m1
   F  M
D 55 34
R 37 25
U 21 NA
```

```
> m1
   F  M
D 55 34
R 37 25
U 21 NA
> m1[1,]
 F  M
55 34
> m1["D",]
 F  M
55 34
```

split() function

- In contrast to tapply, the split function just splits vectors into groups.
- The input is a vector and factor(s)
- The output is a list

```
> ages
[1] 25 26 55 37 21 42
> gender
[1] "M" "M" "F" "F" "F" "M"
> g<-split(ages,gender)
> str(g)
List of 2
 $ F: num [1:3] 55 37 21
 $ M: num [1:3] 25 26 42
```

```
> ages
[1] 25 26 55 37 21 42
> affils
[1] "R" "D" "D" "R" "U" "D"
> a<-split(ages,affils)
> str(a)
List of 3
 $ D: num [1:3] 26 55 42
 $ R: num [1:2] 25 37
 $ U: num 21
```

Split can work to find the indices of factors in vectors

```
> gender
[1] "M" "M" "F" "F" "F" "M"
> split(1:length(gender),gender)
$F
[1] 3 4 5

$M
[1] 1 2 6
```

Data Frame Example

- Use tapply to produce a category table for the mean salary of male|female for under|over 25

```
> s<-data.frame(gender=c("Male","Male","Female","Male","Female","Female"),
+               age=c(47,59,21,32,33,24),
+               income=c(55000,88000,32450,76500,123000,45650))
> s
  gender age income
1   Male  47  55000
2   Male  59  88000
3 Female  21  32450
4   Male  32  76500
5 Female  33 123000
6 Female  24  45650
```

Step 1: Create the new category

```
> s$age.status<-factor(ifelse(s$age>25,">25","<=25"))
> s
  gender age income age.status
1   Male  47  55000        >25
2   Male  59  88000        >25
3 Female  21  32450       <=25
4   Male  32  76500        >25
5 Female  33 123000        >25
6 Female  24  45650       <=25
```

2. Apply factors (as list) to relevant column

```
> s
  gender age income age.status
1   Male  47  55000        >25
2   Male  59  88000        >25
3 Female  21  32450       <=25
4   Male  32  76500        >25
5 Female  33 123000        >25
6 Female  24  45650       <=25
> t1<-tapply(s$income,list(s$gender,s$age.status),mean)
> t1
          <=25       >25
Female 39050 123000.00
Male      NA  73166.67
```

Challenge 4.2

- Create a data frame of student results in different examinations (CT201, CT202, CT203) and programme (2IF1, 2CS1)
- Write a tapply() function to
  - Calculate the mean for each examination
  - Calculate the mean for each examination by programme code (category table)

ADDITIONAL MATERIAL

Higher Dimensional Arrays

- A matrix is a two-dimensional structure
- For example
  - Test Score 1, Test Score 2
  - Each student is a row of data

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

```
> test1<-matrix(c(46,54,71,75,65,79),nrow=3)
> test1
     [,1] [,2]
[1,]   46   75
[2,]   54   65
[3,]   71   79
```

However… more than one test?

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

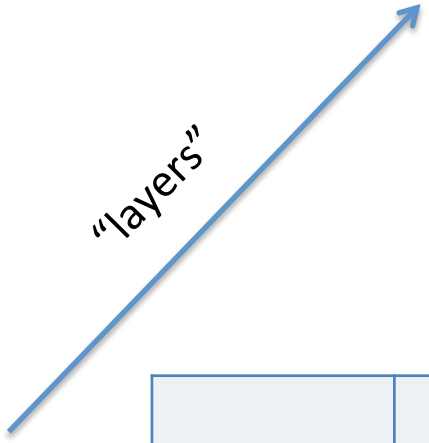
	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

```
> test1<-matrix(c(46,54,71,75,65,79),nrow=3)
> test1
     [,1] [,2]
[1,]   46   75
[2,]   54   65
[3,]   71   79
```

```
> test2<-matrix(c(56,44,61,85,55,69),nrow=3)
> test2
     [,1] [,2]
[1,]   56   85
[2,]   44   55
[3,]   61   69
```

3-Dimensional Structure

```
> results<-array(data=c(test1,test2),dim=c(3,2,2))
> dim(results)
[1] 3 2 2
```



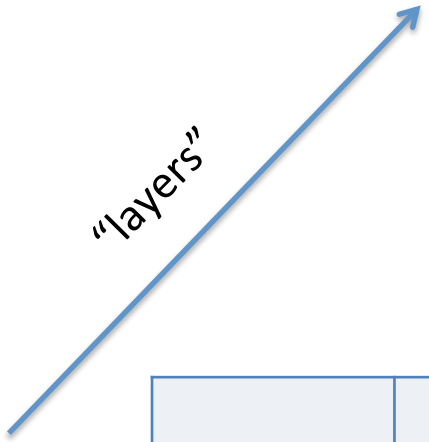
	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

```
> results[,,2]
     [,1] [,2]
[1,]   56   85
[2,]   44   55
[3,]   61   69
```

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

```
> results[,,1]
     [,1] [,2]
[1,]   46   75
[2,]   54   65
[3,]   71   79
```

Processing Data

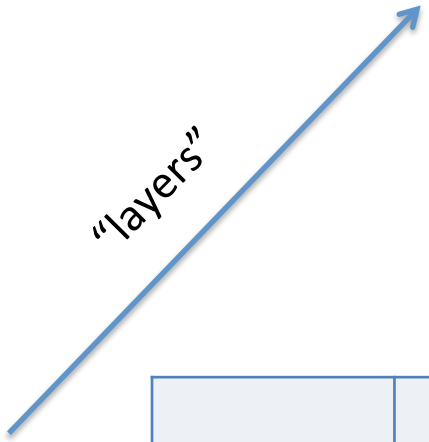


	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

```
> results[1,,]
     [,1] [,2]
[1,]   46   56
[2,]   75   85
```

Processing Data

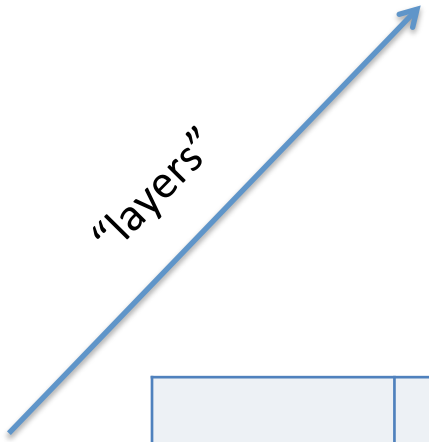


	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

```
> results[,1,]
     [,1] [,2]
[1,]   46   56
[2,]   54   44
[3,]   71   61
```

Processing Data



	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

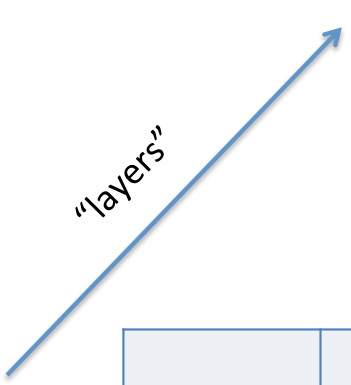
	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79

```
> results[,,1]
     [,1] [,2]
[1,]   46   75
[2,]   54   65
[3,]   71   79
```

```
> results[,,2]
     [,1] [,2]
[1,]   56   85
[2,]   44   55
[3,]   61   69
```

Challenge 4.2

- Results for Student 3, Test 1 (both weeks)
- Student 1 average for Test 2
- Average score in Test 1
- Average Score in Test 2



	[,1]	[,2]
[1,]	56	85
[2,]	44	55
[3,]	61	69

	[,1]	[,2]
[1,]	46	75
[2,]	54	65
[3,]	71	79