Exporting Data and Output from R

Data Set `cpds.csv`

To illustrate the different data exporting possibilities, as well as writing output features in R, we are going to use the file `cpds.csv` located in the github repo:

https://raw.githubusercontent.com/ucb-stat243/stat243-fall-2016/master/data/cpds.csv

To have a working example, let's subset the data set for country `Australia`

```
# assuming that you already have the data in your R session
dat <- read.csv('cpds.csv')

# subset lines for Australia
australia <- subset(dat, country == "Australia")
```

Writing tables

One common task in most data analysis projects involves exporting processed data tables (e.g. clean data sets, or subsets). You can use either `write.table()` or `write.csv()`.

```
# blank separated (default)
write.table(australia, file = 'australia.txt', row.names = FALSE)

# tab-separated value
write.table(australia, file = 'australia.tsv', sep = "\t", row.names = FALSE)

# comma-separated value
write.csv(australia, file = 'australia.csv', row.names = FALSE)
```

Sending output with `cat()`

You can use `cat()` to concatenate and print information to a file. For instance, say you are interested in some descriptive statistics about `unemp` (unemployment):

```
# summary statistics of unemp
min(australia$unemp)
max(australia$unemp)
median(australia$unemp)
mean(australia$unemp)
sd(australia$unemp)
```

The goal is to generate a file `australia-statistics.txt` with the following contents:

```
Australia Unemployment Statistics

Minimum: 1.15
Maximum: 10.90
Median : 5.78
Mean   : 5.52
Std Dev: 2.79
```

Here's one way to start:

```
outfile <- "australia-statistics.txt"
file.create(outfile)

# first line of the file
cat("Australia Unemployment Statistics\n\n", file = outfile)

# subsequent lines appended to the output file
cat("Minimum:", aus_min, "\n", file = outfile, append = TRUE)
cat("Maximum:", aus_max, "\n", file = outfile, append = TRUE)
cat("Median :", aus_med, "\n", file = outfile, append = TRUE)
cat("Mean   :", aus_avg, "\n", file = outfile, append = TRUE)
cat("Std Dev:", aus_sd, "\n", file = outfile, append = TRUE)
```

To make it "prettier" you may consider using `sprintf()`

```
sprintf('Minimum: %s', aus_min)
```

Now let's re-export the lines:

```
cat("Australia Unemployment Statistics\n\n", file = outfile)
cat(sprintf('Minimum: %0.2f', aus_min), "\n", file = outfile, append = TRUE)
cat(sprintf('Maximum: %0.2f', aus_max), "\n", file = outfile, append = TRUE)
cat(sprintf('Median : %0.2f', aus_med), "\n", file = outfile, append = TRUE)
cat(sprintf('Mean   : %0.2f', aus_avg), "\n", file = outfile, append = TRUE)
cat(sprintf('Std Dev: %0.2f', aus_sd), "\n", file = outfile, append = TRUE)
```

Your turn: How would you avoid writing that many calls to `cat()`?

Sending R output to a file with `sink()`

Another interesting function is `sink()`. This function is very useful when you want to export R output as is displayed in the R console. For example, consider the output from `summary()`

```
summary(australia)
```

You could assign the output of `summary(australia)` to an object and then try `writeLines()` to export the results to a file `australia-summary.txt`, but you won't keep the same format of R:

```
aus_summary <- summary(australia)
writeLines(aus_summary, con = "australia-summary.txt")
```

To be able to keep the same output display of R, you must use `sink()`. This function will **divert** R output to the specified file:

```
# sink output
sink(file = "australia-stats2.txt")
# summary statistics of unemp
summary(australia)
# stops diverting output
sink()
```

Your turn: Use `sink()` to send the output from running a linear regression of `unemp` on `realgdpgr` with the function `lm()`. Also export the results from using `summary()` on the regression object. And/or try running a t-test between `unemp` and `realgdpgr` with `t.test()`.

Exporting tables with `xtable()`

Another interesting tool to export tables in LaTeX or HTML formats is provided by the R package `"xtable"` and its main function `xtable()`.

```
library(xtable)

# linear regression
reg <- lm(realgdpgr ~ unemp, data = dat)

# create xtable and export it
reg_table <- xtable(reg)
print(reg_table, type = "latex", file = "reg-table.tex")
print(reg_table, type = "html", file = "reg-table.html")
```

R's Binary Data

R also allows you to save objects in R's binary format with the functions `save()` and `save.image()`. It is customary to use the `RData` extension for the files created by `save()` and `save.image()`. You may also encounter users specifying the old extension `.rda` or some other variation.

You can use `save()` to save specific objects from your current session. For example, here is how to save the data frame `australia`:

```
save(australia, file = 'australia.RData')
```

The difference between `save()` and `save.iumage()` is that the latter saves all the objects in your current session. This is actually the function that is run behind the scenes everytime you quit R and accept to save the so-called *workspace image*.

You can share `australia.RData` with any other R user, regardless of the operating system that they use. To read in binary R files, use `load()`.

Your turn: Subset the data set for another country (not Australia) and export the data using both `write.table()` and `save()`. Compare the size of the produced files.