# CT5102: Programming for Data Analytics

# Week 3:
# Matrices and Data Frames

https://github.com/JimDuggan/CT5102

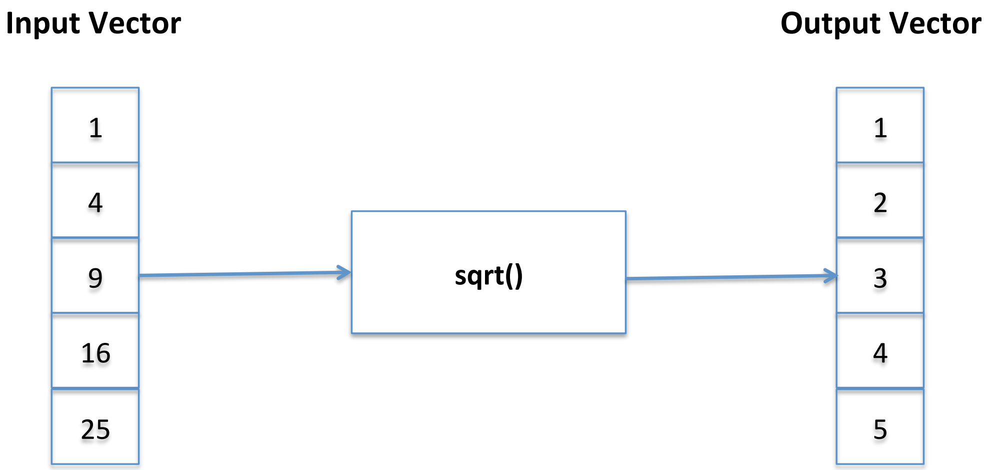
Dr. Jim Duggan,

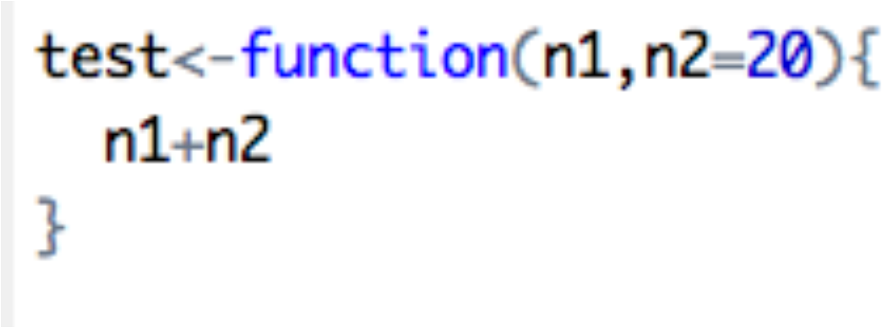
Information Technology,

School of Engineering & Informatics

# One-slide summary to date…

- **Vectors** – fundamental data type in R. *Vectorization*. “No loops required!”
- **Functions** – modularity, divide and conquer, reuse. *Apply family*
- **Lists** – complex data structures



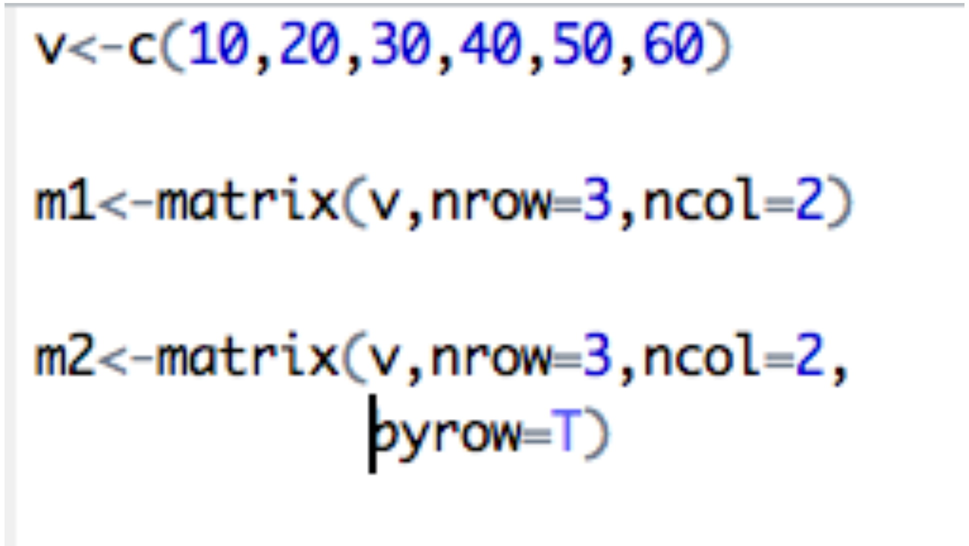


```
test<-function(n1,n2=20){
  n1+n2
}
```

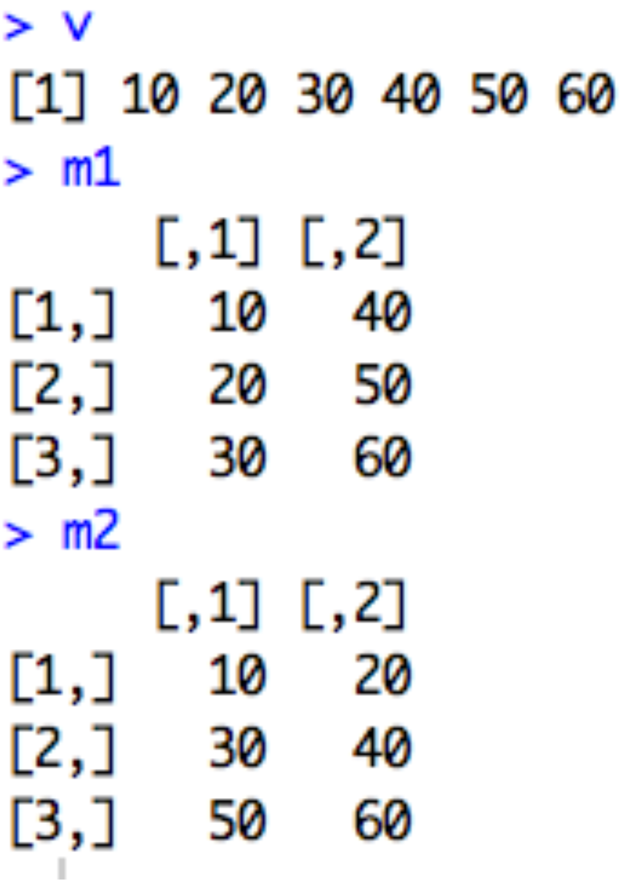
# Matrices

- A matrix is a vector that has a number of rows and a number of columns, as well as having defined modes (Matloff 2009).
- Matrix subscripts, similar to vectors, commence at 1, and there are useful operations to access row and column elements.
- A matrix can be initialized from a vector, where the numbers of rows and columns are specified.
- R stores matrices by column-major order, and by default matrices are filled in this manner.

# Examples

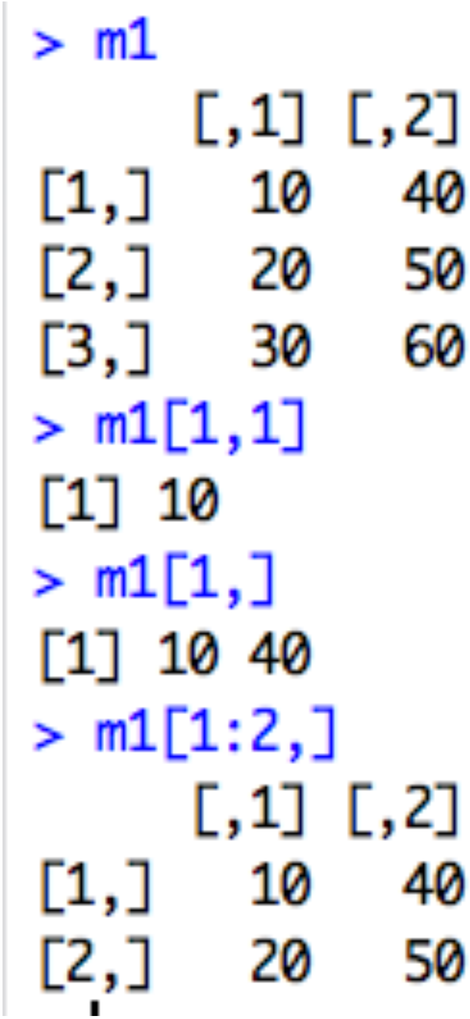


```
v<-c(10,20,30,40,50,60)

m1<-matrix(v,nrow=3,ncol=2)

m2<-matrix(v,nrow=3,ncol=2,
              byrow=T)
```

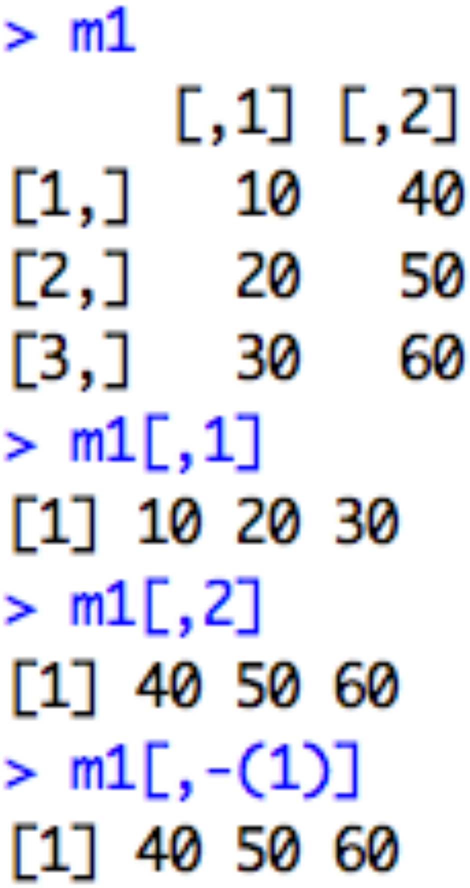


```
> v
[1] 10 20 30 40 50 60
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> m2
     [,1] [,2]
[1,]   10   20
[2,]   30   40
[3,]   50   60
```

# Accessing Elements

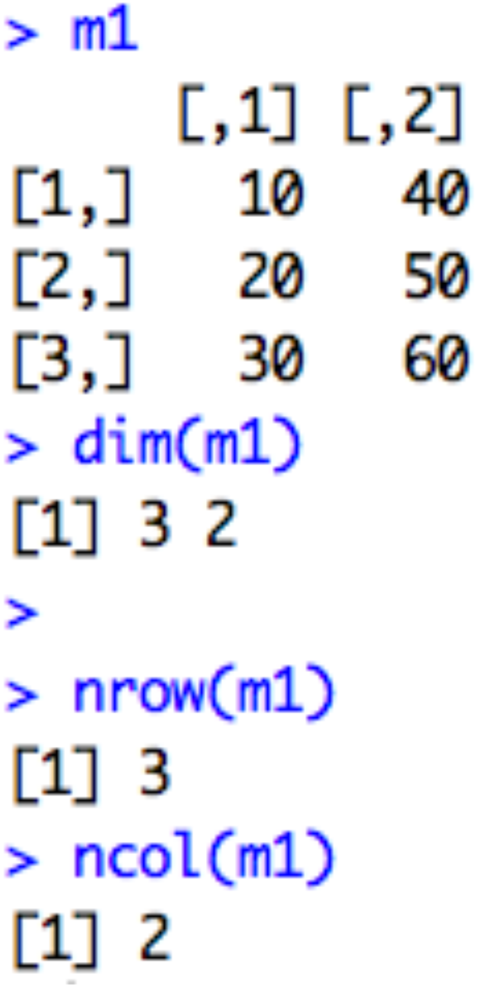


```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> m1[1,1]
[1] 10
> m1[1,]
[1] 10 40
> m1[1:2,]
     [,1] [,2]
[1,]   10   40
[2,]   20   50
```

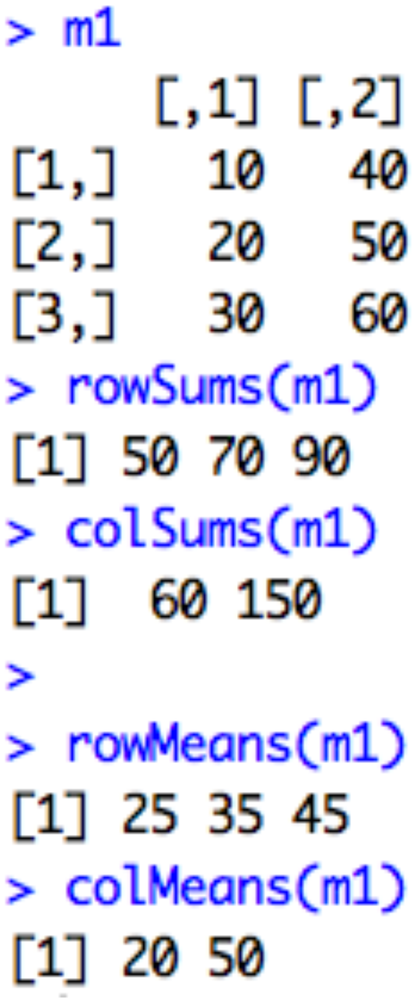


```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> m1[,1]
[1] 10 20 30
> m1[,2]
[1] 40 50 60
> m1[,-(1)]
[1] 40 50 60
```

# Useful Functions for Matrices



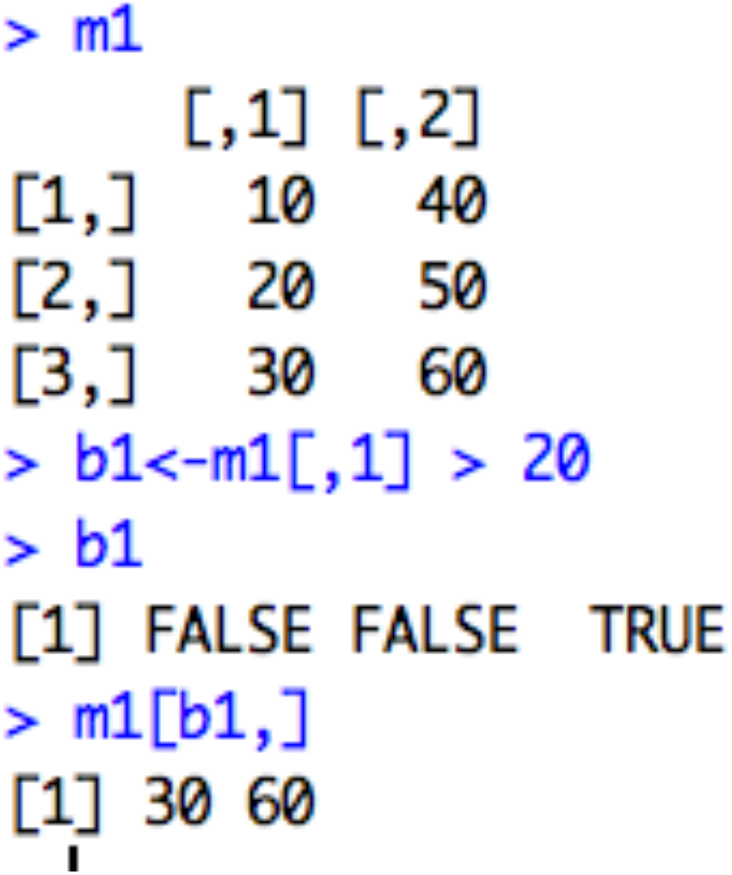
```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> dim(m1)
[1] 3 2
>
> nrow(m1)
[1] 3
> ncol(m1)
[1] 2
```



```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> rowSums(m1)
[1] 50 70 90
> colSums(m1)
[1]  60 150
>
> rowMeans(m1)
[1] 25 35 45
> colMeans(m1)
[1] 20 50
```

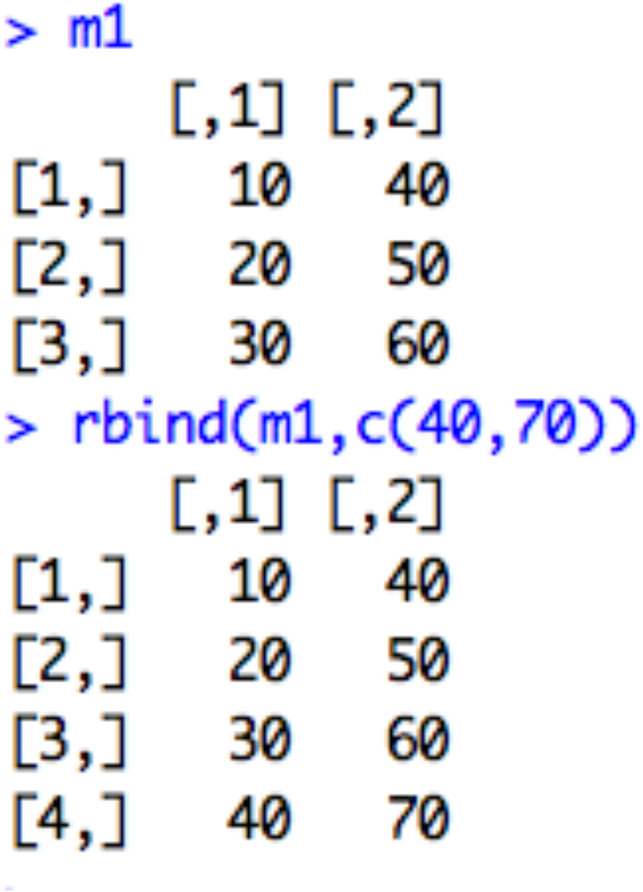
# Filtering Matrices

- Filtering can also be performed on matrices.
- For example, if a query is required to find all rows that have *column 1 values greater than 20*.
- First a logical vector could be applied to the full column with the specified condition

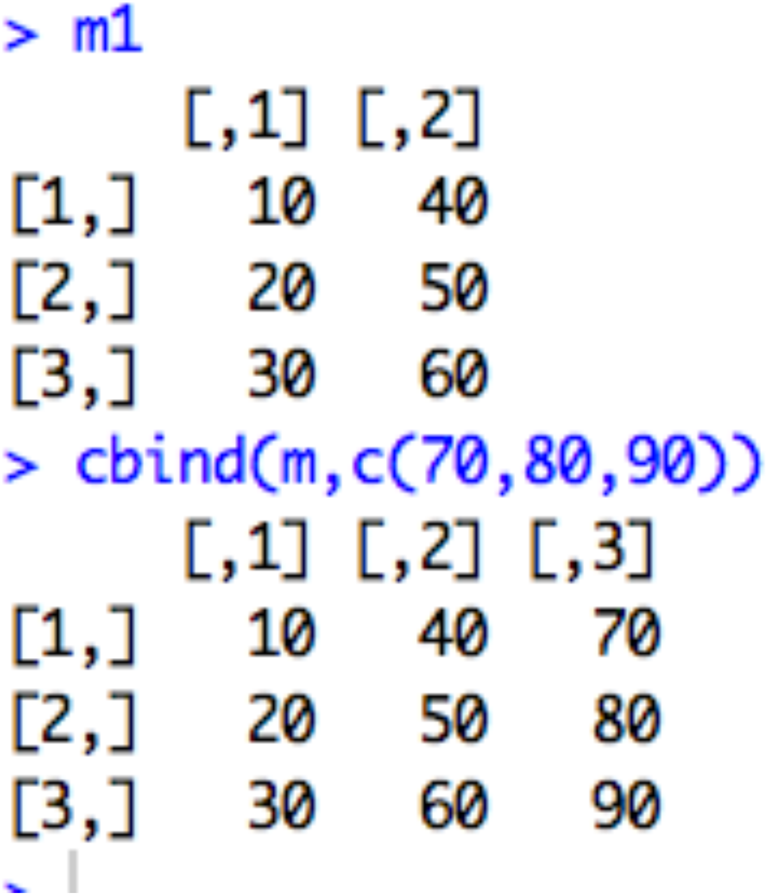


```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> b1<-m1[,1] > 20
> b1
[1] FALSE FALSE  TRUE
> m1[b1,]
[1] 30 60
```

# Appending rows and columns



```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> rbind(m1,c(40,70))
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
[4,]   40   70
```



```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> cbind(m,c(70,80,90))
     [,1] [,2] [,3]
[1,]   10   40   70
[2,]   20   50   80
[3,]   30   60   90
```

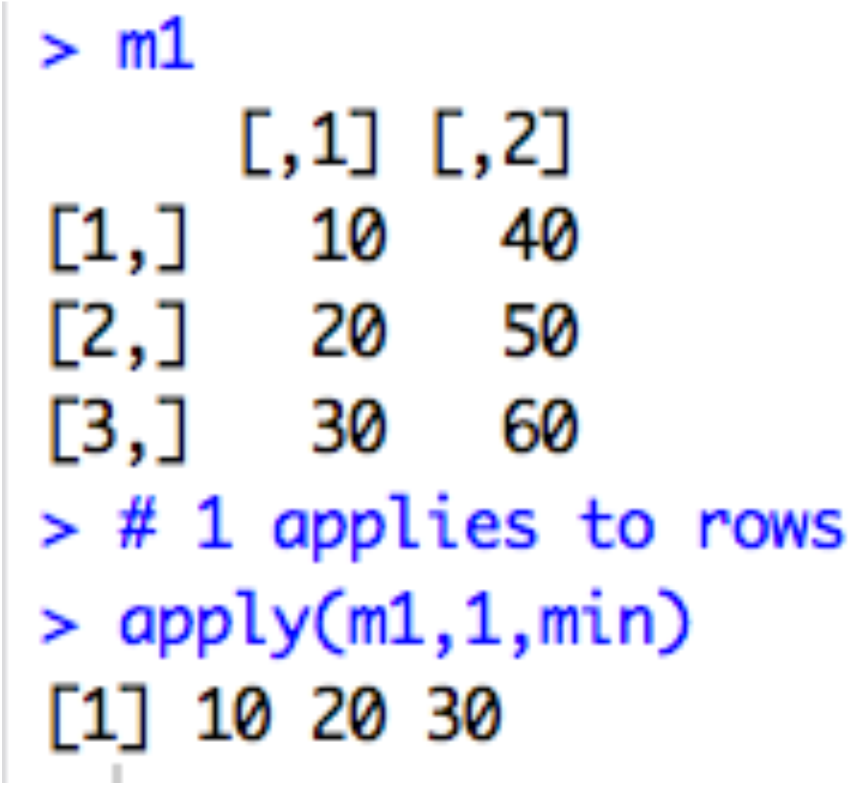
# Sample Matrix Operations

| **Operator or Function** | **Description** |
|---|---|
| A * B | Element-wise multiplication |
| A / B | Element-wise division |
| A %*% B | Matrix multiplication |
| t(A) | Transpose of A |
| e<-eigen(A) | List of eigenvalues and eigenvectors for matrix A |

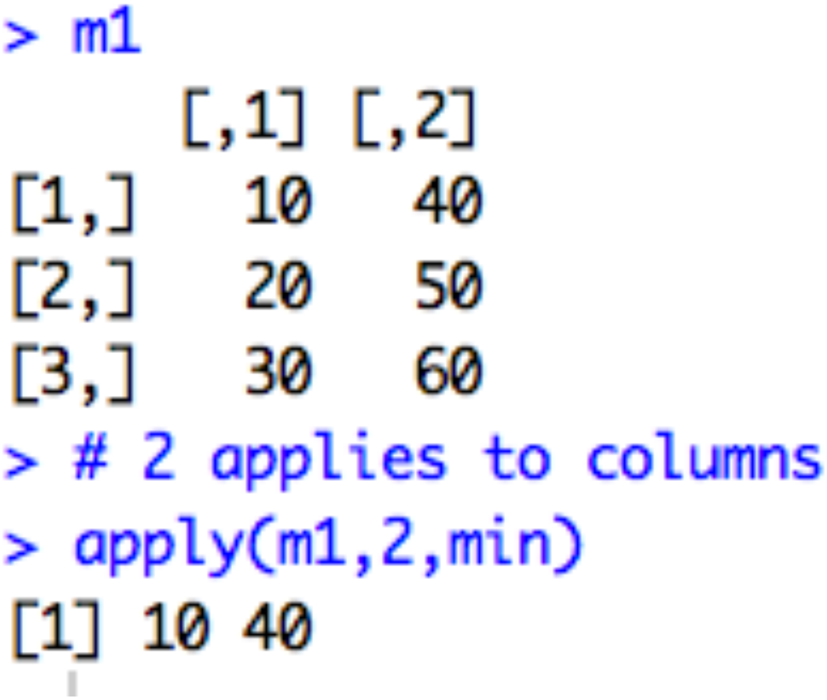
# apply() function

- The **apply()** function can be used to process rows and columns for a matrix, and the general form of this function (Matloff 2009) is **apply(m, dimcode, f, fargs)**, where:
  - **m** is the target matrix
  - **dimcode** identifies whether it's a row or column target. The number 1 applies to rows, whereas 2 applies to columns
  - **f** is the function to be called
  - **fargs** are the optional set of arguments that can be applied to the function **f**.

# apply() in action

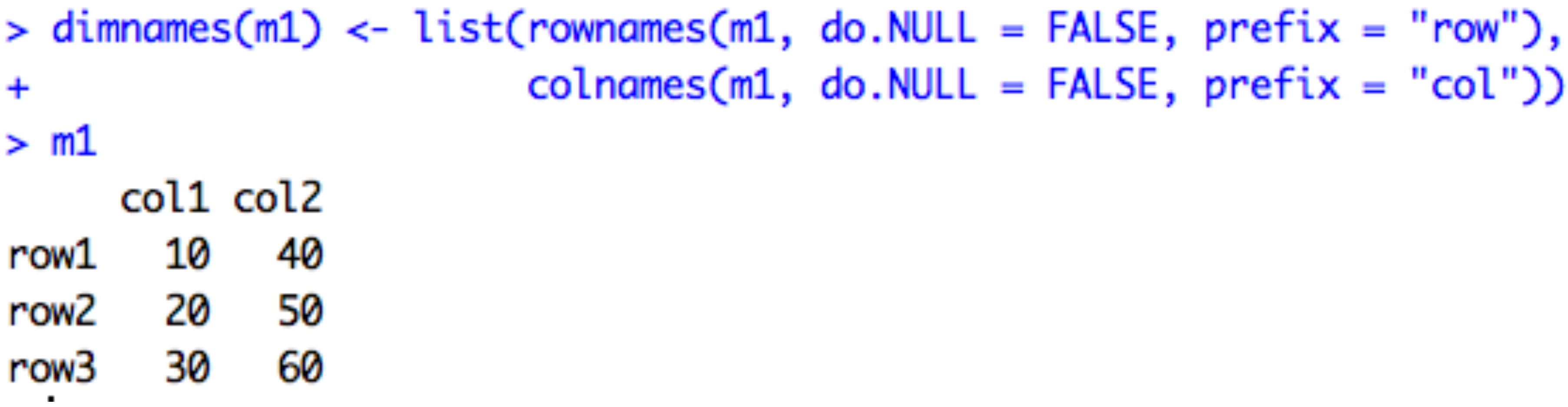


```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> # 1 applies to rows
> apply(m1,1,min)
[1] 10 20 30
```



```
> m1
     [,1] [,2]
[1,]   10   40
[2,]   20   50
[3,]   30   60
> # 2 applies to columns
> apply(m1,2,min)
[1] 10 40
```

# Row and Column Names



```
> dimnames(m1) <- list(rownames(m1, do.NULL = FALSE, prefix = "row"),
+                     colnames(m1, do.NULL = FALSE, prefix = "col"))
> m1
     col1 col2
row1   10   40
row2   20   50
row3   30   60
```

# Challenge 3.1

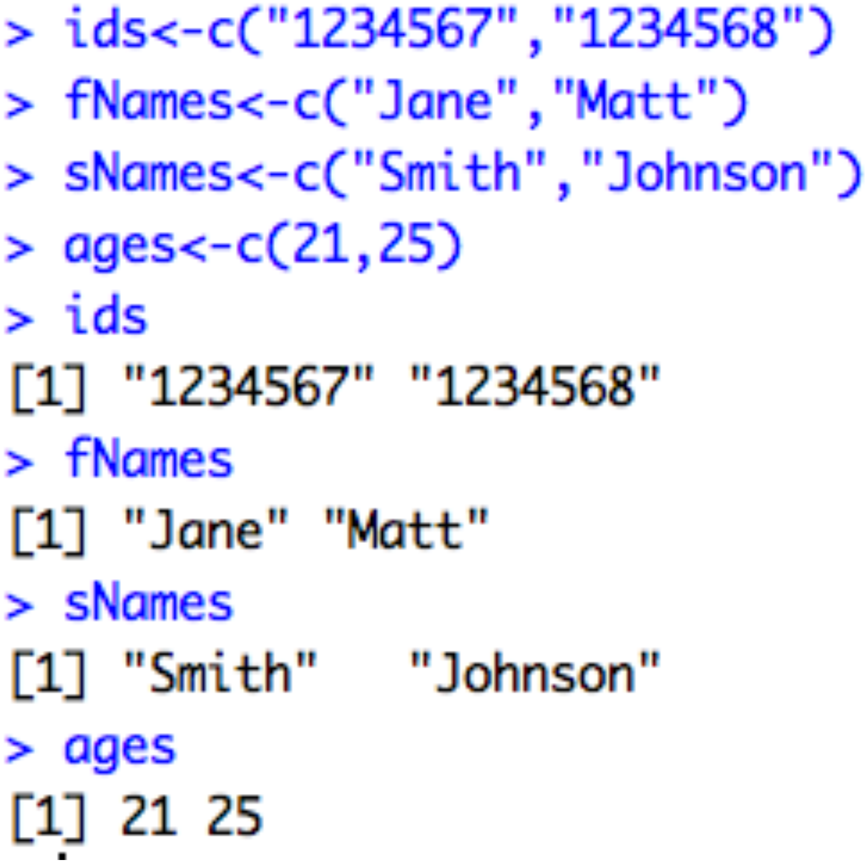
- Create a 5x5 square matrix of random uniform numbers
- Add a new column that contains the maximum value in each row
- Add a new row that contains the maximum value in each column

# Challenge 3.2
# Customer Preference Matrix

- Write a function that creates an nxm matrix of customers and products, and for each customer, assigns a random preference vector showing their preference for a product.
- Customers should express a preference value for a random x% of products
- The function default values should be 10, 10 and 20%
- Preferences should be generated on a uniform scale from 1 to 5 (representing 1 to 5 stars)
- Columns should be named Pr1,Pr2,…,PrN
- Rows should be names Cust1, Cust2, …, CustN

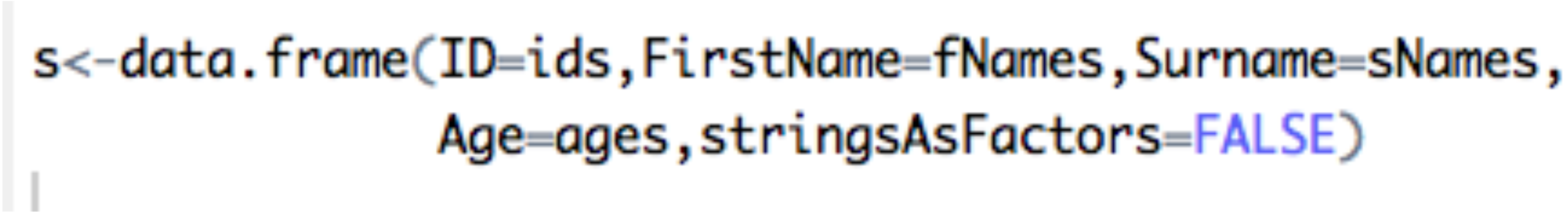
# Data Frames

- A data frame is similar to a matrix, as it has a two-dimensional rows and columns structure
- Differs from a matrix in that each column can have a different mode (Matloff 2009).
- Can be viewed as a set of vectors, organised into a column format

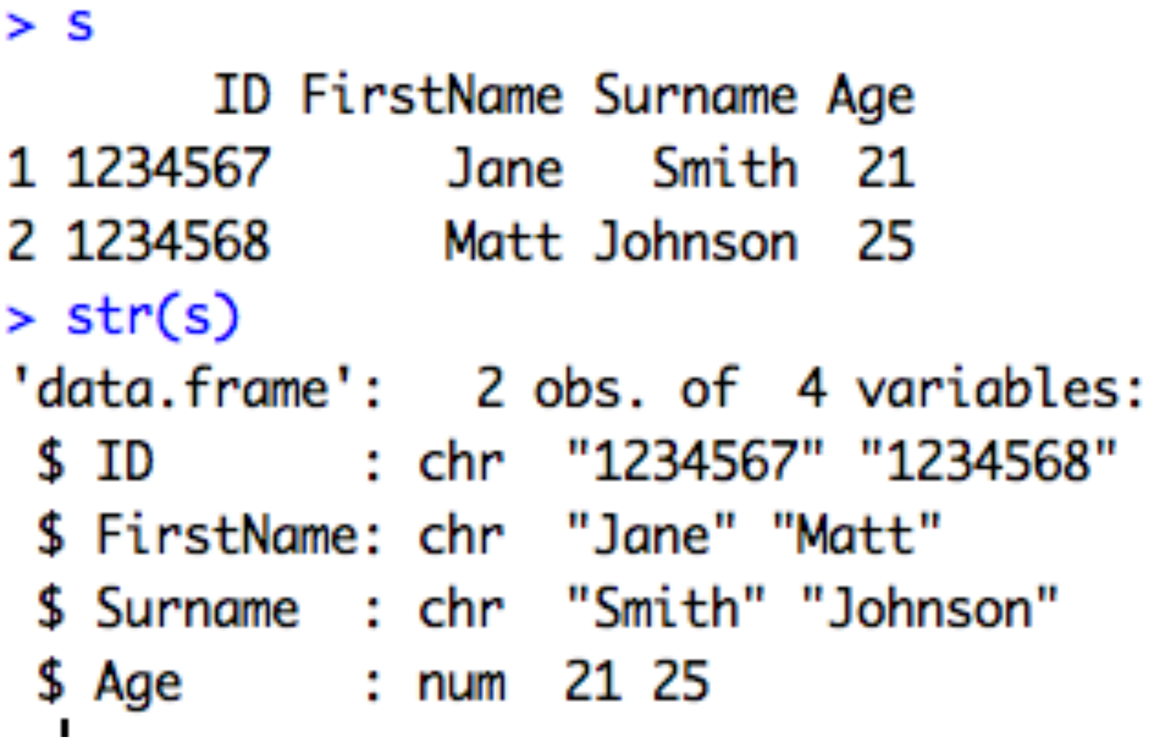


```
> ids<-c("1234567","1234568")
> fNames<-c("Jane","Matt")
> sNames<-c("Smith","Johnson")
> ages<-c(21,25)
> ids
[1] "1234567" "1234568"
> fNames
[1] "Jane" "Matt"
> sNames
[1] "Smith"   "Johnson"
> ages
[1] 21 25
```

# Creating a Data Frame

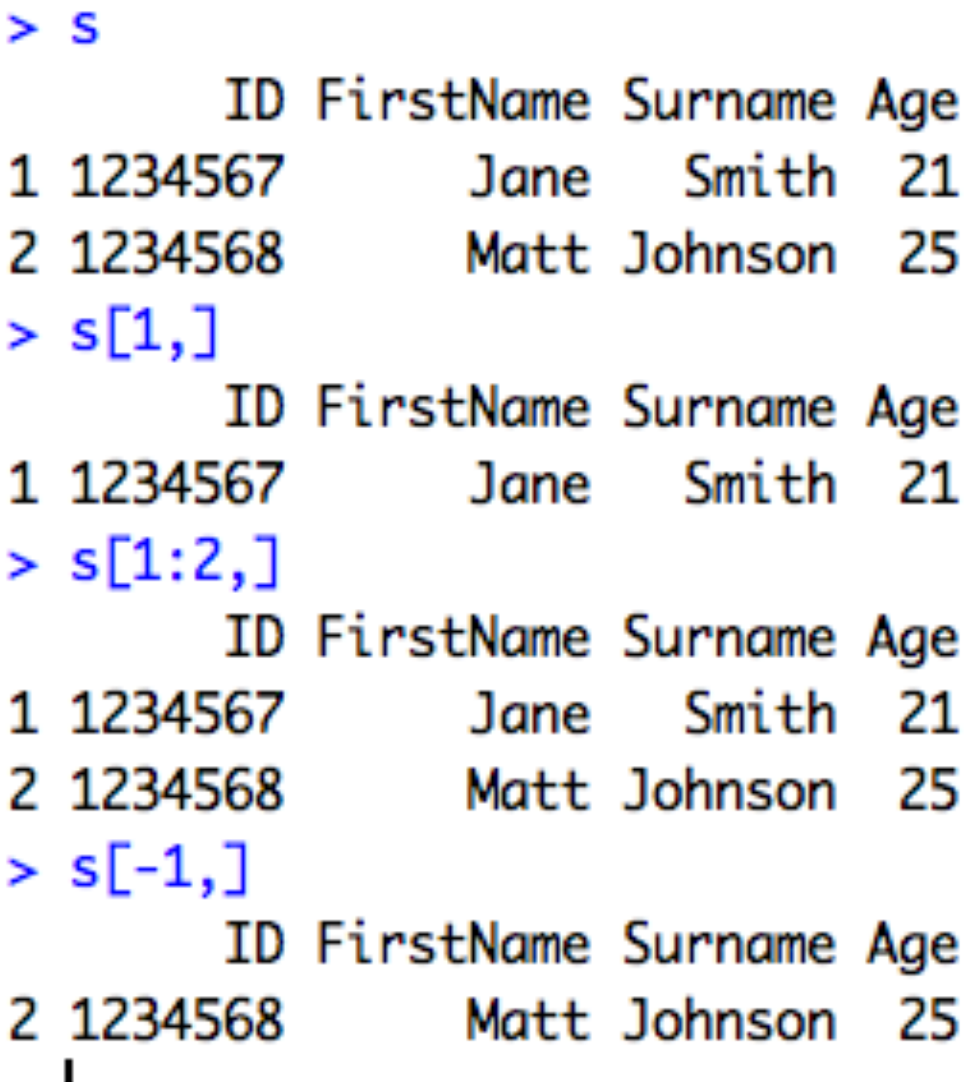


```
s<-data.frame(ID=ids,FirstName=fNames,Surname=sNames,
              Age=ages,stringsAsFactors=FALSE)
```



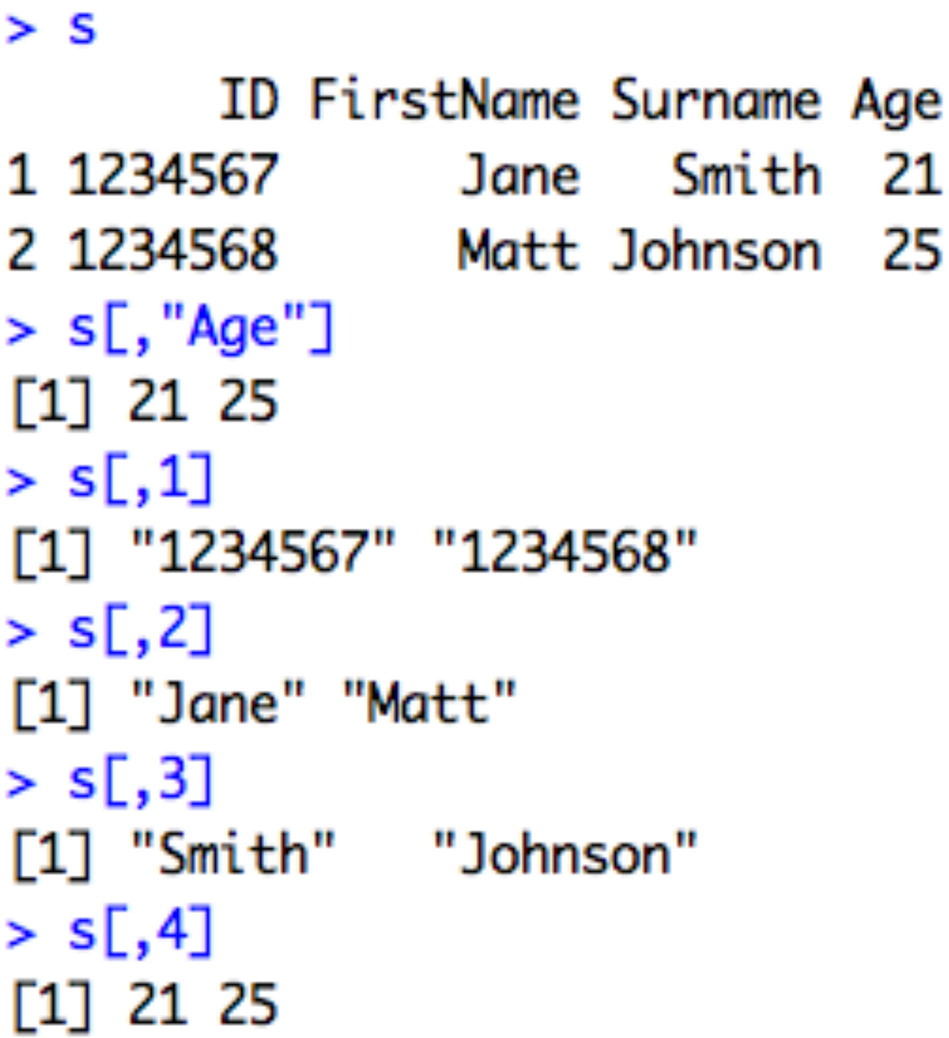
```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> str(s)
'data.frame':   2 obs. of  4 variables:
 $ ID       : chr  "1234567" "1234568"
 $ FirstName: chr  "Jane" "Matt"
 $ Surname  : chr  "Smith" "Johnson"
 $ Age      : num  21 25
```

# Accessing Row Data (matrix method)



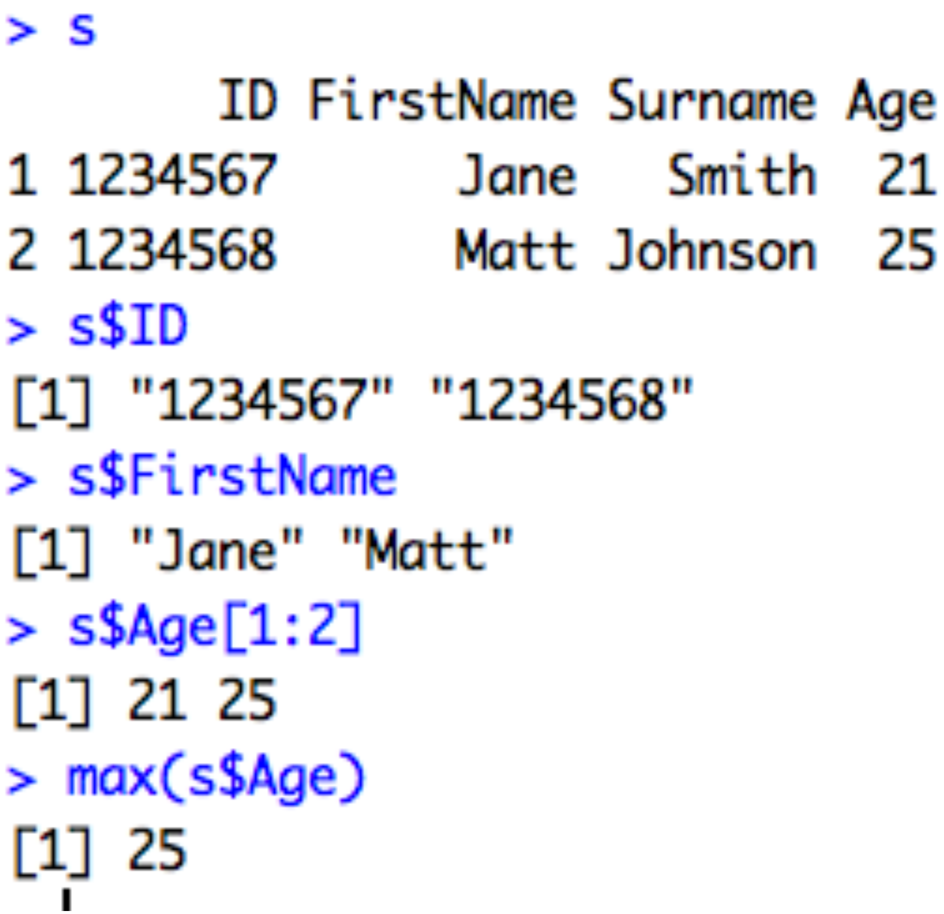
```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> s[1,]
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
> s[1:2,]
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> s[-1,]
       ID FirstName Surname Age
2 1234568      Matt Johnson  25
```

# Accessing Column Data (matrix method)



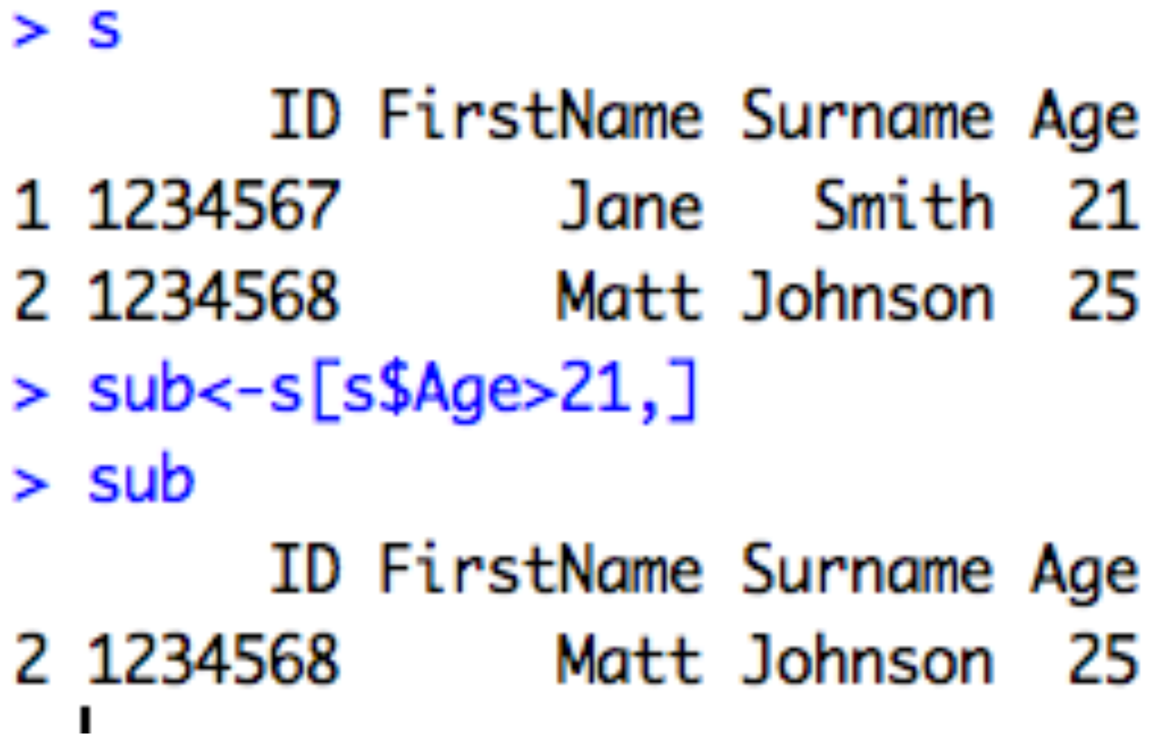
```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> s[,"Age"]
[1] 21 25
> s[,1]
[1] "1234567" "1234568"
> s[,2]
[1] "Jane" "Matt"
> s[,3]
[1] "Smith"   "Johnson"
> s[,4]
[1] 21 25
```

# Accessing Column Data (using tags)



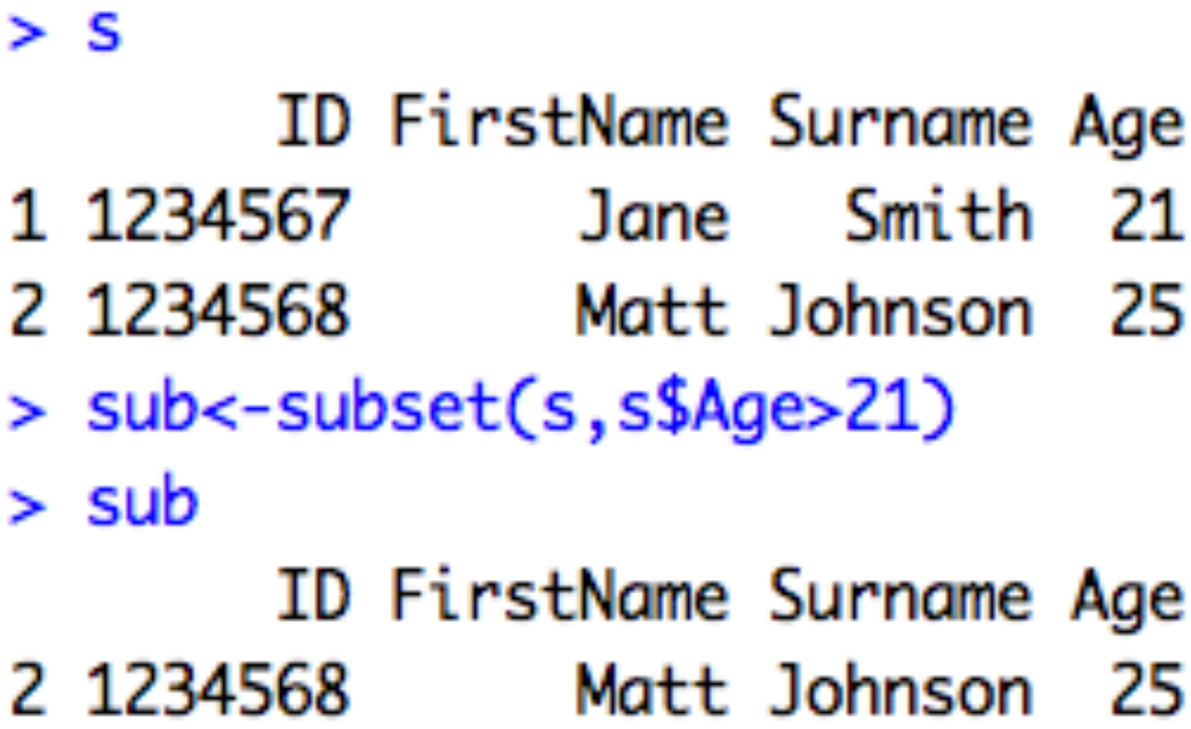
```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> s$ID
[1] "1234567" "1234568"
> s$FirstName
[1] "Jane" "Matt"
> s$Age[1:2]
[1] 21 25
> max(s$Age)
[1] 25
```

# Subsetting Row Data (Matrix method]



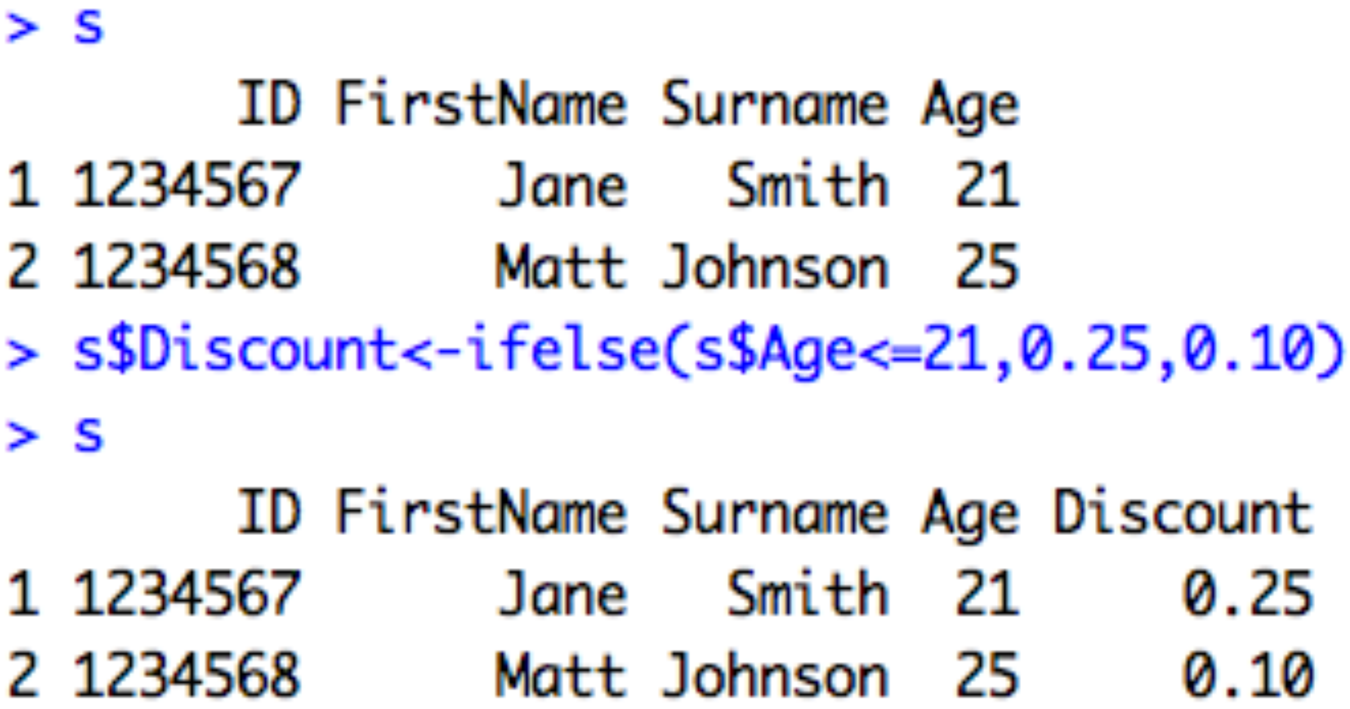
```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> sub<-s[s$Age>21,]
> sub
       ID FirstName Surname Age
2 1234568      Matt Johnson  25
```

# subset() function



```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> sub<-subset(s,s$Age>21)
> sub
       ID FirstName Surname Age
2 1234568      Matt Johnson  25
```

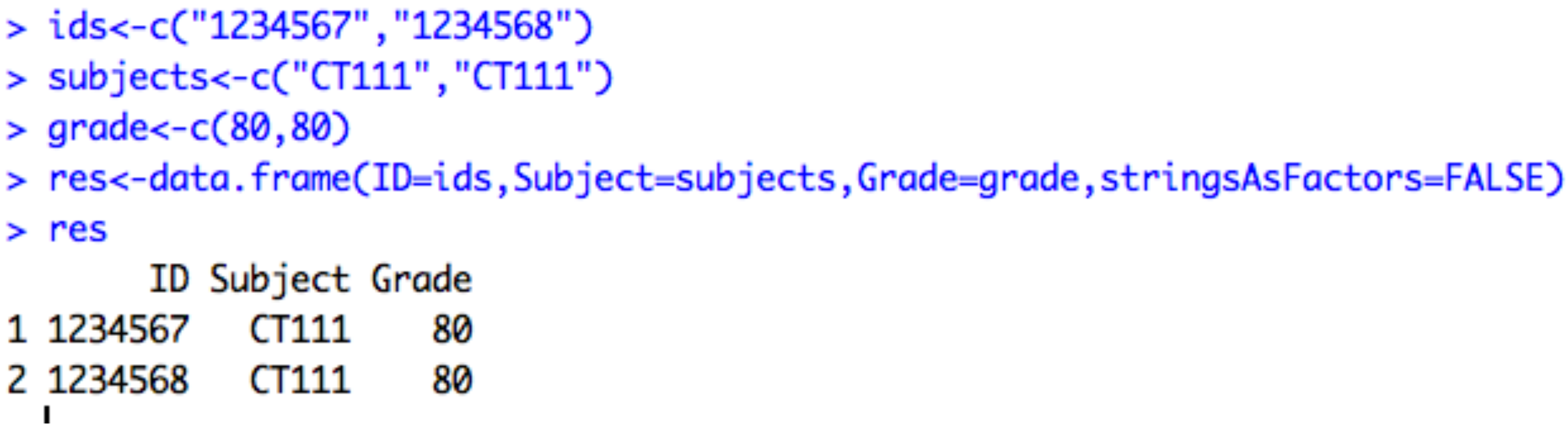
# Adding extra information to a data frame



```
> s
       ID FirstName Surname Age
1 1234567      Jane   Smith  21
2 1234568      Matt Johnson  25
> s$Discount<-ifelse(s$Age<=21,0.25,0.10)
> s
       ID FirstName Surname Age Discount
1 1234567      Jane   Smith  21     0.25
2 1234568      Matt Johnson  25     0.10
```

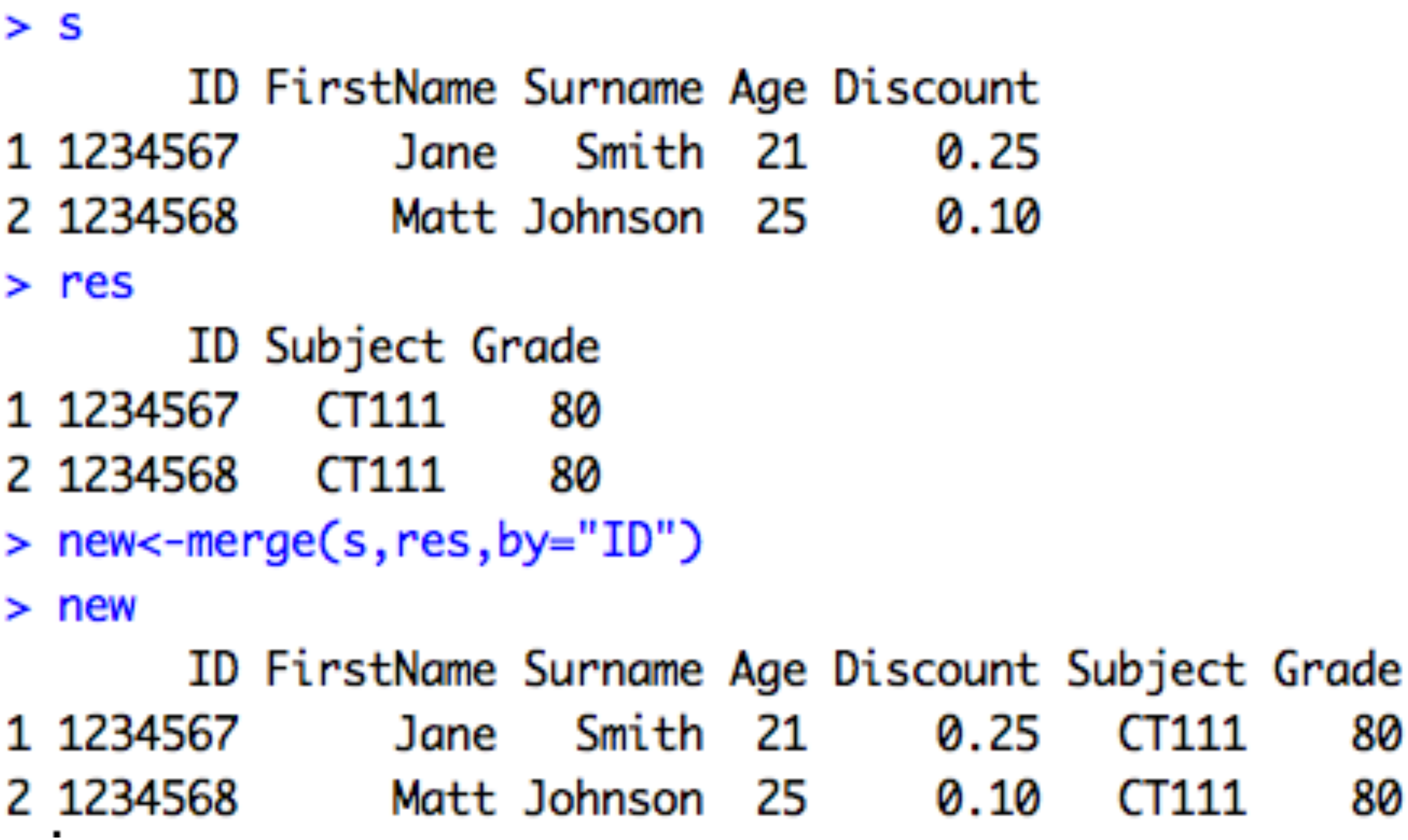
# Merging Data Frames

- For data analytics, opportunities often arise by merging different data sets into a single data frame, and the **merge()** function facilitates this
- In our student example, we could have a second data frame that stores examination results.



```
> ids<-c("1234567","1234568")
> subjects<-c("CT111","CT111")
> grade<-c(80,80)
> res<-data.frame(ID=ids,Subject=subjects,Grade=grade,stringsAsFactors=FALSE)
> res
       ID Subject Grade
1 1234567   CT111    80
2 1234568   CT111    80
```

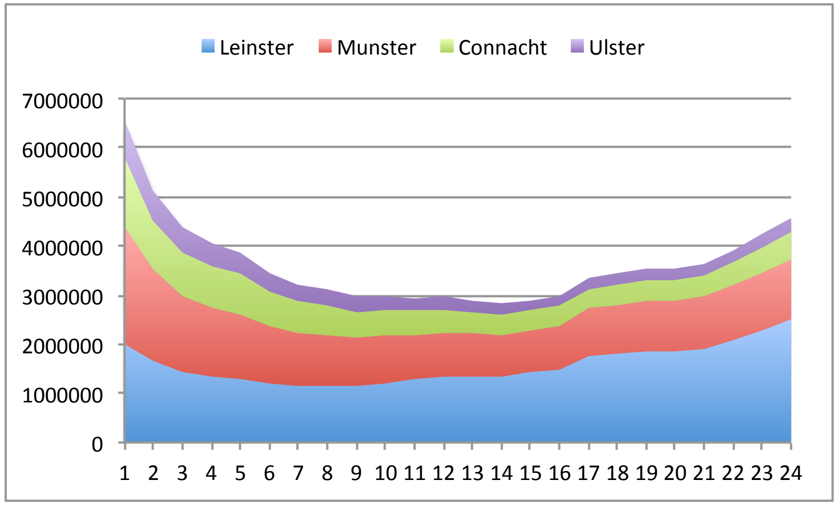
# Merging Code

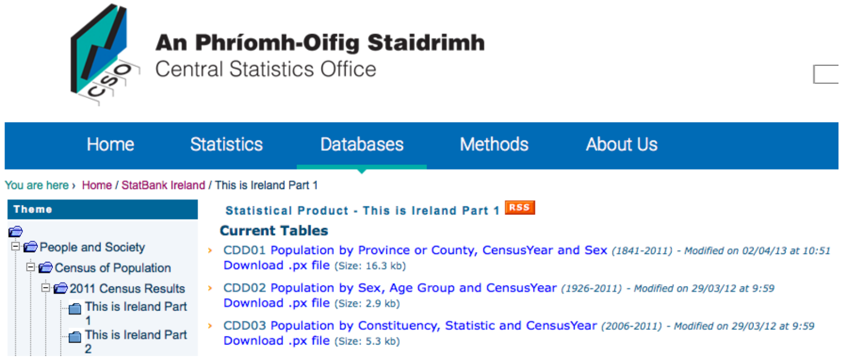


```
> s
       ID FirstName Surname Age Discount
1 1234567      Jane   Smith  21     0.25
2 1234568      Matt Johnson  25     0.10
> res
       ID Subject Grade
1 1234567   CT111    80
2 1234568   CT111    80
> new<-merge(s,res,by="ID")
> new
       ID FirstName Surname Age Discount Subject Grade
1 1234567      Jane   Smith  21     0.25   CT111    80
2 1234568      Matt Johnson  25     0.10   CT111    80
```

# Accessing Real Data Sets

| Year | Leinster | Munster | Connacht | Ulster |
|---|---|---|---|---|
| 1841 | 1973731 | 2396161 | 1418859 | 740048 |
| 1851 | 1672738 | 1857736 | 1010031 | 571052 |
| 1861 | 1457635 | 1513558 | 913135 | 517783 |
| 1871 | 1339451 | 1393485 | 846213 | 474038 |
| 1881 | 1278989 | 1331115 | 821657 | 438259 |
| 1891 | 1187760 | 1172402 | 724774 | 383758 |
| 1901 | 1152829 | 1076188 | 646932 | 345874 |
| 1911 | 1162044 | 1035495 | 610984 | 331165 |
| 1926 | 1149092 | 969902 | 552907 | 300091 |
| 1936 | 1220411 | 942272 | 525468 | 280269 |
| 1946 | 1281117 | 917306 | 492797 | 263887 |
| 1951 | 1336576 | 898870 | 471895 | 253252 |
| 1956 | 1338942 | 877238 | 446221 | 235863 |
| 1961 | 1332149 | 849203 | 419465 | 217524 |
| 1966 | 1414415 | 859334 | 401950 | 208303 |
| 1971 | 1498140 | 882002 | 390902 | 207204 |
| 1979 | 1743861 | 979819 | 418500 | 226037 |
| 1981 | 1790521 | 998315 | 424410 | 230159 |
| 1986 | 1852649 | 1020577 | 431409 | 236008 |
| 1991 | 1860949 | 1009533 | 423031 | 232206 |
| 1996 | 1924702 | 1033903 | 433231 | 234251 |
| 2002 | 2105579 | 1100614 | 464296 | 246714 |
| 2006 | 2295123 | 1173340 | 504121 | 267264 |
| 2011 | 2504814 | 1246088 | 542547 | 294803 |





You are here › Home / StatBank Ireland / This is Ireland Part 1

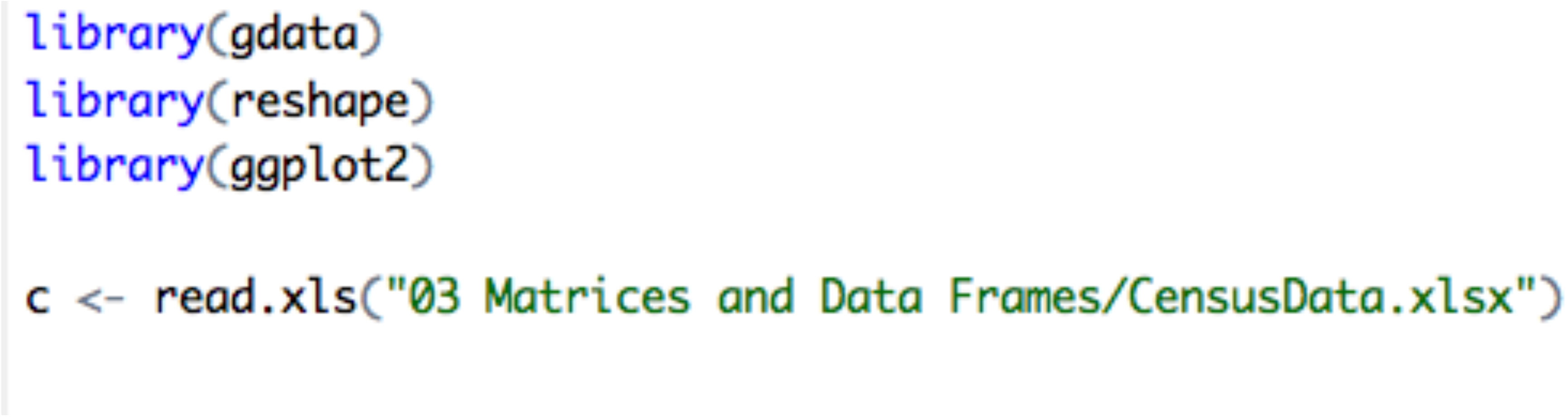
Statistical Product - This is Ireland Part 1

**Current Tables**

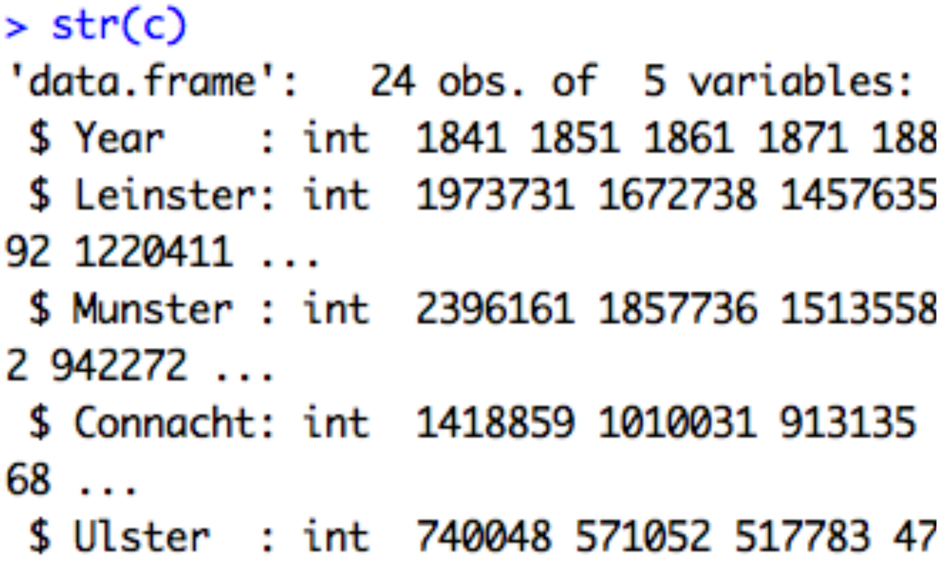
- CDD01 Population by Province or County, CensusYear and Sex *(1841-2011) - Modified on 02/04/13 at 10:51*
  Download .px file (Size: 16.3 kb)
- CDD02 Population by Sex, Age Group and CensusYear *(1926-2011) - Modified on 29/03/12 at 9:59*
  Download .px file (Size: 2.9 kb)
- CDD03 Population by Constituency, Statistic and CensusYear *(2006-2011) - Modified on 29/03/12 at 9:59*
  Download .px file (Size: 5.3 kb)

http://www.cso.ie/px/pxeirestat/Statire/SelectVarVal/Define.asp?maintable=CDD01&PLanguage=0

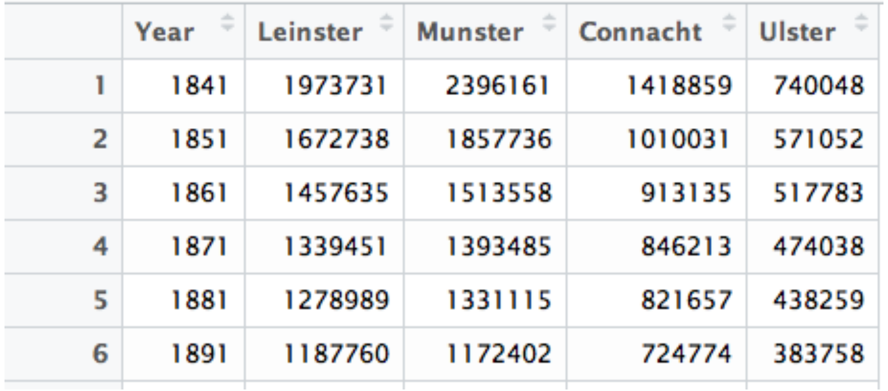
# Opening an Excel File



```
library(gdata)
library(reshape)
library(ggplot2)

c <- read.xls("03 Matrices and Data Frames/CensusData.xlsx")
```



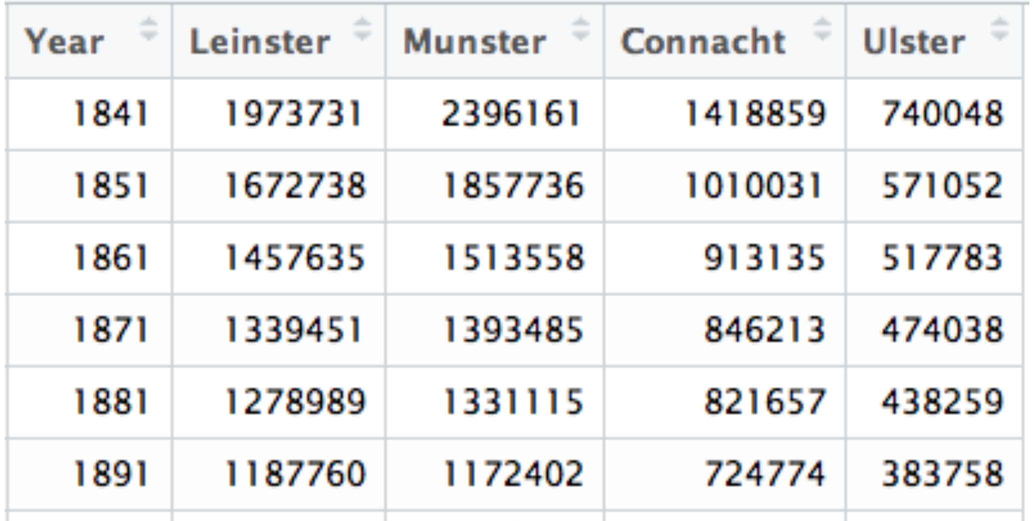
```
> str(c)
'data.frame':	24 obs. of  5 variables:
 $ Year     : int  1841 1851 1861 1871 188
 $ Leinster: int  1973731 1672738 1457635
92 1220411 ...
 $ Munster : int  2396161 1857736 1513558
2 942272 ...
 $ Connacht: int  1418859 1010031 913135
68 ...
 $ Ulster   : int  740048 571052 517783 47
```



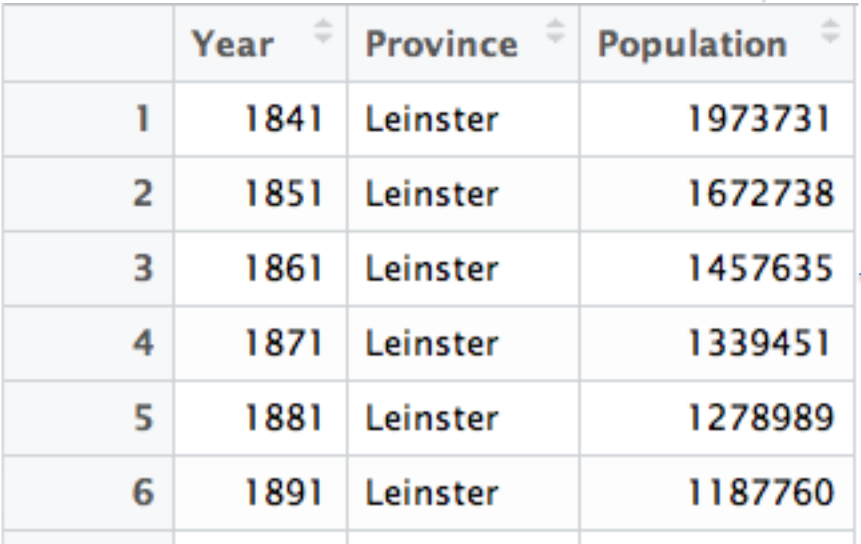
| | Year | Leinster | Munster | Connacht | Ulster |
|---|---|---|---|---|---|
| 1 | 1841 | 1973731 | 2396161 | 1418859 | 740048 |
| 2 | 1851 | 1672738 | 1857736 | 1010031 | 571052 |
| 3 | 1861 | 1457635 | 1513558 | 913135 | 517783 |
| 4 | 1871 | 1339451 | 1393485 | 846213 | 474038 |
| 5 | 1881 | 1278989 | 1331115 | 821657 | 438259 |
| 6 | 1891 | 1187760 | 1172402 | 724774 | 383758 |

# Data format inflexible

- Plotting in R
- Aggregating the data
- Processing data



| Year | Leinster | Munster | Connacht | Ulster |
|---|---|---|---|---|
| 1841 | 1973731 | 2396161 | 1418859 | 740048 |
| 1851 | 1672738 | 1857736 | 1010031 | 571052 |
| 1861 | 1457635 | 1513558 | 913135 | 517783 |
| 1871 | 1339451 | 1393485 | 846213 | 474038 |
| 1881 | 1278989 | 1331115 | 821657 | 438259 |
| 1891 | 1187760 | 1172402 | 724774 | 383758 |

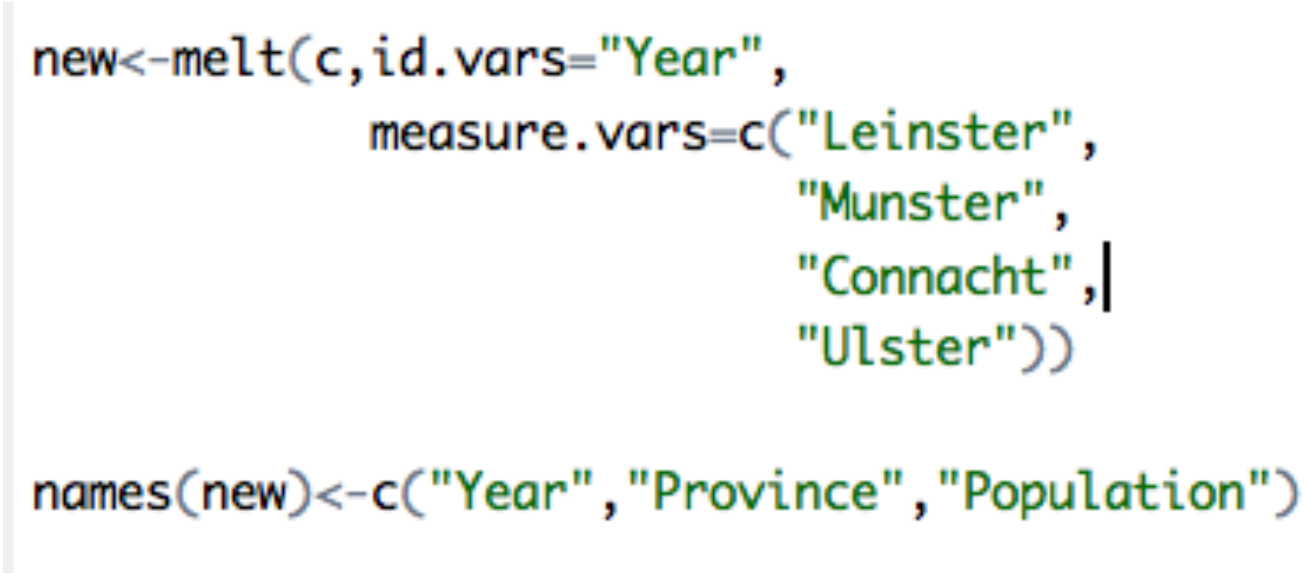


| | Year | Province | Population |
|---|---|---|---|
| 1 | 1841 | Leinster | 1973731 |
| 2 | 1851 | Leinster | 1672738 |
| 3 | 1861 | Leinster | 1457635 |
| 4 | 1871 | Leinster | 1339451 |
| 5 | 1881 | Leinster | 1278989 |
| 6 | 1891 | Leinster | 1187760 |

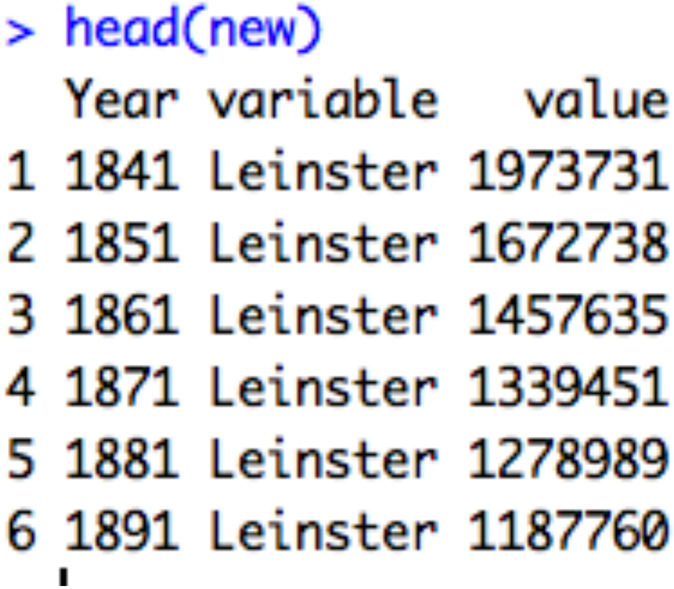
# melt() function – reshape library

melt(data, id.vars, measure.vars)

- data   Data set to melt
- id.vars    Id variables. If blank, will use all non measure.vars variables. Can be integer (variable position) or string (variable name)
- measure.vars  Measured variables. If blank, will use all non id.vars variables. Can be integer (variable position) or string (variable name)



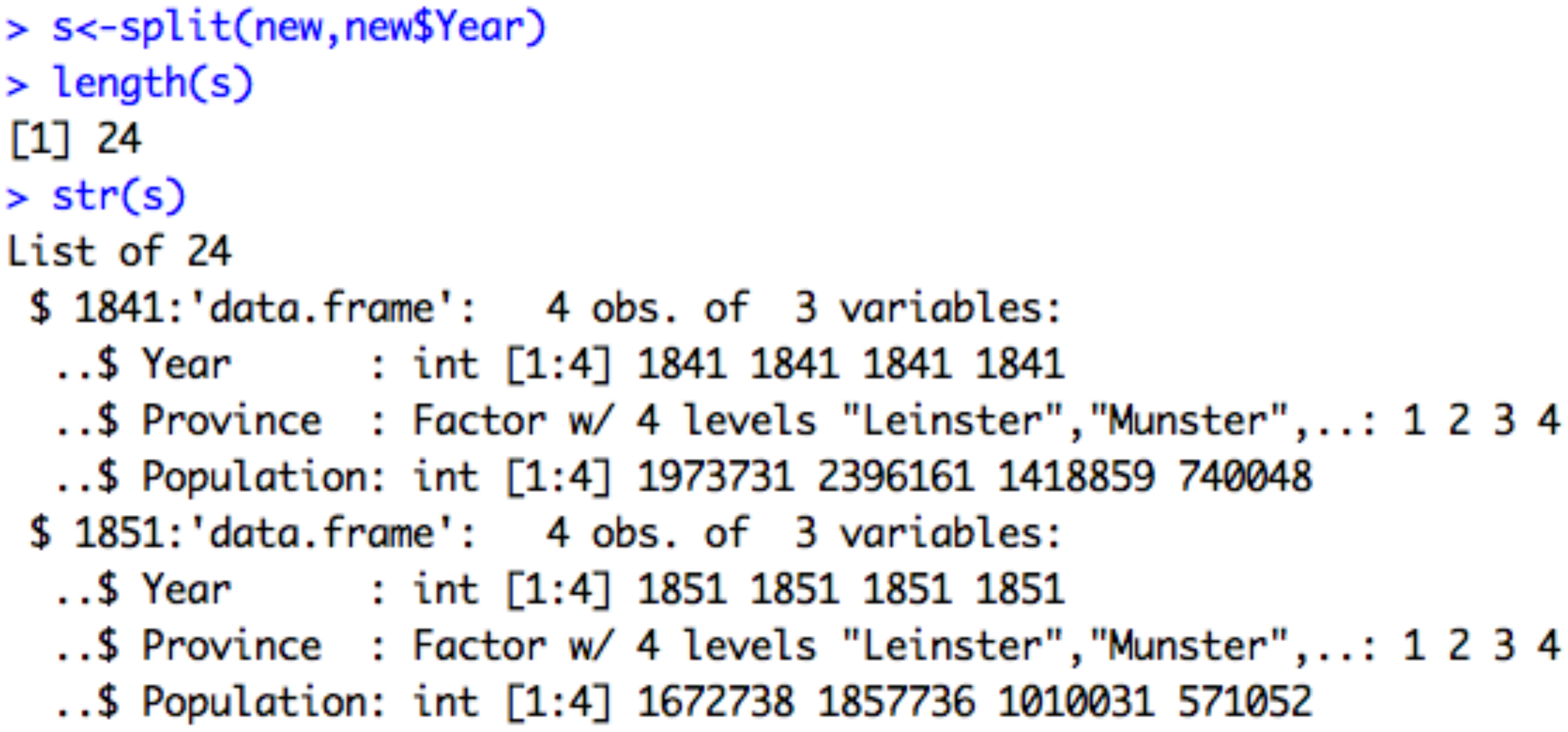
```
new<-melt(c,id.vars="Year",
            measure.vars=c("Leinster",
                           "Munster",
                           "Connacht",
                           "Ulster"))

names(new)<-c("Year","Province","Population")
```



```
> head(new)
  Year variable   value
1 1841 Leinster 1973731
2 1851 Leinster 1672738
3 1861 Leinster 1457635
4 1871 Leinster 1339451
5 1881 Leinster 1278989
6 1891 Leinster 1187760
```

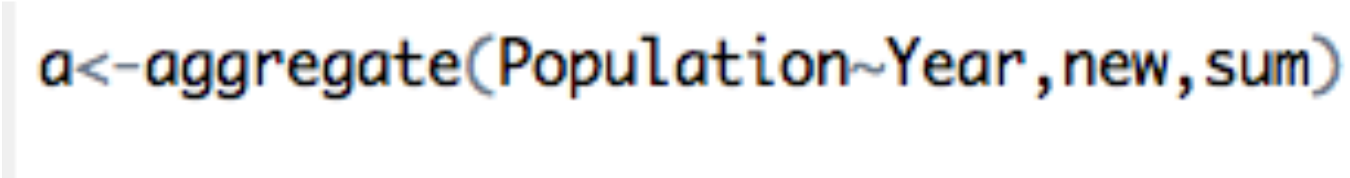
# split()

- **split** divides the data in the vector x into the groups defined by f. The replacement forms replace values corresponding to such a division. unsplit reverses the effect of split.

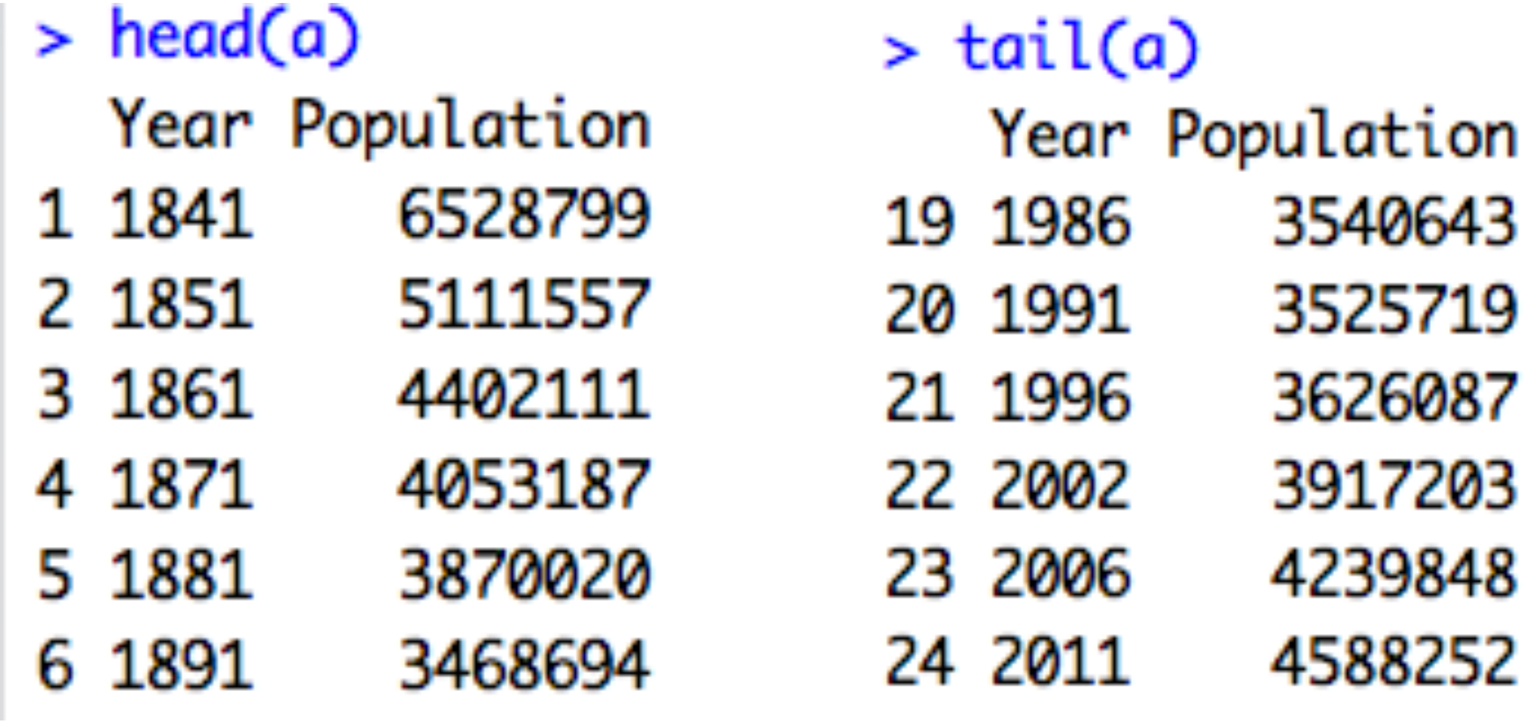


```
> s<-split(new,new$Year)
> length(s)
[1] 24
> str(s)
List of 24
 $ 1841:'data.frame':	4 obs. of  3 variables:
  ..$ Year      : int [1:4] 1841 1841 1841 1841
  ..$ Province  : Factor w/ 4 levels "Leinster","Munster",..: 1 2 3 4
  ..$ Population: int [1:4] 1973731 2396161 1418859 740048
 $ 1851:'data.frame':	4 obs. of  3 variables:
  ..$ Year      : int [1:4] 1851 1851 1851 1851
  ..$ Province  : Factor w/ 4 levels "Leinster","Munster",..: 1 2 3 4
  ..$ Population: int [1:4] 1672738 1857736 1010031 571052
```

# aggregate()



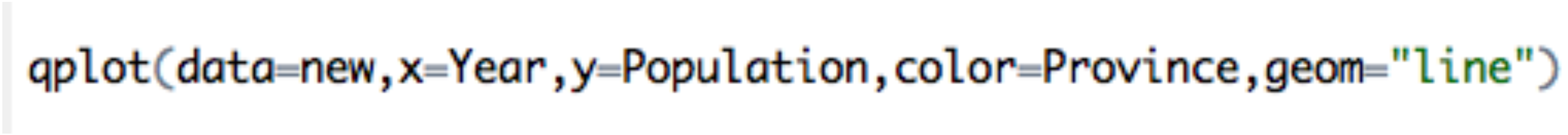
```
a<-aggregate(Population~Year,new,sum)
```



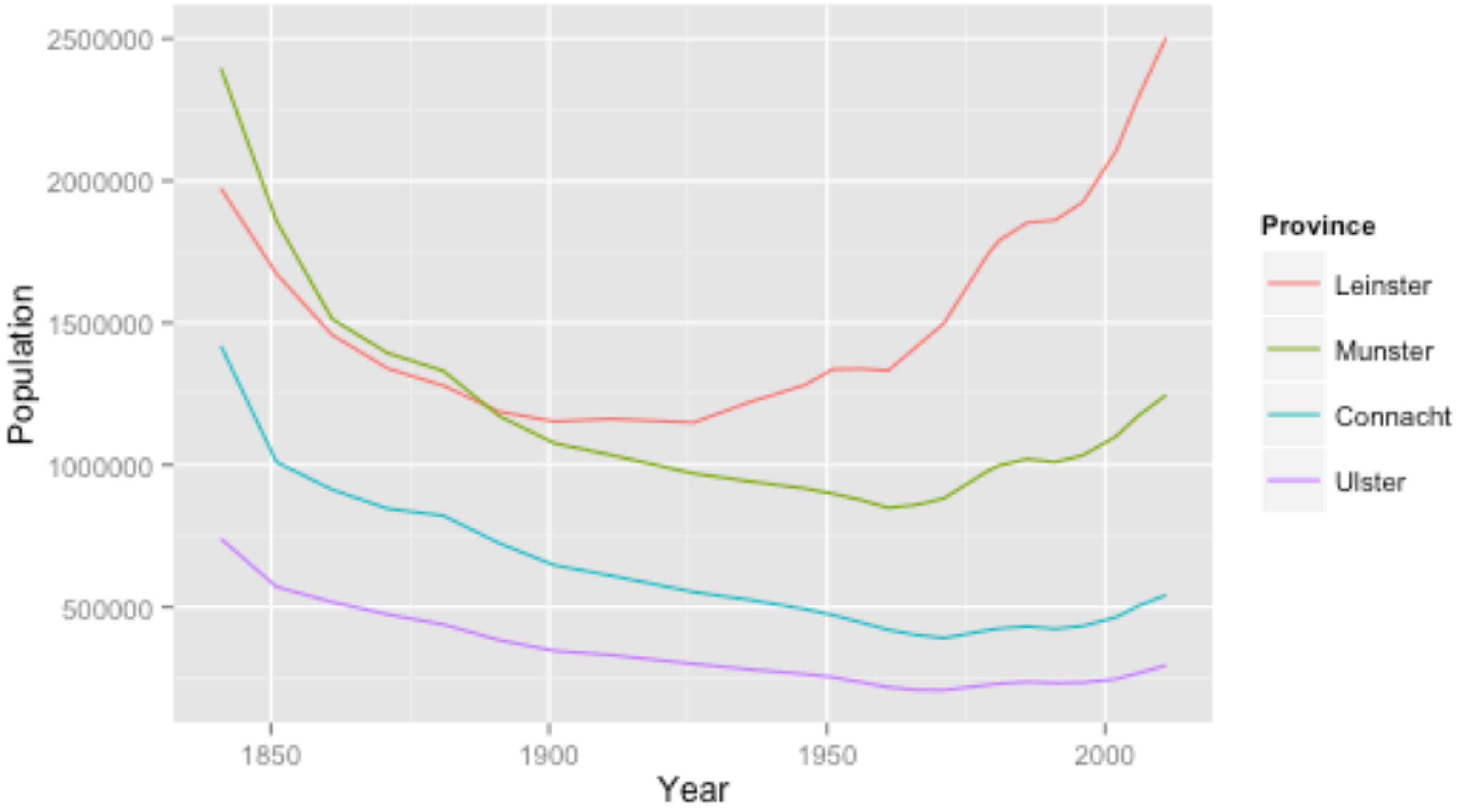
```
> head(a)
  Year Population
1 1841    6528799
2 1851    5111557
3 1861    4402111
4 1871    4053187
5 1881    3870020
6 1891    3468694
```

```
> tail(a)
   Year Population
19 1986    3540643
20 1991    3525719
21 1996    3626087
22 2002    3917203
23 2006    4239848
24 2011    4588252
```

# qplot()



```
qplot(data=new,x=Year,y=Population,color=Province,geom="line")
```


Population
Year
Province
Leinster
Munster
Connacht
Ulster