Data Frames in S-PLUS

Most data will be stored in data frames, rectangular arrays which usually are formed by combining columns of vectors. **states98** is an example of a data frame.

```
> class(states98)
```

Here is one way to construct a new data frame: first, construct two vectors which will form two of the columns (variables) of the data frame.

```
> x <- seq(0,1,length=10)
> x
> y <- rep(c("a","b"),5)
> y
> my.data <- data.frame(X=x,Y=y)
> my.data
```

Or, to view it in a data window,

```
> guiOpenView("DataFrame","my.data")
```

Or at the menu, **Data** > **Select Data...**.

Add another column to a data frame:

Method 1

```
> my.data <- cbind(my.data, Z=1:10)
```

`cbind()` for *column*-binding.

```
> dim(my.data) # dimension of data frame
```

Method 2

Suppose you want the new column to be $\log(Z)$, which we'll call `logZ`:

```
> my.data$logZ <- log(my.data$Z)
> my.data
```

Method 3

Or, now that we have 4 columns we want a 5th and we don't care about giving it a specific name:

```
> my.data[,5] <-2:11
> my.data
```

Subscripting

To work with portions of the data, use subscripting: **my.data**[*row.index*, *col.index*]

```
> my.data[5,2] # value in row 5, column 2
> my.data[1:5, 1] # rows 1-5 of column 1
> my.data[1:5, "X"] # rows 1-5 of column named "X"
> my.data[1:5, ] #rows 1-5 of all columns
> my.data[c(1,4),c(2,3)] # rows 1, 4 of cols. 2 and 3
> my.data[ ,c(2,3)] # all rows of cols. 2 and 3
> my.data[-c(1,10),1] # exclude rows 1 and 10 of col. 1
> my.data[my.data$Y=="a",1] # rows where Y equals "a", col. 1
> my.data[my.data$Y!="a",3] # rows where Y does not equal "a", col. 3
```

attach

The `attach` command gives you access to the columns of the data frame.

```
> attach(my.data)
> mean(X)
> Y
> Z[c(1,3)] # 1st and 3rd elements of Z
```

However, you will not be able to *change* any of the values of the columns in the data frame with **my.data** attached.

```
> X <- 5:15
> X
```

But take a look at the data frame itself.

```
> my.data
```

To actually make changes to the data in the data frame:

```
> detach("my.data")
> my.data$X <- 21:30 # method 1
> my.data
> my.data[,1] <- 31:40 # method 2
> my.data
```

Working database

S-Plus stores newly created or imported data in the **working database**, or the database that is in position 1 in the list of search paths:

```
> search()

[1] "C:\\Splus6\\users\\math285" "splus"
[3] "stat"    "data"
[5] "trellis" "nlme3"
```

Working database is `math285`. The remaining fold-

ers listed above are S-PLUS provided databases. When S-PLUS looks for an object (say a data set you call), it searches the folders in the order given above and uses the first instance of that object it finds. So, for example, if you have a data set called `dog` in the folder `math285` and there is also a data set called `dog` in folder `stat`, S-PLUS will use the version in `math285`.

> `attach(my.data)`

> `search()`

```
[1] "C:\\Splus6\\users\\math285" "my.data"
[3] "splus"                      "stat"
[5] "data"                       "trellis"
[7] "nlme3"                      "menu"
[9] "sgui"                       "winspj"
[11] "main"
```

Notes:

- `my.data` is now in position 2. That's why we can access the columns of `my.data`, but why we cannot make changes to the columns. **Only data in the database in position 1 can be changed!**
- This also means that if you have a data object called `X` in the working database (`math285`) and you have a column labelled `X` in **my.data**, then calling `X` will invoke the version in `math275` and not the column in **my.data**!
- The working database is listed as `math285` but if you look there, you will not actually see the data you've imported. Technically, the working database is `math285\\.Data`
- Recall that the `objects()` command will give you a listing of the data objects in the working database (position 1 of search path).

  > `objects()`