Learning *R*

Carl James Schwarz

StatMathComp Consulting by Schwarz
cschwarz.stat.sfu.ca @ gmail.com

The LIST data structure

Table of Contents I

R Basics - Lists

R basics - Lists

LISTS are a more general data structure than vectors and dataframes.
A dataframe is a special kind of list where each element of the list is a vector of the same length.

LIST is a collection of elements of any type and of any size, including sub-lists.
A common way to report the results of a modelling function

```
1 fit.cal.fat <- lm( calories ~ fat, data=cereal)
2 str(fit.cal.fat)
```

R basics - Lists

Accessing elements of a list:

```
1  str(fit.cal.fat)
2
3  fit.cal.fat$coefficients  # better to use coef() function
4  fit.cal.fat$coefficients[1]
5  # subtle difference between [k] and [[k]]
6  x <- fit.cal.fat[1]
7  x; str(x)
8  x[1]
9
10 y <- fit.cal.fat[[1]]
11 y; str(y)
12 y[1]
13
14 # lists within lists.
15 fit.cal.fat$model$calories
```

Always try and use the $ syntax for list elements.
Almost always use [[k]] to access list elements by subscripting.

R basics - Lists

Creating a list:

```
1  age <- c(56, 56, 28, 23, 22)
2  height <- c(185, 162, 185, 167, 190)
3  f.names <- c('Carl', "Lois", 'Matthew', 'Marianne', 'David')
4  people <- list(yob=2013-age, height=height, names=f.names)
5  str(people)
6
7  type <- c("dog", "cat")
8  p.names <- c("fido", "roger")
9  pets <- list(species=type, names=p.names)
10 str(pets)
11
12 schwarz <- list(humans=people, animals=pets)
13 str(schwarz)
14 schwarz$humans
15 schwarz$humans$yob
```

See *help(list)*

newlist <- list(elem1=..., elem2=..., elem3=...)

Lists - Exercise

Return to the cereals dataset.

- Read data into a data frame.
- Make a scatter plot of calories vs. grams of fat with a smoother.
- Run a regression of calories vs. grams of fat
- Put the data frame, the scatter plot, and the regression fit into a single list structure with named elements.
- Practise accessing parts of the list.

Lists - Exercise

```
cereal <- read.csv(file.path*(...,'cereal.csv'),
                   header=TRUE, as.is=TRUE,
                   strip.white=TRUE)
cereal[1:5,]
str(cereal)

# Make the plot
plot <- ggplot2::ggplot(data=cereal, aes(x=fat, y=calories))
  ggtitle("Number of calories vs. grams of fat")+
  geom_point( position=position_jitter(h=.5, w=.1))+
  geom_smooth()
plot
```

Lists - Exercise

```
1  # Do the fit
2  reg.fit <- lm(calories ~ fat, data=cereal)
3  reg.fit
4
5
6  # create a list structure
7  results <- list(data=cereal, plot=plot, fit=reg.fit)
```

Lists - Exercise

```
1  names(results)
2
3  str(results[1])
4  str(results[[1]])
5
6  results[[1]][1:5,]
7  results$data[1:5,]
```

Lists - Exercise

```
> names(results)
[1] "data" "plot" "fit"
>
> str(results[1])
List of 1
 $ data:'data.frame': 77 obs. of  15 variables:
  ..$ name    : chr [1:77] "100%_Bran" "100%_Natural_Bran" "
  ..$ mfr     : chr [1:77] "N" "Q" "K" "K" ...
  ..$ type    : chr [1:77] "C" "C" "C" "C" ...

> str(results[[1]])
'data.frame': 77 obs. of  15 variables:
 $ name    : chr  "100%_Bran" "100%_Natural_Bran" "All-Bran"
 $ mfr     : chr  "N" "Q" "K" "K" ...
 $ type    : chr  "C" "C" "C" "C" ...
```

Lists - Exercise

```
> results[1][1:5,]
Error in results[1][1:5, ] : incorrect number of dimensions

> results[[1]][1:5,]
                        name mfr type calories protein fat so
1                  100%_Bran   N    C       60       4   1
2          100%_Natural_Bran   Q    C      110       3   5

> results$data[1:5,]
                        name mfr type calories protein fat so
1                  100%_Bran   N    C       60       4   1
2          100%_Natural_Bran   Q    C      110       3   5
```

Lists - Exercise

```
1  results$plot
2
3  results[3]$coefficients
4  results[[3]]$coefficients
5  results$fit$coefficients
6  summary(results$fit)$r.squared
```

Lists - Exercise

```
> results[3]$coefficients
NULL

> results[[3]]$coefficients
(Intercept)         fat
  95.131579    9.806005

> results$fit$coefficients
(Intercept)         fat
  95.131579    9.806005

> summary(results$fit)$r.squared
[1] 0.2083875
```

Lists - Integration with *plyr* package

A very common paradigm is to use the *plyr* to do long complex computations on chunks of the data and store into a list for later processing

```
1 res <- plyr::dlply(df, "splitvar", function(x){
2      fit <- fitted model on x
3      plot <- specialized plot
4      list(fit=fit, pot=plot})
5
6 report <- plyr::ldply(res, function(x){
7     stuff <- extract from x$fit
8     data.frame(good stuff)
9 })
```

Refer to advanced usage of the *plyr* package for details.

Lists - Summary

- Common output data structure from modelling functions.
- Useful for storing different types of data structures, e.g. plots + data.frame + model results
- Data frames are special type of list.
- Use $ or [[]] to access list elements.