

Data.Frames

Gordon Erlebacher

Quick Review

- Vectors: collection of elements of the same type
 - if elements are of different types, they are coerced to the same type, or there is an error
- Lists: collection of elements of different types
- Factors: collection of categories. Can convert between vector and factor (using `factor()`)
- Matrix: data presented in two-dimensional format
 - data can be accessed by row or by column

Quick Review

- vector
 - **[...]** returns a vector of **one or more** elements
 - **[[..]]** returns a **single** element
- list
 - **[...]** returns a list of **one or more** elements
 - **[[..]]** returns a **single** element
- matrix
 - **[a,b]** returns a submatrix of **one or more rows and one or more columns**
 - **[[a,b]]** returns a **single** element of the matrix

Quick Review

- Some functions useful for vectors
 - seq, rep, sample, summary, mean, sum, rnorm, etc.
- Some functions useful for vectors, lists, matrix
 - summary, class, str

Data.Frames

- Data is often stored as a collection of columns
- Each observation is a single row
- Each column represents data characteristics
- Each column has a label
- Each column has an underlying type (numeric, character, factor)
- Each column can be accessed via its label
- Many functions operate on data.frames
 - Linear regression (lm), ANOVA (aov), summary(), plot(), ...
 - Transformation, aggregation (with, by, aggregate, tapply, etc.)

Creating a data.frame

```
df = data.frame(vector1, vector2, vector3, ...)
```

Each vector is a column

Each vector can have elements of different types
(integer, reals, strings, booleans)

Each vector is of the same length

Data.Frames

```
> weekday=c("Mon","Tue","Wed","Thur","Fri")
```

```
> cups.of.coffee <- c(2,3,1,3,2)
```

```
> weekday
```

vector

```
[1] "Mon" "Tue" "Wed" "Thur" "Fri"
```

```
> cups.of.coffee
```

vector

```
[1] 2 3 1 3 2
```

```
> d = data.frame(factor(weekday), cups.of.coffee)
```

```
> d
```

weekday cups.of.coffee

1	Mon	2
2	Tue	3
3	Wed	1
4	Thur	3
5	Fri	2

data.frame

Two columns

Each column is a vector

Each column has a **header**

Each row has a **label**

Data.Frames

```
> d = data.frame(weekday, cups.of.coffee)
```

```
> str(d)
```

```
'data.frame':  5 obs. of  2 variables:
```

```
$ weekday      : Factor w/ 5 levels "Fri","Mon","Thur",...: 2 4 5 3 1
```

```
$ cups.of.coffee: num  2 3 1 3 2
```

Column Extraction

```
> d[1]
  weekday
1    Mon
2    Tue
3    Wed
4   Thur
5    Fri
```

Vector Extraction

```
> d[[1]]
[1] Mon Tue Wed Thur Fri
Levels: Fri Mon Thur Tue Wed
```

Data Frames

```
> a = data.frame(c(1,2),c("a","b"))
> a
  c.1..2.  c..a....b..
1      1      a
2      2      b
> str(a)
'data.frame':  2 obs. of  2 variables:
 $ c.1..2.  : num  1 2
 $ c..a....b.: Factor w/ 2 levels "a","b":
 1 2
```

```
> mode(a)
[1] "list"
> mode(a[1])
[1] "list"
> mode(a[[1]])
[1] "numeric"
```

Default column names based on content

Data Frames

```
> a = data.frame(x=c(1,2),y=c("a","b"))
```

```
> a
```

Column names

```
  x y
```

```
1 1 a
```

```
2 2 b
```

```
> str(a)
```

```
'data.frame':  2 obs. of  2 variables:
```

```
$ x: num  1 2
```

```
$ y: Factor w/ 2 levels "a","b": 1 2
```

Data Frames

```
> x = c(1,2)
```

vector of integers

```
> y = c("a","b")
```

vector of strings

```
> a = data.frame(x,y)
```

variable names become header names

```
> a
```

vector of strings become factors

```
  x y
```

```
1 1 a
```

```
2 2 b
```

```
> str(a)
```

```
'data.frame':  2 obs. of  2 variables:
```

```
$ x: num  1 2
```

```
$ y: Factor w/ 2 levels "a","b": 1 2
```

List versus dataframe

- A list is heterogenous
 - each element can be a different kind of object
 - these elements might have no relation to each other. They can be a mixture of lists, vectors, matrices, dataframes, etc.
- A dataframe is a collection of columns
 - each column is a vector
 - each column has the same number of rows
 - the columns are stored in a list

datasets that are data.frames

- quakes
- attitude
- Insectspray

- How to determine the type of a dataset?
- Find another dataset of type “data.frame”

UCBAdmissions data.frame

- What are types of columns?
- What is the fraction of accepted males, accepted females. Does there appear to be any bias?
- Calculate these fractions for each department.
- Compute these results using your own function, stored in your own script.

dataset: quakes

earthquake data

?quakes

example(quakes)

class(quakes)

str(quakes)

Problem

- Determine the **average depth and magnitude** of quakes that occur over **Japan**
- Same question, but replace Japan by the U.S.
- Do both problems using your own function.

List of vectors

```
> li = list(head(quakes$lat), head(quakes$long),  
+ head(quakes$depth))
```

```
> li
```

```
[[1]]
```

```
[1] -20.42 -20.62 -26.00 -17.97 -20.42 -19.68
```

```
[[2]]
```

```
[1] 181.62 181.03 184.10 181.66 181.96 184.31
```

```
[[3]]
```

```
[1] 562 650 42 626 649 195
```

Each element of the list is a column of head(quakes)

Equivalent approach

```
> li = list(q$lat, q$long, q$depth)
```

```
> li
```

```
[[1]]
```

```
[1] -20.42 -20.62 -26.00 -17.97 -20.42 -19.68
```

```
[[2]]
```

```
[1] 181.62 181.03 184.10 181.66 181.96 184.31
```

```
[[3]]
```

```
[1] 562 650 42 626 649 195
```

Another Equivalent Approach

```
> q = head(quakes)
> li = with(q, list(lat, long, depth))
> li
[[1]]
[1] -20.42 -20.62 -26.00 -17.97 -20.42 -19.68

[[2]]
[1] 181.62 181.03 184.10 181.66 181.96 184.31

[[3]]
[1] 562 650 42 626 649 195
```

Easier to read than
`list(q$lat, q$long, q$depth)`
especially with long variable names

List of vector to data.frame

```
> q = head(quakes)
> li = with(q, list(lat, long, depth))
> dfr = as.data.frame(li)
```

```
> head(dfr,2)
  c..20.42...20.62...26...17.97...20.42...19.68.
1                                -20.42
2                                -20.62
  c.181.62..181.03..184.1..181.66..181.96..184.31.
1                                181.62
2                                181.03
  c.562L..650L..42L..626L..649L..195L.
1                                562
2                                650
>
```

Need to create column names (they disappeared when using the vector names `q$lat`, etc.)

Adding names

```
> q = head(quakes)
> li = with(q, list(lat, long, depth))
> dfr = as.data.frame(li)
> names(dfr) = c("lat", "long", "depth")
> dfr
```

	lat	long	depth
1	-20.42	181.62	562
2	-20.62	181.03	650
3	-26.00	184.10	42
4	-17.97	181.66	626
5	-20.42	181.96	649
6	-19.68	184.31	195

colnames and names do the same thing on data.frames, but not in general

Alternative

```
> li = with(q, list(latitude=lat, longitude=long, depth=depth))  
> dfr = as.data.frame(li)  
> dfr
```

	latitude	longitude	depth
1	-20.42	181.62	562
2	-20.62	181.03	650
3	-26.00	184.10	42
4	-17.97	181.66	626
5	-20.42	181.96	649
6	-19.68	184.31	195

What is a Factor

- Factor is a category, which is not meant to be numeric, and is usually not used in calculations
- Examples
 - eye color, flow names, school names
 - A dataset with 1000 students from 4 schools, would have the schools considered as factors
 - Numbers can also be factors when used as categories (their value is not important)

Numeric vs Factor

- Data is most organized in columns
 - numerical or factor
- Data can either be
 - numerical (quantitative by default) or factor
 - character (factor by default)
- Cannot do math with factors

Example

```
df = data.frame(c(1:5), c(6,10))
```

```
ef = df
```

```
ef[2] = as.factor(ef[[2]]) # note the [[...]]
```

```
df[2] + df[2] # works as expected
```

```
ef[2] + ef[2] # cannot add factors!!
```

Numerical Columns

```
> d = data.frame(c(1:10),c(13:15))
```

Error in data.frame(c(1:10), c(13:15)) :

arguments imply differing number of rows: 10, 3

```
> d = data.frame(c(1:10), c(13:22))
```

```
> s = d[1] + d[2]
```

```
> head(s)
```

c.1.10.

1	14
2	16
3	18
4	20
5	22
6	24

Same number of rows in
each argument

head: first few elements

[..] column selection

add two columns:
generate a new column

Naming Columns

```
> a = data.frame(x=c(1,2,3),y=c(10,12,20))
```

```
> str(a)
```

```
'data.frame':  3 obs. of  2 variables:
```

```
 $ x: num  1 2 3
```

```
 $ y: num 10 12 20
```

```
> names(a)
```

```
[1] "x" "y"
```

```
> names(a) = c("price", "goal")
```

```
> str(a)
```

```
'data.frame':  3 obs. of  2 variables:
```

```
 $ price: num  1 2 3
```

```
 $ goal : num 10 12 20
```

Dataframe Attributes

```
> a = data.frame(price=c(1,2,3),goal=c(10,12,20))  
> attributes(a)
```

```
$names
```

```
[1] "price" "goal"
```

```
$row.names      # See next page
```

```
[1] 1 2 3
```

```
$class
```

```
[1] "data.frame"
```

```
rownames(a)[1] = "sta 1"
```

```
rownames(a) = c("sta 1", "sta 2", "sta 3")
```

Row Names

```
a = data.frame(price=c(1,2,3),goal=c(10,12,20))
```

```
> a
```

	price	goal
1	1	10
2	2	12
3	3	20

```
> row.names(a) <- c("orange", "apple", "grapefruit")
```

```
> a
```

	price	goal
orange	1	10
apple	2	12
grapefruit	3	20

Combining Data.frame

- Consider the quakes dataset (part of R)
- `q = quakes`
`class(q)` `# data.frame`

```
> head(quakes)
   lat  long depth mag stations
1 -20.42 181.62  562 4.8       41
2 -20.62 181.03  650 4.2       15
3 -26.00 184.10   42 5.4       43
4 -17.97 181.66  626 4.1       19
5 -20.42 181.96  649 4.0       11
6 -19.68 184.31  195 4.0       12
```

Add Columns

```
> head(quakes,2)
```

	lat	long	depth	mag	stations
1	-20.42	181.62	562	4.8	41
2	-20.62	181.03	650	4.2	15

```
> head(cbind(quakes, quakes[2]))
```

	lat	long	depth	mag	stations	long
1	-20.42	181.62	562	4.8	41	181.62
2	-20.62	181.03	650	4.2	15	181.03
3	-26.00	184.10	42	5.4	43	184.10
4	-17.97	181.66	626	4.1	19	181.66
5	-20.42	181.96	649	4.0	11	181.96
6	-19.68	184.31	195	4.0	12	184.31

Add Rows

```
> q = head(quakes,5)
> q
  lat  long depth mag stations
1 -20.42 181.62  562 4.8      41
2 -20.62 181.03  650 4.2      15
3 -26.00 184.10   42 5.4      43
4 -17.97 181.66  626 4.1      19
5 -20.42 181.96  649 4.0      11
```

```
> qr = rbind(q[3,], q[-c(2,3),], q[c(1,2),]); qr
  lat  long depth mag stations
3 -26.00 184.10   42 5.4      43
1 -20.42 181.62  562 4.8      41
4 -17.97 181.66  626 4.1      19
5 -20.42 181.96  649 4.0      11
5 1 -20.42 181.62  562 4.8      41
6 -20.62 181.03  650 4.2      15
```

Row names are either taken from original dataframe or made up by R

Row names are either taken from original dataframe or made up by R

Accessing data.frame components

```
> q = head(quakes,5); q
  lat  long depth mag stations
1 -20.42 181.62  562 4.8      41
2 -20.62 181.03  650 4.2      15
3 -26.00 184.10   42 5.4      43
4 -17.97 181.66  626 4.1      19
5 -20.42 181.96  649 4.0      11
```

Many ways to access
dataframe elements!

`q[3]`, `q[[3]]`, `q["long"]`, `q[["long"]]`, `q$long`

`q[,3]`, `q[[3,4]]`, `q["3","mag"]`

`q[c(2,4),"depth"]` : subset of a dataframe

```
> q[c(2,4), "depth"]  
[1] 650 626
```

returns a vector

```
> q[c(2,4), "depth", drop=F]  
  depth  
2   650  
4   626  
>
```

returns a dataframe

```
> q[c(2,4), c("long", "depth")]  
[1] 650 626
```

returns a dataframe

```
q[3]
  depth
1  562
2  650
3   42
4  626
5  649
```

Returns a dataframe of one column

```
> q[c(2,3)]
  long depth
1 181.62  562
2 181.03  650
3 184.10   42
4 181.66  626
5 181.96  649
```

Both
methods
return
a dataframe

```
> q[, c(2,3)]
  long depth
1 181.62  562
2 181.03  650
3 184.10   42
4 181.66  626
5 181.96  649
```

```
> q[[3]]  
[1] 562 650 42 626 649
```

[[...]] always returns a vector

[[...]] always returns a single element

vector --> single element

list --> element of list (can be anything)

dataframe --> returns a vector (a single column)

**columns of dataframe are homogeneous
All elements have the same type (like a vector)**

More complex subsets

- Extract all the women giving birth in March who weigh less than 120 lbs
- List all students with a grade of less than 80 percent who studied between 1996 and 1999
- Use the command subset

?InsectSprays

Effectiveness of Insect Sprays

Description:

The counts of insects in agricultural experimental units treated with different insecticides.

Usage:

`InsectSprays`

Format:

A data frame with 72 observations on 2 variables.

[,1]	count	numeric	Insect count
[,2]	spray	factor	The type of spray

head(InsectSprays)

	count	spray
1	10	A
2	7	A
3	20	A
4	14	A
5	14	A
6	12	A

I'd like all records with a count > 14

```
> s= subset(InsectSprays, subset=count > 14)
```

```
> head(s)
```

	count	spray
3	20	A
8	23	A
9	17	A
10	20	A
14	17	B
15	21	B

Alternative:
`s[s$count > 14, ]`

Note: there are no counts greater than 14!

```
## S3 method for class 'data.frame'  
subset(x, subset, select, drop = FALSE, ...)
```

rows **columns**

```
> head( subset(InsectSprays,subset = count > 14,select=count) )  
  count  
3     20  
8     23  
9     17  
10    20  
14    17  
15    21
```

```
> head( InsectSprays[InsectSprays$count > 14, 1] )  
[1] 20 23 17 20 17 21
```

Returns a vector. In order to return a dataframe, add “drop=F” as a 3rd argument to the dataframe.

```
> head( InsectSprays[InsectSprays$count > 14, 1, drop=F] )  
  count  
3    20  
8    23  
9    17  
10   20  
14   17  
15   21
```

drop overrides strange behavior
of data.frames when 2nd argument
is a single column.

Returns a dataframe (if I use **drop=F**)

Next Lessons

- We will discuss hypothesis testing, testing for normality, equality of variance, frequency analysis using tables and plots