

R 개요

특징

- 무료, 오픈소스
- 다양한 컴퓨터 플랫폼에서 사용
- 응용영역 꾸준히 확장 / 머신러닝, 데이터마이닝 등
- 인터프리터 언어 : 컴파일 과정이 필요없고, 직접 실행
- 뛰어난 그래픽 처리 능력
- 다양한 데이터 원천으로부터 데이터 불러오기 쉽게 가능

R 이해하기

작업환경

- 현재의 작업 디렉토리 관련
 - `getwd()` : 현재 작업 디렉토리 리턴
 - `setwd(디렉토리)` : 작업디렉토리 변경
- `RProfile.site` 파일 : R 시작시 실행시킬 수 있는 옵션 저장
 - `C:\Program Files\R<R-Version>\etc` 폴더에 위치
 - 원하는 코드를 추가해놓으면, 새로운 R세션 시작시마다 자동 실행
- R 종료 : `q()` 함수
- 중간중간에 작업공간 저장 : `save.image()`
 - `save()` 함수 이용시 개별 변수를 파일로 저장
 - `save(변수/함수이름, file = "파일이름")`
 - `load()` 함수 이용시 불러오기 가능
 - `load("파일이름")`
- 현재 작업공간에 생성된 변수/함수/리스트 보기(문자 벡터 형태) : `ls()`
 - 아무것도 없으면 `character(0)` 반환
 - 있을 시 해당 변수/함수의 이름을 모두 반환
 - `ls.str()` 입력시 더 자세한 정보(구조까지)
- 생성한 함수가 더 이상 필요 없을때 `rm(변수/함수 이름)` 사용
 - 작업공간에서 영원히 제거, 여러개 동시에 제거 가능(콤마로 구분)
 - 인수로 `list = ls()`를 넣으면 모든 객체 한꺼번에 삭제
- 최근 사용한 명령어 보기 : `history`(라인의 개수)
 - `Inf`를 인수로 넘기면 기록된 전체 명령어 라인 출력
 - 실행했던 명령들은 작업디렉토리에 `.Rhistory` 파일로 저장
- 바로 직전 실행했던 명령어의 실행 결과 : **.Last.value**에 저장
- 명명규칙
 - 모든 이름은 문자/마침표로 시작, 문자/숫자/밑줄,마침표만 사용
 - 키워드는 사용 불가

- 대소문자 구별
- 주로 변수 이름에는 Dotted Style, 함수 이름에는 Camel Case 사용

패키지

- R에서 제공되는 함수와 데이터셋은 모두 패키지에 보관
 - 기본적으로 표준패키지가 함께 설치됨
- 어떤 패키지가 설치 되었는지 보기 : `library()`
 - 인수없이 사용시 설치된 패키지 리스트가 설명과 함께 팝업 창(Source 영역)
- `installed.packages()` : 설치된 패키지에 대한 좀 더 자세한 정보(행렬 형태)
 - 행 = 설치된 패키지 / 열 = 패키지에 대한 정보
- 어떤 패키지의 함수/데이터셋을 이용하려고 할 시 해당 패키지가 메모리에 적재되어있어야 함
 - 표준패키지가 설치된 데있더라도 자동으로 메모리에 적재되지는 않음
 - `search()` 함수 : 메모리에 적재된 패키지 리스트 출력(문자열 벡터 반환)
 - `.GlobalEnv` : 사용자가 생성한 변수나 함수가 저장된 작업공간
 - 나머지는 `package : <package name>`의 형태로 저장
 - 반환 된 문자열 벡터 = 탐색경로 이 벡터의 순서대로 요청받은 데이터셋/함수를 찾음
 - base 패키지에서 제공하는 함수의 이름과 같은 함수를 사용자가 정의할 경우, 사용자가 정의한 함수를 우선적으로 실행(작업공간에 있기 때문)
 - Namespace 활용 가능 <패키지이름>::함수 를 사용하면 해당 패키지의 함수 실행
 - `library(패키지이름)` 사용시 해당 패키지를 메모리에 적재함
 - 이미 적재되어있는 애를 다시 적재해도 무방(강 알아서 실행 안함)
 - `detach(package=패키지이름)` 사용시 해당 패키지를 메모리에서 제거
- 패키지 설치 는 `install.packages("패키지이름")` 함수 이용
 - CRAN 사이트로부터 해당 패키지를 받아옴
- 패키지가 설치 된 위치 : `.libPaths()` 함수
 - 새로운 경로를 지정하면 로컬 라이브러리에 패키지 저장 가능
- 패키지에 대한 사용방법/정보 확인 : `help(package="패키지이름")` 또는 `library(help=패키지이름) ls("package:패키지이름")`은 해당 패키지에 포함된 객체정보 확인(메모리에 적재되어있어야 함)
- `data()` : 현재 보유하고 있는 데이터셋 확인
 - 인자로 데이터셋 이름을 입력해 내용 확인
 - 인자로 `package = "패키지이름"` 전달 가능 -> 해당 패키지의 데이터셋 중 검색/모든 데이터셋 출력
 - `head(데이터셋)` 입력시 앞쪽 여섯개 데이터, `tail(데이터셋)` 입력시 뒤쪽 여섯개 데이터 출력
 - `head(데이터셋,숫자)`, `tail(데이터셋,숫자)`로 원하는 숫자만큼 데이터 출력
 - `help(데이터셋)`으로 해당 데이터셋에 대한 자세한 정보 확인 = ?데이터셋

도움말

- `help.start()` 함수 실행시 다양한 매뉴얼의 인터넷 창 ON
- 인수에 대한 간단한 정보 : `args(함수이름)`
- 함수의 예제 코드 : `example(함수이름)`
- 컴퓨터에 설치되어있는 모든 패키지중 검색 `help.search("패키지이름")`

- ??패키지와 같은 기능, 해당 문자열이 포함된 것들을 모두 찾아줌
 - 정규표현식도 이용가능 ex) q\$: q로 끝나는 것, [7-9] : 7~9포함, xy+ : xy포함
- 특정 패키지의 함수 검색도 가능 help(함수, package = "패키지이름")
- RSiteSearch() : r-project의 검색에진 활용 : 해당 검색어에 대한 결과 웹페이지 반환
- findFn("검색어") : 검색어에 대한 검색결과가 테이블 형식으로 출력

데이터 구조

- 데이터 분석을 위해 필요한 첫 작업 : 분석정보가 포함된 데이터셋을 분석에 적합한 형식으로 생성
- 데이터 구조를 선택, 데이터 입력/불러오기
- 벡터, 팩터, 행렬, 배열, 데이터프레임, 리스트 등 다양한 데이터 구조를 가짐
 - 각자 데이터 유형, 생성 방식, 구조적 복잡성, 원소접근 방법에 있어서 서로 다른 특성을 지님
- 벡터 : **동일한 유형**의 데이터를 원소로 갖는 데이터 집합
 - 숫자로 구성 -> 숫자 벡터
 - 문자로 구성 -> 문자 벡터
 - 논리값으로 구성 -> 논리 벡터
- 팩터 : 벡터의 한 특수한 형태 / 범주형 값을 원소로 갖는 데이터 집합
 - 범주값의 순서가 의미 있을 때 : 서열 팩터
- 행렬 : 두개의 차원으로 구성된 **2차원 벡터**, 벡터로 구성되기 때문에 다 동일한 데이터 타입
- 배열 : 행렬을 확장한 형태, 세 개 이상의 차원을 가질 수 있음
- 데이터프레임 : 행렬과 유사하지만, 각 열은 서로 다른 데이터 유형을 가질 수 있음 -> 유연성이 높다
 - DB, 통계페이지에서 일반적으로 다루는 데이터셋과 유사한 구조
 - R에서 가장 빈번하게 다루는 데이터 구조
- 리스트 : 가장 복잡한 형태의 구조, 객체집합으로서 다양한 객체를 하나의 이름 아래 저장
 - 서로 다른 데이터 유형을 한꺼번에 수용 가능