

Matrices and Data Frames

	age	sex	size	count	census	

Matrices and Data Frames

Matrix

	[,1]	[,2]	[,3]	[,4]
[1,]	60	9	35	63
[2,]	45	75	3	40
[3,]	82	64	14	15
[4,]	12	7	52	72
[5,]	4	81	18	91
[6,]	95	59	100	74
[7,]	31	79	27	8
[8,]	46	30	39	80
[9,]	89	76	38	78
[10,]	67	32	51	25

Data Frame

	state	sex	diag	death
1	NSW	M	10905	11081
2	NSW	M	11029	11096
3	NSW	M	9551	9983
4	NSW	M	9577	9654
5	NSW	M	10015	10290
6	NSW	M	9971	10344
7	NSW	M	10746	11135
8	NSW	M	10042	11069
9	NSW	M	10464	10956

Matrices and Data Frames

Matrix

	[,1]	[,2]	[,3]	[,4]
[1,]	60	9	35	63
[2,]	45	75	3	40
[3,]	82	64	14	15
[4,]	12	7	52	72
[5,]	4	81	18	91
[6,]	95	59	100	74
[7,]	31	79	27	8
[8,]	46	30	39	80
[9,]	89	76	38	78
[10,]	67	32	51	25

Data Frame

	state	sex	diag	death
1	NSW	M	10905	11081
2	NSW	M	11029	11096
3	NSW	M	9551	9983
4	NSW	M	9577	9654
5	NSW	M	10015	10290
6	NSW	M	9971	10344
7	NSW	M	10746	11135
8	NSW	M	10042	11069
9	NSW	M	10464	10956

Used as your primary data object. Essentially a spreadsheet.

Matrices and Data Frames

Matrix

	[,1]	[,2]	[,3]	[,4]
[1,]	60	9	35	63
[2,]	45	75	3	40
[3,]	82	64	14	15
[4,]	12	7	52	72
[5,]	4	81	18	91
[6,]	95	59	100	74
[7,]	31	79	27	8
[8,]	46	30	39	80
[9,]	89	76	38	78
[10,]	67	32	51	25

Used frequently in mathematical applications, models

More computationally efficient

Data Frame

	state	sex	diag	death
1	NSW	M	10905	11081
2	NSW	M	11029	11096
3	NSW	M	9551	9983
4	NSW	M	9577	9654
5	NSW	M	10015	10290
6	NSW	M	9971	10344
7	NSW	M	10746	11135
8	NSW	M	10042	11069
9	NSW	M	10464	10956


Used as your primary data object. Essentially a spreadsheet.

Matrices

To create a matrix():

- Requires data
- Number of rows or columns

Must be a multiple of your data length



```
> mat1 <- matrix(1:25, nrow = 5)
> mat1
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
```

Matrices

To create a matrix():

- Requires data
- Number of rows or columns

Want to put values by rows instead of columns?

- `byrow = TRUE`

Must be a multiple of your data length



```
> mat1 <- matrix(1:25, nrow = 5)
> mat1
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
```

```
> mat1 <- matrix(1:25, nrow = 5, byrow = TRUE)
> mat1
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
[5,]   21   22   23   24   25
```

Matrices

Labeling your rows and columns

`colnames()`

```
> colnames(mat1) <- c("this", "is", "a", "5x5", "matrix")
> mat1
      this is  a 5x5 matrix
[1,]  1  6 11 16  21
[2,]  2  7 12 17  22
[3,]  3  8 13 18  23
[4,]  4  9 14 19  24
[5,]  5 10 15 20  25
```

Matrices

Labeling your rows and columns

`colnames()`

```
> colnames(mat1) <- c("this", "is", "a", "5x5", "matrix")
> mat1
      this is  a 5x5 matrix
[1,]  1  6 11 16  21
[2,]  2  7 12 17  22
[3,]  3  8 13 18  23
[4,]  4  9 14 19  24
[5,]  5 10 15 20  25
```

`rownames()`

```
> rownames(mat1) <- c("very", "very", "very", "interesting", "names")
> mat1
      this is  a 5x5 matrix
very  1  6 11 16  21
very  2  7 12 17  22
very  3  8 13 18  23
interesting  4  9 14 19  24
names  5 10 15 20  25
```


Matrices

Piece together a matrix or add to one:

- `rbind()` and `cbind()`

```
> x <- sample(1:100, 5)
> y <- sample(1:100, 5)
> z <- sample(1:100, 5)
> x
[1] 55 86 54 10 37
> y
[1] 67 75 14 13 29
> z
[1] 44 31 97 84 99
> mat1 <- cbind(x,y,z)
> mat1
      x  y  z
[1,] 55 67 44
[2,] 86 75 31
[3,] 54 14 97
[4,] 10 13 84
[5,] 37 29 99
```

Matrices

Piece together a matrix or add to one:

- `rbind()` and `cbind()`

```
> new_vec <- sample(1:100, 3)
> new_vec
[1] 89 48 33
> mat1 <- rbind(mat1, new_vec)
> mat1
```

	x	y	z
	55	67	44
	86	75	31
	54	14	97
	10	13	84
	37	29	99
new_vec	89	48	33

```
> x <- sample(1:100, 5)
> y <- sample(1:100, 5)
> z <- sample(1:100, 5)
> x
[1] 55 86 54 10 37
> y
[1] 67 75 14 13 29
> z
[1] 44 31 97 84 99
> mat1 <- cbind(x,y,z)
> mat1
```

	x	y	z
[1,]	55	67	44
[2,]	86	75	31
[3,]	54	14	97
[4,]	10	13	84
[5,]	37	29	99

Data Frames

Like a matrix, but can have any class of data in a given column

- Because each column is essentially a vector, the class of data must be consistent in each column

	Site	plot	Posicion	Especie	Census	a	b
1	PLR	1	10	PITTTR	5	0.00600	1.09100
2	PLR	1	11	VOCHFE	2	0.00602	0.11924
3	PLR	1	12	TABIRO	2	0.00640	-0.25360
4	PLR	1	13	VIROSU	4	0.00630	-0.42860
5	PLR	1	14	PROTTE	5	0.00570	-1.76940
6	PLR	1	15	PROTTE	5	0.00570	-1.76940

Data Frames

Create a `data.frame()`

- Provide objects to turn into columns

```
> data.frame(height = sample(150:200,5),  
+           weight = sample(110:250, 5),  
+           response = c('yes', 'yes', 'no', 'yes', 'no')  
+           )  
  height weight response  
1    188    200      yes  
2    168    173      yes  
3    182    232       no  
4    191    175      yes  
5    200    246       no
```

Data Frames

Most functions that work with matrices work with data frames

- rownames, colnames, rbind, cbind etc...

Use `dim()` to get dimensions, and `str()` to summarize your data frame

```
> dat
  height weight response
1    157   126     yes
2    171   161     yes
3    150   249     no
4    178   131     yes
5    197   181     no
```

```
> dim(dat)
[1] 5 3
> str(dat)
'data.frame':   5 obs. of  3 variables:
 $ height  : int  157 171 150 178 197
 $ weight  : int  126 161 249 131 181
 $ response: Factor w/ 2 levels "no","yes": 2 2 1 2 1
```

Data Frames

Most functions that work with matrices work with data frames

- rownames, colnames, rbind, cbind etc...

Use `dim()` to get dimensions, and `str()` to summarize your data frame

Row, Column

```
> dat
  height weight response
1    157   126     yes
2    171   161     yes
3    150   249     no
4    178   131     yes
5    197   181     no
```

```
> dim(dat)
[1] 5 3
> str(dat)
'data.frame':   5 obs. of  3 variables:
 $ height  : int  157 171 150 178 197
 $ weight  : int  126 161 249 131 181
 $ response: Factor w/ 2 levels "no", "yes": 2 2 1 2 1
```

Data Frames

Most functions that work with matrices work with data frames

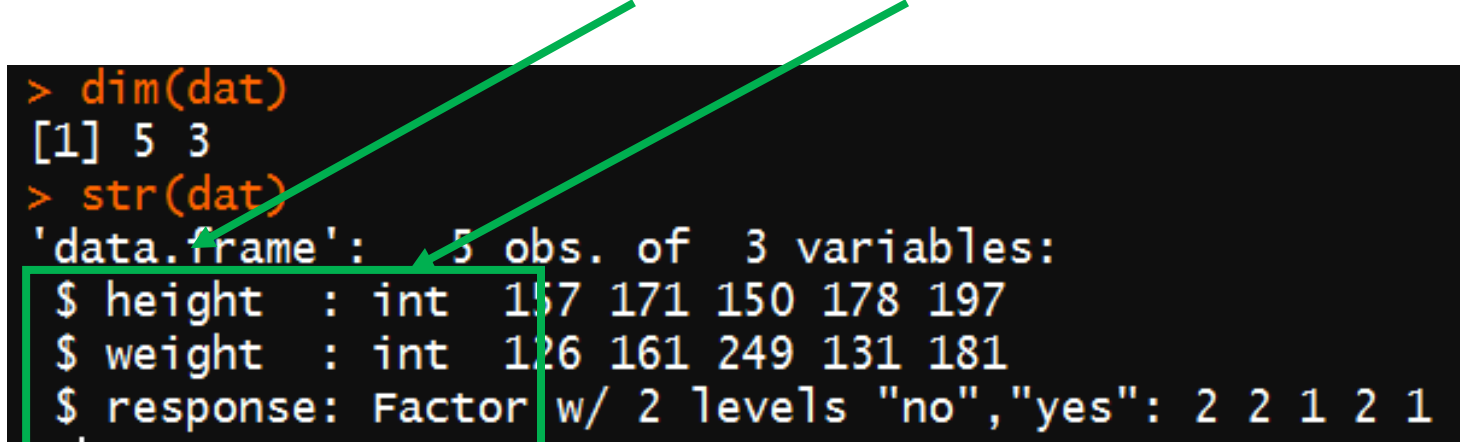
- rownames, colnames, rbind, cbind etc...

Use `dim()` to get dimensions, and `str()` to summarize your data frame

Column name : data class

```
> dat
  height weight response
1    157   126     yes
2    171   161     yes
3    150   249     no
4    178   131     yes
5    197   181     no
```

```
> dim(dat)
[1] 5 3
> str(dat)
'data.frame':   5 obs. of  3 variables:
 $ height  : int  157 171 150 178 197
 $ weight  : int  126 161 249 131 181
 $ response: Factor w/ 2 levels "no", "yes": 2 2 1 2 1
```



Subsetting 2-Dimensional Objects

	,1	,2	,3	,4	,5
	\$age	\$sex	\$size	\$count	\$census
1,					
2,					
3,					
4,					

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

object[row,col]

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

object[row,col]

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

object[row,col]



Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

object[row,col]

single value, vector, or nothing

Indices

```
> mat1[2,2]  
[1] 95
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

```
> mat1[2,2]  
[1] 95
```

```
> mat1[2:4,2]  
[1] 95 79 71
```

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

```
> mat1[2,2]  
[1] 95
```

```
> mat1[2:4,2]  
[1] 95 79 71
```

```
> mat1[2:4,2:4]  
      [,1] [,2] [,3]  
[1,]  95  76  17  
[2,]  79  93  38  
[3,]  71  81  94
```

Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

```
> mat1[2,2]  
[1] 95
```

```
> mat1[2:4,2]  
[1] 95 79 71
```

```
> mat1[2:4,2:4]  
      [,1] [,2] [,3]  
[1,]  95  76  17  
[2,]  79  93  38  
[3,]  71  81  94
```

```
> mat1[,2]  
[1] 58 95 79 71 50
```


Indices

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	84	58	39	70	65
[2,]	40	95	76	17	45
[3,]	63	79	93	38	21
[4,]	64	71	81	94	90
[5,]	25	50	20	41	82

```
> mat1[2,2]  
[1] 95
```

```
> mat1[2:4,2]  
[1] 95 79 71
```

```
> mat1[2:4,2:4]  
      [,1] [,2] [,3]  
[1,]  95  76  17  
[2,]  79  93  38  
[3,]  71  81  94
```

```
> mat1[,2]  
[1] 58 95 79 71 50
```

```
> mat1[3, ]  
[1] 63 79 93 38 21
```

\$ Operator

Data frames can be subset the same way as matrices

We also have a special way to subset data frames

- The \$ operator

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
5    174    236       no
```

\$ Operator

Data frames can be subset the same way as matrices

We also have a special way to subset data frames

- The \$ operator

```
> dat
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162      no
4    186    185     yes
5    174    236      no
```

```
> str(dat)
'data.frame':  5 obs. of  3 variables:
 $ height : int  185 184 181 186 174
 $ weight : int  225 159 162 185 236
 $ response: Factor w/ 2 levels "no", "yes": 2 2 1 2 1
```

\$ Operator

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
5    174    236       no
```

```
> dat$height
[1] 185 184 181 186 174
```

\$ Operator

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
5    174    236       no
```

```
> dat$height
[1] 185 184 181 186 174
```

```
> dat$weight
[1] 225 159 162 185 236
```

\$ Operator

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162      no
4    186    185      yes
5    174    236      no
```

```
> dat$height
[1] 185 184 181 186 174
```

```
> dat$weight
[1] 225 159 162 185 236
```

```
> dat$response
[1] yes yes no  yes no
Levels: no yes
```

Logic

\$ operator is useful for logical subsets

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
5    174    236       no
```

Logic

\$ operator is useful for logical subsets

```
> dat
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
5    174    236       no
```

```
> dat[dat$height > 180,]
  height weight response
1    185    225      yes
2    184    159      yes
3    181    162       no
4    186    185      yes
```


Logic

\$ operator is useful for logical subsets

```
> dat
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
5    174    236     no
```

```
> dat[dat$height > 180,]
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
```

```
> dat[dat$response == "yes",]
  height weight response
1    185    225     yes
2    184    159     yes
4    186    185     yes
```

Logic

\$ operator is useful for logical subsets

```
> dat
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
5    174    236     no
```

```
> dat[dat$height > 180,]
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
```

```
> dat[dat$response == "yes",]
  height weight response
1    185    225     yes
2    184    159     yes
4    186    185     yes
```

```
> dat$weight[dat$response == "no"]
[1] 162 236
```

Logic

\$ operator is useful for logical subsets

```
> dat
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
5    174    236     no
```

```
> dat[dat$height > 180,]
  height weight response
1    185    225     yes
2    184    159     yes
3    181    162     no
4    186    185     yes
```

```
> dat[dat$response == "yes",]
  height weight response
1    185    225     yes
2    184    159     yes
4    186    185     yes
```

```
> dat$weight[dat$response == "no"]
[1] 162 236
```

Applications?

Comparison of height between yes and no responses

- T-Test

	height	weight	response
1	197	162	yes
2	175	235	yes
3	185	203	yes
4	194	166	yes
5	197	235	no
6	158	113	yes
7	181	144	yes
8	169	153	no
9	188	241	yes
10	177	205	yes
11	184	123	no
12	176	144	no
13	170	133	no
14	160	141	yes
15	161	183	yes

Logic

Comparison of height between yes and no responses

- T-Test

	height	weight	response
1	197	162	yes
2	175	235	yes
3	185	203	yes
4	194	166	yes
5	197	235	no
6	158	113	yes
7	181	144	yes
8	169	153	no
9	188	241	yes
10	177	205	yes
11	184	123	no
12	176	144	no
13	170	133	no
14	160	141	yes
15	161	183	yes

```
t.test(x = dat$height[dat$response == "yes"],  
       y = dat$height[dat$response == "no"])
```

Logic

Comparison of height between yes and no responses

- T-Test

	height	weight	response
1	197	162	yes
2	175	235	yes
3	185	203	yes
4	194	166	yes
5	197	235	no
6	158	113	yes
7	181	144	yes
8	169	153	no
9	188	241	yes
10	177	205	yes
11	184	123	no
12	176	144	no
13	170	133	no
14	160	141	yes
15	161	183	yes

```
t.test(x = dat$height[dat$response == "yes"],  
       y = dat$height[dat$response == "no"])
```

Welch Two Sample t-test

```
data: dat$height[dat$response == "yes"] and dat$height[dat$response == "no"]  
t = 0.58311, df = 25.845, p-value = 0.5649
```