This document contains some of the difficulties I’ve come across when working on case study 2. For some of them, I know how and why the errors occurred. For some of them, I only know how to avoid the errors.

Let’s start with one in the function ‘cy.getLocationSalary’. After we followed the codes in the textbook, we found that locations listed in outcomes are not formatted: We can see spaces and “\r\n” following by the locations. We need to use “strsplit” to get rid of “\r\n” following by locations in the results first. By using 'unlist', we can extract the second element in location, which is the exact location with spaces (The first and third elements are blank with spaces). We then use “gsub” to remove the spaces within the location. These two steps cannot be done in one step, otherwise it will somehow dropped locations and return “” as results.

```{r}

cy.getLocationSalary = function(doc)

{

rawans = xpathSApply(doc, "//div[@class = 'job-info-main'][1]/div", xmlValue)

ans<-strsplit(rawans,"\r\n")

ans[1]<-gsub("[[:space:]]", "",unlist(strsplit(rawans[1],"\r\n"))[2])

names(ans) = c("location", "salary")

ans

}

```

In function “cy.getPostLinks”, we have to use “unnamed” command to remove things like '/data-scientist-job-230135' for each link. They act as the names of each of the links listed.

The reason why we have to remove them is because we will use “htmlParse” command on the outcomes (which are URL links). Things like '/data-scientist-job-230135' are not hyperlinks, which will generate errors when using “htmlParse”.

```{r}

cy.getPostLinks = function(doc, baseURL = 'http://www.cybercoders.com/search/')

{

if(is.character(doc))

doc = htmlParse(doc)

links = getNodeSet(doc, "//div[@class = 'job-title']/a/@href")

unname(getRelativeURL(as.character(links), baseURL))

}

```

In function “cy.getNextPageLink”, the function didn’t work well at first. The URL it generated didn’t take us to the next page. We later figured out that it was because the character “./” lied between “search” and “?page=” in the URL returned by “getRelativeURL” . We need “gsub” to remove the character. Here I am not quite sure why do I have to substitute “./” with “search/” but it actually works perfectly.

```{r}

cy.getNextPageLink = function(doc, baseURL = docName(doc))

{

if(is.na(baseURL))

baseURL = "http://www.cybercoders.com/search"

link = getNodeSet(doc, "//a[@rel='next']/@href")

if(length(link) == 0) return(character())

else return(gsub("/./","search/",getRelativeURL(link[[1]],baseURL)))

}

```

In the process of reading in sample links and extracting information, I found a mistake of using “lapply” and “sapply”. We have to use lapply rather than sapply when using cy.readPost although there seemed to be no difference at this step. That is because the outcomes “dsPosts” we get by using “lapply” are values. If we use “sapply”, the outcomes are data, which may generate errors such as 'subscript out of bounds' in the future process. (That’s the only reason I can explain.)

```{r}

dataSciPosts = cyberCodersLinks("Data Scientist")

dataSPosts = cyberCodersLinks("Data Science")

combined = unique(c(dataSciPosts,dataSPosts))

set.seed(2015)

sample25<-sample(1:length(combined),25,replace = F)

sampleLinks<-unlist(sapply(sample25, function(x) combined[x]))

Links = sapply(sampleLinks, htmlParse)

dsPosts = lapply(Links, cy.readPost)

```

When plotting the dochart, I tried to use “as.integer” to extract the frequencies of each skill and use “dotchart” to make plot and rename Y label to replace the original Y label (which is the order of each skill) with skill names. However, I found the skills and their frequencies didn’t match. We don't have to use “as.integer” in dotchart command. Although there will be warning message showing up, the plot it generated has the name of different skills as y labels and has sorted in descending order. We don't have to worry about matching skills with their frequencies.

We also notice that we need to enlarge the size of window when drawing word cloud plot in order to avoid the case that some of the key words are too long to fit in the plot.

```{r}

windows(width=12,height=12)

dotchart(sort((cyber.DsSkills[cyber.DsSkills > 5])),

main = "Skills from Data Scientist on CyberCoders")

library(wordcloud)

windows(width=12,height=12)

wordcloud(names(cyber.DsSkills), cyber.DsSkills)

```

When we dealing with the salaries, we tried to extract the lower bound and upper bound of salaries under the instructions in the book. However, it didn’t work because the regular expressions given in the book might not be right. Therefore, we use another way to do it. We first “unname” the list of salaries to make it neat. We then use “strsplit” to extract numbers from the list of salaries and make them numeric and we also need to remove the items which didn’t mention the number in the salary. We then put the numbers in a data frame which has two columns called “Lower Bound” and “Higher Bound”.

In order to use “kable” command, we need to give the name of each column. Otherwise, “kable” command won’t work.

```{r}

cy.processSalary =

function(posts)

{

tmp = unname(sapply(posts,`[[`,'salary'))

vals = na.omit(as.numeric(unlist(strsplit(unlist(tmp), "[^0-9]+"))))\*1000

ans = data.frame(matrix(vals, nrow=length(vals)/2, byrow=T))

names(ans) = c('Lower Bound', 'Higher Bound')

ans

}

```

```{r}

sl = cy.processSalary(dsPosts)

kable(summary(sl), format = 'markdown', caption = "Salary Summary")

```

When we attempted to draw the pie chart of the frequency of locations, we had difficulty extracting the frequency from the output of ‘summary’ command because the results are stored in one column. Therefore, we try to use ‘table’ function and ‘as.data.frame’ to separate the results into two columns. We can then use the elements in the columns to draw the pie chart.

```{r}

cyber.DsLocation <- unname(unlist(lapply(dsPosts,`[[`,"location")))

cyber.DsLocation

for (i in 1:25)

cyber.DsLocation[i] <- (unlist(strsplit(cyber.DsLocation[i],","))[2])

cyber.DsLocation <- as.data.frame(cyber.DsLocation)

kable(summary(cyber.DsLocation), format = 'markdown')

cyber.DsLocation <- table(cyber.DsLocation)

cyber.DsLocation <- as.data.frame(cyber.DsLocation)

names(cyber.DsLocation) <- c("State","Count")

library(ggplot2)

attach(cyber.DsLocation)

ggplot(cyber.DsLocation, aes(x='',y=Count, fill=State))+labs(title="Click by sign group (05/15-05/30) Signed in",fill="Clicks")+geom\_bar(width = 1,stat="identity")+coord\_polar(theta="y")

```