L09 Data Management II

EPID 799B

Fall 2016

Xiaojuan Li

Outline

Working on the whole dataset

- Working with columns
- Sorting data frames
- Subsetting data frames
- Merging data frames
- Joining data frames
- Transposing data frames

Announcements

- Homework 2 due Monday 9/26

Adding a column

```
data$newcol <-  5
# If you assign a single value to the new column, the entire column will
# be filled with that value.

data$newcol <- vec
# If the length of the vector needs to match the number of rows in the
# data frame.
```

Renaming columns

```
# Option 1 - Rename by numeric position:
        names(data)[1] <- "newname"

# Option 2 - Rename by name:
        names(data)[names(data)=="col1"] <- "newname"
```

Deleting a column

```
# Option 1 - assign NULL to that column
        data$col1 <- NULL

# Option 2 - use the subset() function
        data <- subset(data, select= -col1)

# Delete multiple columns
        data <- subset(data, select=c(-col1, -col2))
```

Rearranging columns

```
# Option 1 - reorder by numeric position:
        data <- data[c(1,3,2)]          # list-style indexing
        data <- data[, c(1,3,2)]        # matrix-style indexing

# Option 2 - reorder by name
        data <- data[c("col1", "col3", "col2")]
```

Sorting a data frame

# Sort based on 1 variable, the default order is ascending

```
newdata1<- data[order(data$col1), ]
```

# Sort based on 2 variables, use –variable to request descending order

```
newdata1 <- data[order(data$col1, -data$col3), ]
```

Subsetting data frames

```
# By logical
        newdata <- data[ which(data$gender=="F" & data$age > 65), ]
        newdata <- subset(data, gender== "M" & age > 25,
                              select=weight:income)

# By position
        newdata <- data[c(-3,-5)]
        newdata <- data[c(1,5:10)]

# By name
        newdata <- data[1:5, c("col1", "col2", "col5"]
```

Merging data frames

# If the indexing variable has the same name in the two data frames

```
merge(data1, data2, by="ID")
```

# If the indexing variable has different names

```
merge(data1, data2, by.x="ID1", by.y=“ID2")
```

# If multiple indexing variables

```
merge(data1, data2, by=c("ID", "Country"))
```

Joining data frames

```
# Combine column-wise - the number of rows must match
# Avoid using cbind() when inputs are not of the same type.
        cbind(df1, df2)

# Combine row-wise, both the number and names of columns must
# match, but the orders do not have to match
        rbind(df1, df2)
```

Converting data frames

	name	sex	before	after
1	bob	m	150	152
2	amy	f	135	130
3	kai	m	190	180

Converting data frames – from wide to long

```
library(reshape2)

melt(data, id.vars=c(“name", “sex"), variable.name=“timepoint",
value.name=“weight")

# all non-ID columns are by default measure variables. Their names go
# into column defined by variable.name, their values go into column
# defined by value.name.
```

	name	sex	timepoint	weight
1	bob	m	before	150
2	amy	f	before	135
3	kai	m	before	190
4	bob	m	after	152
5	amy	f	after	130
6	kai	m	after	180

Converting data frames – from long to wide

```
library(reshape2)
dcast(data, name + sex ~ timepoint, value.var=“weight")
# ID variables are before ~
# variable variables are after ~. When there is more than one variable
# variable, the values are combined with an underscore.
```

	name	sex	before	after
1	amy	f	135	130
2	bob	m	150	152
3	kai	m	190	180