R Lesson 4

Using the matrix Function

- Creates a matrix from a single vector
- General form

  matrix(*data*, *nrow*=n, *ncol*=n, *byrow*=FALSE)

  – *data:* a data vector to be converted
  – *nrow:* specify desired number of rows
  – *ncol:* specify desired number of columns
  – *byrow:* if FALSE matrix filled by columns, otherwise by rows ← Default is FALSE

Using the matrix Function

- Example:

```
mat1 <- matrix(1:12, nrow=4, byrow=TRUE)
```

Accessing Multiple Dimensions

cpidf[??]

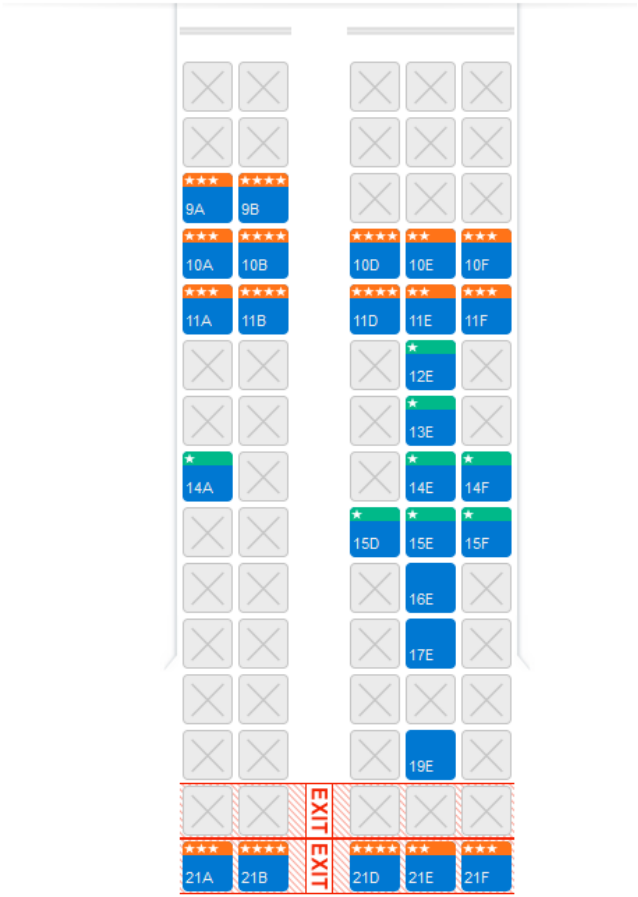
```
        Country CPI
1   New Zealand 9.5
2       Denmark 9.4
3       Finland 9.4
4        Sweden 9.3
5     Singapore 9.2
6        Norway 9.0
7   Netherlands 8.9
8     Australia 8.8
9   Switzerland 8.8
10       Canada 8.7
11   Luxembourg 8.5
12    Hong Kong 8.4
13      Iceland 8.3
14      Germany 8.0
15        Japan 8.0
16      Austria 7.8
```

cpicb[??]

```
      countries     CPI
[1,] "New Zealand" "9.5"
[2,] "Denmark"     "9.4"
[3,] "Finland"     "9.4"
[4,] "Sweden"      "9.3"
[5,] "Singapore"   "9.2"
[6,] "Norway"      "9"
[7,] "Netherlands" "8.9"
[8,] "Australia"   "8.8"
[9,] "Switzerland" "8.8"
[10,] "Canada"     "8.7"
[11,] "Luxembourg" "8.5"
[12,] "Hong Kong"  "8.4"
[13,] "Iceland"    "8.3"
[14,] "Germany"    "8"
[15,] "Japan"      "8"
[16,] "Austria"    "7.8"
```

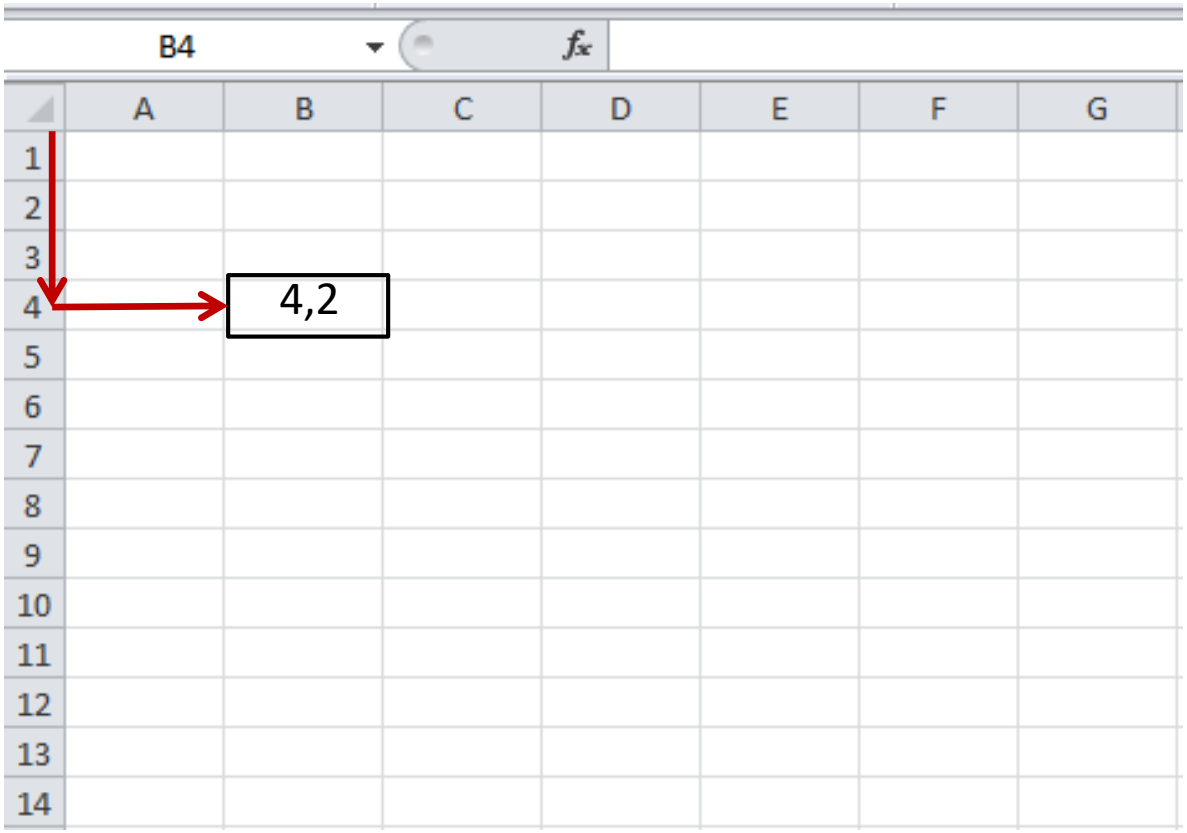
Accessing Multiple Dimensions

Flight #2292 - McDonnell Douglas Super MD-80



Accessing Multiple Dimensions

R is backwards compared to Excel



Accessing Multiple Dimensions

cpidf[row,col]

```
        Country CPI
1   New Zealand 9.5
2       Denmark 9.4
3       Finland 9.4
4        Sweden 9.3
5     Singapore 9.2
6        Norway 9.0
7   Netherlands 8.9
8     Australia 8.8
9   Switzerland 8.8
10       Canada 8.7
11   Luxembourg 8.5
12    Hong Kong 8.4
13      Iceland 8.3
14      Germany 8.0
15        Japan 8.0
16      Austria 7.8
```

cpicb[row,col]

```
      countries     CPI
[1,] "New Zealand" "9.5"
[2,] "Denmark"     "9.4"
[3,] "Finland"     "9.4"
[4,] "Sweden"      "9.3"
[5,] "Singapore"   "9.2"
[6,] "Norway"      "9"
[7,] "Netherlands" "8.9"
[8,] "Australia"   "8.8"
[9,] "Switzerland" "8.8"
[10,] "Canada"     "8.7"
[11,] "Luxembourg" "8.5"
[12,] "Hong Kong"  "8.4"
[13,] "Iceland"    "8.3"
[14,] "Germany"    "8"
[15,] "Japan"      "8"
[16,] "Austria"    "7.8"
```

Accessing Multiple Dimensions

- Accessing sub-elements (matrix or data frame):
  - mat1[2,3] *# element in the second row and third column*
  - mat1[1:2,2:3] *# sub-matrix of the first two rows and the second and third columns*
  - mat1[2,] *# the second row*
  - mat1[,2] *# the second column*
  - mat1[,-2] *# matrix with the second column removed*

Accessing Data Frames

- Treat a row as a vector: cpidf[row#,]
- Treat a column as a vector (all rows)
  1. FrameName[,col#]
  2. FrameName$ColName
  3. attach(FrameName)
  4. with(FrameName, ColName or function using Colname)
- Get all column names: names(FrameName)
- Get all row names: row.names(FrameName)

Additional Matrix Functions

- dim(mat1) – returns size of matrix
- rowSums(mat1) or colSums(mat1) - summarize
- rowMeans(mat1) or colMeans(mat1) - average
- apply(mat1,1,sum)
  - *not as fast as row…/col… but more robust in handling missing values (they may return NaN)*
  - *can perform other functions besides sum and means*
- NOTE: mat1 can also be indexes or expression

Applying Functions Across Data

- General form:
  apply(*array*, *rc*, *function*, …)
  - *array:* matrix or array to analyze (R coerces if needed)
    **Be sure to remove unwanted vectors
  - *rc:* specifies boundaries of application
    1=rows, 2=columns, c(1,2)=all cells
  - *function:* functions like mean, median, sqrt
  - *…:* additional arguments to the function
- sapply – apply for vectors
- lapply – apply for lists
- tapply – apply for ragged arrays

Putting it all Together

- Indexing Vectors

  myvector[membernumbers]

  cpi$Country[1:5]

  cpi$Country[c(1,5)]

- Indexing Table-like structures

  mymatrix[rows, columns]

  cpi[, 2:16]

Applying Functions Across Data

- apply(, , ,)
  - matrix or array (R tries to coerce)
  - 1=rows, 2=columns, c(1,2)=all cells
  - functions like mean, median, sqrt
  - additional arguments to the function
- apply(mymatrix, 1 , sum , na.rm=TRUE)
- cpi[, 2:11]

Substituting Expressions

- cpi[, 2:11]

Suppose we wanted to analyze only those years after 2005?

```
names(cpi)>'y2005'
```

Combining Data Frames

- Merge two data frames by common columns or row names, or do other versions of database *join* operations.
- General form:

merge(*frame1*, *frame2*, *all* = FALSE)

- *frame*n – data frames being joined
- *all*
  - FALSE – Only rows with matching data from data frames are included
  - TRUE – nonmatching rows added to output

Combining Data Frames

Country	CPI
New Zealand	9.5
Denmark	9.4
Finland	9.4
Sweden	9.3

Country	Capital
Austria	Vienna
Denmark	Copenhagen
Finland	Helsinki
Iceland	Reykjavik

Example:

merge(cpis, capitals, all=TRUE)