Introduction to R: **Data Frames**

Day 1, Part C

In this lecture

1. Composition of a data frame
2. Viewing a data frame
3. Selecting columns
4. Creating columns
5. Modifying columns
6. Removing columns
7. Selecting rows
8. Removing rows

Data frames

So far we've covered three data structures for ordered collections of values:

- Vectors (1 dimension)
- Matrices (2 dimensions)
- Arrays (N dimensions)

These three data structures require that all values be of the same class (e.g., character, numeric, integer, or logical).

Data frames

So far we've covered three data structures for ordered collections of values:

- Vectors (1 dimension)
- Matrices (2 dimensions)
- Arrays (N dimensions)

These three data structures require that all values be of the same class (e.g., character, numeric, integer, or logical).

For spreadsheet-like data, R uses data frames (`data.frame`).

Data frames are essentially collections of vectors of the same length where each vector forms a column of a table; these vectors *do not need to be the same class*.

Creating a data frame from vectors

A data frame can be constructed by combining several related vectors:

```
> iso3 <- c("CAN", "USA", "MEX")
> pop <- c(35.16, 318.9, 122.3)
> admin1 <- c(13L, 51L, 31L)
> spanish <- c(FALSE, FALSE, TRUE)
```

```
> df <- data.frame(iso3, pop, admin1, spanish)
> df
  iso3    pop admin1 spanish
1  CAN  35.16     13   FALSE
2  USA 318.90     51   FALSE
3  MEX 122.30     31    TRUE
> rm(iso3, pop, admin1, spanish)  # clean up the work space
```

Data frame structure

You can check if an object is a data frame using `is.data.frame()`:

```
> is.data.frame(df)
[1] TRUE
```

And you can see what class each column is using the `str()` function:

```
> str(df)
'data.frame':   3 obs. of  4 variables:
 $ iso3   : Factor w/ 3 levels "CAN","MEX","USA": 1 3 2
 $ pop    : num  35.2 318.9 122.3
 $ admin1 : int  13 51 31
 $ spanish: logi  FALSE FALSE TRUE
```

Viewing a data frame

Like other data structures, you can view a data frame by printing it in the console:

```
> df
  iso3    pop admin1 spanish
1  CAN  35.16     13   FALSE
2  USA 318.90     51   FALSE
3  MEX 122.30     31    TRUE
```

Viewing a data frame

For larger data frames this is often cumbersome. . .

```
> data(airquality)  # load the airquality data
> airquality
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
5      NA      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23     299  8.6   65     5   7
8      19      99 13.8   59     5   8
9       8      19 20.1   61     5   9
10     NA     194  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16     256  9.7   69     5  12
13     11     290  9.2   66     5  13
14     14     274 10.9   68     5  14
15     18      65 13.2   58     5  15
16     14     334 11.5   64     5  16
17     34     307 12.0   66     5  17
18      6      78 18.4   57     5  18
19     30     322 11.5   68     5  19
20     11      44  9.7   62     5  20
21      1       8  9.7   59     5  21
```

Viewing a data frame

In this case, the `head()` and `tail()` functions are useful:

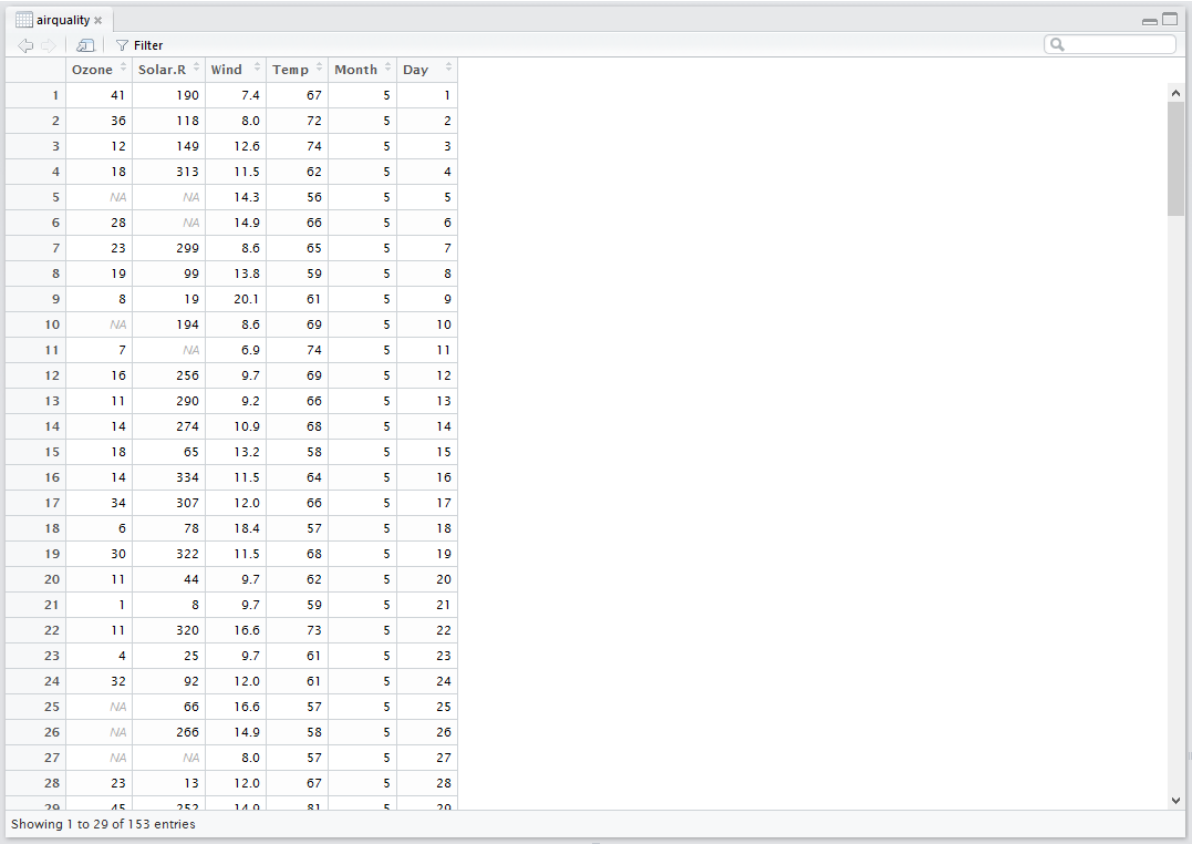
```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

```
> tail(airquality, 3)
    Ozone Solar.R Wind Temp Month Day
151    14     191 14.3   75     9  28
152    18     131  8.0   76     9  29
153    20     223 11.5   68     9  30
```

Viewing a data frame

R also has a built-in viewer for data frames:

```
> View(airquality)
```



airquality

Filter

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10
11	7	NA	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15
16	14	334	11.5	64	5	16
17	34	307	12.0	66	5	17
18	6	78	18.4	57	5	18
19	30	322	11.5	68	5	19
20	11	44	9.7	62	5	20
21	1	8	9.7	59	5	21
22	11	320	16.6	73	5	22
23	4	25	9.7	61	5	23
24	32	92	12.0	61	5	24
25	NA	66	16.6	57	5	25
26	NA	266	14.9	58	5	26
27	NA	NA	8.0	57	5	27
28	23	13	12.0	67	5	28
29	45	252	14.9	81	5	29

Showing 1 to 29 of 153 entries

Viewing a data frame

The `summary()` command is useful for summarizing a data frame's contents:

```
> summary(airquality)
     Ozone           Solar.R           Wind       
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400  
 Median : 31.50   Median :205.0   Median : 9.700  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500  
 Max.   :168.00   Max.   :334.0   Max.   :20.700  
 NA's   :37       NA's   :7                       
      Temp           Month            Day      
 Min.   :56.00   Min.   :5.000   Min.   : 1.0  
 1st Qu.:72.00   1st Qu.:6.000   1st Qu.: 8.0  
 Median :79.00   Median :7.000   Median :16.0  
 Mean   :77.88   Mean   :6.993   Mean   :15.8  
 3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :97.00   Max.   :9.000   Max.   :31.0  
```

Viewing a data frame

`nrow()`, `ncol()`, and `dim()` are useful for figuring out a data frame's size:

```
> nrow(airquality)
[1] 153
> ncol(airquality)
[1] 6
> dim(airquality)
[1] 153   6
```

Viewing a data frame

`nrow()`, `ncol()`, and `dim()` are useful for figuring out a data frame's size:

```
> nrow(airquality)
[1] 153
> ncol(airquality)
[1] 6
> dim(airquality)
[1] 153   6
```

and you can get a list of columns using `names()`:

```
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"
[6] "Day"
```

Selecting columns

A single column can be selected *by number* or *by name*:

```
> df[, 2]
[1]  35.16 318.90 122.30
> df[, "pop"]
[1]  35.16 318.90 122.30
```

When using a name, R provides a convenient shorthand using $:

```
> df$pop
[1]  35.16 318.90 122.30
```

Selecting columns

A data frame column is a vector, so all of the operations that are valid for vectors are also valid for data frame columns:

```
> mean(df$pop)
[1] 158.7867
> log(df$pop)
[1] 3.559909 5.764878 4.806477
> df$pop/df$admin1
[1] 2.704615 6.252941 3.945161
> df$iso3 == "USA"
[1] FALSE  TRUE FALSE
> df$iso3[df$pop > 200]
[1] USA
Levels: CAN MEX USA
```

Creating columns

You can create a new column in a data frame using one of the assignment operators, similar to how you create a vector:

```
> df$gdp <- c(1827, 16770, 1261)
```

Creating columns

You can create a new column in a data frame using one of the assignment operators, similar to how you create a vector:

```
> df$gdp <- c(1827, 16770, 1261)
```

Similarly, you can create new columns that are functions of existing columns:

```
> df$log_gdp <- log(df$gdp)
> df$gdp_pc <- (1e+09 * df$gdp)/(1e+06 * df$pop)
```

```
> df
  iso3    pop admin1 spanish   gdp  log_gdp   gdp_pc
1  CAN  35.16     13   FALSE  1827 7.510431 51962.46
2  USA 318.90     51   FALSE 16770 9.727347 52587.02
3  MEX 122.30     31    TRUE  1261 7.139660 10310.71
```

Modifying columns

You can also modify existing columns in a data frame using one of the assignment operators

```
> df$gdp <- df$gdp * 1e+09
> df[, "pop"] <- df[, "pop"] * 1e+06
```

```
> df
  iso3       pop admin1 spanish       gdp  log_gdp   gdp_pc
1  CAN  35160000     13   FALSE 1.827e+12 7.510431 51962.46
2  USA 318900000     51   FALSE 1.677e+13 9.727347 52587.02
3  MEX 122300000     31    TRUE 1.261e+12 7.139660 10310.71
```

Removing columns

You can remove a single column from a data frame by assigning it to NULL:

```
> df$log_gdp <- NULL
> df[, "gdp_pc"] <- NULL
```

```
> head(df)
  iso3       pop admin1 spanish       gdp
1  CAN  35160000     13   FALSE 1.827e+12
2  USA 318900000     51   FALSE 1.677e+13
3  MEX 122300000     31    TRUE 1.261e+12
```

Removing columns

This will also work for multiple columns at once:

```
> df[, c("admin1", "spanish")] <- NULL
> head(df)
  iso3       pop       gdp
1  CAN  35160000 1.827e+12
2  USA 318900000 1.677e+13
3  MEX 122300000 1.261e+12
```

Removing columns

Alternatively, you can subset to just the columns you want to keep and then reassign the object:

```
> airquality <- airquality[, c("Ozone", "Wind", "Month", "Day")]
```

```
> head(airquality)
  Ozone Wind Month Day
1    41  7.4     5   1
2    36  8.0     5   2
3    12 12.6     5   3
4    18 11.5     5   4
5    NA 14.3     5   5
6    28 14.9     5   6
```

Selecting rows

A subset of rows of a data frame can be selected *by index* (position):

```
> df[1, ]
  iso3      pop       gdp
1  CAN 35160000 1.827e+12
```

```
> df[2:3, ]
  iso3       pop       gdp
2  USA 318900000 1.677e+13
3  MEX 122300000 1.261e+12
```

```
> df[c(1, 3), ]
  iso3       pop       gdp
1  CAN  35160000 1.827e+12
3  MEX 122300000 1.261e+12
```

Selecting rows

. . . or by *logical statements*:

```
> df[c(T, F, F), ]
  iso3      pop       gdp
1  CAN 35160000 1.827e+12
```

```
> df[df$iso3 != "CAN", ]
  iso3       pop       gdp
2  USA 318900000 1.677e+13
3  MEX 122300000 1.261e+12
```

```
> df[df$gdp < 1e+13, ]
  iso3       pop       gdp
1  CAN  35160000 1.827e+12
3  MEX 122300000 1.261e+12
```

Removing rows

You can remove rows entirely by selecting just the ones you want to keep and then assigning or reassigning to an object:

```
> df <- df[df$gdp < 1e+13, ]
> df
  iso3       pop       gdp
1  CAN  35160000 1.827e+12
3  MEX 122300000 1.261e+12
```

```
> airquality_nomiss <- airquality[!is.na(airquality$Ozone), ]
> nrow(airquality)
[1] 153
> nrow(airquality_nomiss)
[1] 116
```

Things to keep in mind...

Both rows and columns can be selected using a `df[,]` notation.

In this framework, everything *before* the comma tells R what row(s) you want...

```
> df[df$pop > 1e+08, ]
  iso3       pop       gdp
3  MEX 122300000 1.261e+12
```

and everything *after* the comma tells R what column(s) you want:

```
> df[, "iso3"]
[1] CAN MEX
Levels: CAN MEX USA
```

Things to keep in mind...

You can also select rows and columns at the same time using `df[,]` notation:

```
> df[df$pop > 1e+08, "iso3"]
[1] MEX
Levels: CAN MEX USA
```

Things to keep in mind...

You can also select rows and columns at the same time using `df[,]` notation:

```
> df[df$pop > 1e+08, "iso3"]
[1] MEX
Levels: CAN MEX USA
```

or using $ notation:

```
> df$iso3[df$pop > 1e+08]
[1] MEX
Levels: CAN MEX USA
```

Things to keep in mind...

If you want to store the output of these selections you must assign them to an object:

```
> big_pop <- df$iso3[df$pop > 1e+08]
```

```
> airquality_may <- airquality[airquality$Month == 5, ]
```

Things to keep in mind...

If you want to store the output of these selections you must assign them to an object:

```
> big_pop <- df$iso3[df$pop > 1e+08]
```

```
> airquality_may <- airquality[airquality$Month == 5, ]
```

Note that the object you store your selection in might be either a vector or a data.frame, depending on the nature of your selection.

```
> class(big_pop)
[1] "factor"
> class(airquality_may)
[1] "data.frame"
```

Things to keep in mind...

(Re)assignment can also be used to store the output of your selections using the original object name:

```
> nrow(airquality)
[1] 153
> range(airquality$Month)
[1] 5 9
```

```
> airquality <- airquality[airquality$Month == 5, ]
```

```
> nrow(airquality)
[1] 31
> range(airquality$Month)
[1] 5 5
```