RData 저장 및 읽기/ save() / load()

R을 사용하며 느꼈던 가장 큰 단점은 당연, 메모리 관리이다.

그 다음으로는 "연산 속도" 그리고 "코드의 효율성"이다.

R은 설계 특성상 매우 큰 데이터를 이용해 작업을 수행할 때 문제가 발생할 수 있다.

이는 데이터를 물리적 메모리에 저장해야 하기 때문이다.

쉽게 말해, R은 모든 연산을 메모리에 올려두고 실행하기 때문에, 종종 메모리 및 연산량 문제로 R Studio가 강제 종료된다.

R Studio가 강제 종료되면 작업중이던 메모리가 다 삭제되기 때문에, 기껏 만들어 놨던 여러 결과 프레임들을 다시 뽑아야하는 수고로움이 따른다.

프로젝트 수행으로 데이터 사이즈 5.2GB에 feature 56개, obs 약 33만개의 데이터로 분석작업을 했다.

메모리가 8GB여서, 이 프로젝트를 수행할 당시 메모리 문제와 속도 문제를 가장 많이 겪었다. 이 문제들을 해결하면서 알게된 몇가지 팁들을 이번 포스팅에서 공유하고자 한다.

R의 메모리 문제를 해결하기 위해서, 비교적 큰 데이터는 RData로 저장해놓고 읽어오는 것이 효율적인 방법이라고 할 수 있다.

RData 저장

```
## save( 저장할 데이터명, file = "저장 경로\\저장할 데이터의 파일명.RData" )

save(work_data, file = "C:\\Users\\Hyejin\\Desktop\\work_data.RData")
```

RData 읽기

```
## load( file = "경로\\파일명.RData" )
load(file="C:\\Users\\Hyejin\\Desktop\\work_data.RData")
```

작업공간의 모든 객체 저장 및 읽기

하나의 데이터 뿐만 아니라, 모든 작업 내역을 그대로 저장할 수 있다. save() 대신 save.image()를 사용하면 된다.

```
## 저장 save.image(file = "저장경로\\파일명.RData")
## 읽기 load( "저장경로\\파일명.RData" )
```

사용자 정의 함수 저장

많은 함수를 작성한 경우에는, 이 함수들을 하나의 R 파일로 저장한 후에 필요할때 마다 source() 함수를 이용해서 불러오는 것이 좋다.

한글이 포함된 경우, 인코딩 오류를 방지하기 위해 다음의 옵션을 사용하면 된다. => encoding="utf-8"

```
source('myFunction.R', encoding='utf-8')
```