Additional items

The purpose of this section is to provide a list of additional items which are important but did not necessarily fit into the other sections. The program used in this section is Add.R.

- Search path: The `search()` function displays the *search path* for R. For example, suppose two packages contain functions with the same name and both packages are loaded into R. The function from the package that appears first in the search path is the one that will be used.

```
> library(package = "MRCV")
> library(package = "binGroup")
> search()

 [1] ".GlobalEnv"         "package:binGroup"   "package:MRCV"
 [4] "package:knitr"      "package:stats"      "package:graphics"
 [7] "package:grDevices" "package:utils"      "package:datasets"
[10] "Autoloads"          "package:base"
```

  R looks first for functions from the `binGroup` package.

- Working directory (folder): The `setwd()` function sets the working directory for all R files that you read in or write out. This can be helpful when one needs to read in a large number of data files or when one needs to use `source()` for a large number of programs.

```
> setwd(dir = "C:\\data")  # On my computer
> gpa <- read.table(file = "GPA.txt", header=TRUE, sep = "")
> head(gpa)

  HS.GPA College.GPA
1   3.04        3.10
2   2.35        2.30
3   2.70        3.00
```

```
4   2.55        2.45
5   2.83        2.50
6   4.32        3.70
```

- Reading in data files on the Internet: Data files can be read into R directly using the usual functions but with an Internet address given in the file argument.

```
> # From Analysis of Categorical Data with R website
> gpa <- read.table(file = "http://www.chrisbilder.com/categorical/ApdxA/gpa
 header = TRUE, sep = "")
> head(gpa)

  HS.GPA College.GPA
1   3.04        3.10
2   2.35        2.30
3   2.70        3.00
4   2.55        2.45
5   2.83        2.50
6   4.32        3.70
```

  Unfortunately, the full address does not show in the above code. The web address is `http://www.chrisbilder.com/categorical/ApdxA/gpa.txt`.

- Debugging: A simple way to debug a program is to use `traceback()`. This function will display what functions were run prior to an error.

```
> qnorm(p = c(0.025, 0.975), mean = 0, sd = 1)

[1] -1.959964  1.959964

> pnorm(q = c(-1.96, 1.96), mean = 0, sd = 1)

[1] 0.0249979 0.9750021

> qnorm(mean = 0, sd = 1)  # Code is incorrect
```

```
Error in qnorm(mean = 0, sd = 1):  argument "p" is missing, with no default

> traceback()

No traceback available
```

Unfortunately, `traceback()` did not provide useful information when I ran it via LyX. If you run the code in my program, you will obtain some useful information.

- Three dimensional or higher data objects: A matrix is a two dimensional array that stores information. The `array()` function can be used to to more generally store information in a higher number of dimensions. For example, this can be helpful to store data in a contingency table format:

```
> #' Contingency tables - Data entered by column with strata
> #   Data is used in Chapter 4 of Analysis of Categorical Data
> c.table <- array(data = c(44, 47, 118, 23, 32, 18, 28, 86, 39, 48, 36, 34,
 18, 23, 12, 18, 62, 45, 51), dim = c(5,2,2), dimnames = list(Ideology =
 c("VL", "SL", "M", "SC", "VC"), Party = c("Democrat", "Republican"), gender
 c("Female", "Male")))
> c.table

, , gender = Female

        Party
Ideology Democrat Republican
      VL       44         18
      SL       47         28
      M       118         86
      SC       23         39
      VC       32         48

, , gender = Male

        Party
Ideology Democrat Republican
      VL       36         12
```

```
      SL       34         18
      M        53         62
      SC       18         45
      VC       23         51

> c.table[1,1,1] # row 1, column 1, and strata 1

[1] 44

> c.table[1,1,]  # row 1 and column 1

Female   Male
    44     36

> c.table[1,,1]  # row 1 of strata 1

  Democrat Republican
        44         18

> c.table[,1,1]  # column 1 of strata 1

 VL  SL   M  SC  VC
 44  47 118  23  32

> c.table[,,1]   # strata = 1

        Party
Ideology Democrat Republican
      VL       44         18
      SL       47         28
      M       118         86
      SC       23         39
      VC       32         48
```

- Chaining commands with a pipeline: The `magrittr` package allows one to put commands together into essentially one command.[1] This is called creating a pipeline (or pipe for short). An example of a place outside of R that can use this type of

[1] See `https://mran.microsoft.com/web/packages/magrittr/vignettes/magrittr.html` for a vignette

syntax is Linux. Those who prefer this type of syntax think it can made code easier to read. Below are a few simplistic ways of using a pipe:

```
> library(package = magrittr)
> qnorm(p = c(0.025, 0.975), mean = 0, sd = 1) %>% pnorm(mean = 0, sd = 1)

[1] 0.025 0.975

> cereal <- read.csv(file = "C:\\data\\cereal.csv")
> cereal$sugar <- cereal$sugar_g/cereal$size_g
> cereal$fat <- cereal$fat_g/cereal$size_g
> cereal$sodium <- cereal$sodium_mg/cereal$size_g
> subset(x = cereal, subset = Shelf == 1, select = c(sugar, fat)) %>%
 lm(formula = sugar ~ fat, data = .)


Call:
lm(formula = sugar ~ fat, data = .)

Coefficients:
(Intercept)          fat
     0.2225       1.3156

> subset(x = cereal, subset = Shelf == 1, select = c(sugar, fat)) %$%
 cor(sugar, fat)

[1] 0.264103
```

The %>% operator pipes a command from the left to the right. By default, the result from the left side goes into what would normally be the first argument on the right. When one wants to put the left side result somewhere else, a period can be used to denote it on the right side (second example). Also, when one wants to use components of a list, the %$% operator can be used for the pipe (third example).

- Creating groups for a vector: The `cut()` function groups data

by chosen intervals.

```
> set.seed(7121)
> y <- rnorm(n = 100, mean = 0, sd = 1)
> groups <- cut(x = y, breaks = -3:3)
> head(groups)

[1] (-1,0]  (0,1]   (-2,-1] (-1,0]  (0,1]   (1,2]
Levels: (-3,-2] (-2,-1] (-1,0] (0,1] (1,2] (2,3]

> table(groups)

groups
(-3,-2] (-2,-1]  (-1,0]   (0,1]   (1,2]   (2,3]
      1      11      40      34      14       0
```