

Reshaping the Data

Reshaping the Data

- Reshaping involves transposing, stacking the column(s) within itself
- Reshaping is necessary for forming the analysis data which is required for statistical model building

ID	Metric1	Metric2	Metric3
1	10	5	6
2	20	4	6
3	30	3	6



ID	variable	value
1	Metric1	10
2	Metric1	20
3	Metric1	30
1	Metric2	5
2	Metric2	4
3	Metric2	3
1	Metric3	6
2	Metric3	6
3	Metric3	6

Reshaping Functions

- `t()`
- reshape2 package: `melt()`

t()

- Interchanging the data from rows to columns or columns to rows is called transposing the data
- Function t() transposes the data in matrix or data frame
- Whether input type of t() is matrix or data frame, return type is always a matrix object

```
> d <- c(12,23,14,45)
> e <- c(34,24,90,13)
> y <- data.frame(d,e)
> y
```

	d	e
1	12	34
2	23	24
3	14	90
4	45	13

```
> tr <- t(y)
> tr
```

	[,1]	[,2]	[,3]	[,4]
d	12	23	14	45
e	34	24	90	13

```
> items <- read.csv("items.csv")
>
> trnsp <- t(items)
> trnsp
```

	[,1]	[,2]
Item.ID	"121 001"	"121 002"
Item.Name	"Parker Quink Roller Ball Pen Refill, Blue"	"Pilot V7 Liquid Ink"
Item.Type	"Pen"	"Pen"
Brand	"Parker"	"Pilot"
Price	" 69"	"135"
UOM	"Piece"	"Pack"

melt()

- There is a need many times to stack a set of columns in order to make it “analysis friendly”
- This can be done using melt() function

*Syntax : melt(data, id.vars, variable.name=“variable”,
value.name=“value”,...)*

Where

data	: data frame object
id.vars	: vector of id variables
variable.name	: name of variable used to store measured variable names
value.name	: name of variable used to store values

Example of melt()

quality

	Id	A	B	C
1	1	97	93	99
2	2	73	14	94
3	3	93	93	87
4	4	100	55	66
5	5	23	77	59

```
melted1 <- melt(quality, id=c("Id"))
```

	Id	variable	value
1	1	A	97
2	2	A	73
3	3	A	93
4	4	A	100
5	5	A	23
6	1	B	93
7	2	B	14
8	3	B	93
9	4	B	55
10	5	B	77
11	1	C	99
12	2	C	94
13	3	C	87
14	4	C	66
15	5	C	59

Example of melt()

quality

	Id	A	B	C
1	1	97	93	99
2	2	73	14	94
3	3	93	93	87
4	4	100	55	66
5	5	23	77	59

```
melted2 <- melt(quality, id=c("Id"), variable.name = "Policy",
  value.name = "Score")
```

	Id	Policy	Score
1	1	A	97
2	2	A	73
3	3	A	93
4	4	A	100
5	5	A	23
6	1	B	93
7	2	B	14
8	3	B	93
9	4	B	55
10	5	B	77
11	1	C	99
12	2	C	94
13	3	C	87
14	4	C	66
15	5	C	59

melt() with more than one IDs

	Id	Group	Policy.A	Policy.B	Policy.C
1	1	A	97	93	99
2	2	A	73	14	94
3	3	B	93	93	87
4	4	B	100	NA	66
5	5	B	23	77	59

```
melted3 <- melt(quality2, id=c("Id","Group"),variable.name = "Policy",
  value.name = "Score")
```

	Id	Group	Policy	Score
1	1	A	Policy.A	97
2	2	A	Policy.A	73
3	3	B	Policy.A	93
4	4	B	Policy.A	100
5	5	B	Policy.A	23
6	1	A	Policy.B	93
7	2	A	Policy.B	14
8	3	B	Policy.B	93
9	4	B	Policy.B	NA
10	5	B	Policy.B	77
11	1	A	Policy.C	99
12	2	A	Policy.C	94
13	3	B	Policy.C	87
14	4	B	Policy.C	66
15	5	B	Policy.C	59

Combining the Data

Combining the Data

- We can combine the data horizontally and vertically
- Horizontally we can combine with `cbind()` and `merge()`
- Vertically we can combine with `rbind()`

rbind() and cbind()

- Matrices can also be created by *using* functions ***cbind()*** and ***rbind()***.
- ***rbind()*** binds that rows and ***cbind()*** binds the columns

```
> a <- 1 : 5
> a
[1] 1 2 3 4 5
> b <- 46 : 50
> b
[1] 46 47 48 49 50
> cb <- cbind(a,b)
> cb
      a  b
[1,] 1 46
[2,] 2 47
[3,] 3 48
[4,] 4 49
[5,] 5 50
```

```
> rb <- rbind(a,b)
> rb
      [,1] [,2] [,3] [,4] [,5]
a        1    2    3    4    5
b       46   47   48   49   50
```

merge()

- The combining with a matched key (set of columns) of the datasets can be done with the help of merge function

Syntax : merge(X , Y , by.x = c(),by.y = c(), all.x, all.y)

where X , Y: data frames to merge

by.x , by.y : specifies the key from two datasets by which

to merge. If the column names are same then

we can just specify them in by= option

all.x : logical. If all.x is TRUE then extra rows will be added to the output from data X

all.y : logical. If all.y is TRUE then extra rows will be added to the output from data Y

Example of merge()

```
ord <- merge(orders, ord_Details, by = "Order.ID")
```

Orders

	Order.ID	Order.Date	Place.of.Shipment	Payment.Terms
1	32 90 001	31-Dec-10	Pune	Cheque
2	32 90 002	6-Jan-11	Nasik	Online
3	32 90 003	14-Jan-11	Ahmednagar	Cash
4	32 90 004	18-Feb-11	Nanded	Cheque
5	32 90 005	19-Feb-11	Kolhapur	Cash
6	32 90 006	1-Mar-11	Buldhana	Online
7	32 90 007	4-Mar-11	Nasik	Online
8	32 90 008	15-Mar-11	Pune	Online

Ord_Details

	Order.ID	Item.ID	Qty
1	32 90 001	121 021	7
2	32 90 001	121 003	49
3	32 90 001	121 023	1
4	32 90 001	121 018	9
5	32 90 001	121 015	29

ord

	Order.ID	Order.Date	Place.of.Shipment	Payment.Terms	Item.ID	Qty
1	32 90 001	31-Dec-10	Pune	Cheque	121 021	7
2	32 90 001	31-Dec-10	Pune	Cheque	121 003	49
3	32 90 001	31-Dec-10	Pune	Cheque	121 023	1
4	32 90 001	31-Dec-10	Pune	Cheque	121 018	9
5	32 90 001	31-Dec-10	Pune	Cheque	121 015	29
6	32 90 001	31-Dec-10	Pune	Cheque	121 014	44
7	32 90 001	31-Dec-10	Pune	Cheque	121 001	15

Merging with more than one field

Lab_Uric

	PatientID	Lab_test_Date	Uric_Acid
1	A01	2014-06-02	3.5
2	A01	2014-07-14	3.9
3	A02	2014-05-03	4.5
4	A02	2014-08-12	5.6
5	A03	2015-02-18	6.4
6	A03	2015-02-27	6.5
7	A04	2015-01-23	5.6
8	A04	2015-02-19	6.5

Lab_Keto17

	PID	Lab_test_Date	KETO17
1	A01	2014-06-02	8.7
2	A01	2014-07-14	9.5
3	A02	2014-05-03	10.4
4	A02	2014-08-12	20.9
5	A03	2015-02-18	10.3
6	A03	2015-02-27	10.8
7	A04	2015-01-23	17.8
8	A04	2015-02-19	15.2

adlb

	PatientID	Lab_test_Date	Uric_Acid	KETO17
1	A01	2014-06-02	3.5	8.7
2	A01	2014-07-14	3.9	9.5
3	A02	2014-05-03	4.5	10.4
4	A02	2014-08-12	5.6	20.9
5	A03	2015-02-18	6.4	10.3
6	A03	2015-02-27	6.5	10.8
7	A04	2015-01-23	5.6	17.8
8	A04	2015-02-19	6.5	15.2

```
adlb <- merge(Lab_Uric, Lab_Keto17, by.x = c("PatientID", "Lab_test_Date"),
              by.y = c("PID", "Lab_test_Date"))
```