R 스터디 가이드
- 도움이 되길 바랍니다 -
(이 문서의 구조와 내용은 수시로 업데이트되며 변경됨을 미리 알아주시길 부탁드립니다)

iHELP Working Group
Chel Hee Lee & Eugene Jung

May 4, 2013

Part I

R을 처음 접하는 분들을 위하여

Chapter 1

시작하면서

통계소프트웨어 R을 사용하는데 이 문서가 도움이 되길 바랍니다. 아래와 같은 내용에 중점을 두고자 하였습니다.

- 원리중심의 문제해결을 설명하고자, 다양한 패키지들로부터 제공되는 함수들에 대한 사용법에 대한 설명은 가급적이면 피하려고 하였습니다. 따라서, 이 문서내에서는 특정 패키지가 해결할 수 있는 경우를 제외하고는 BASE (기본) 시스템을 이용하여 문제를 해결할 수 있도록 정리하려고 노력합니다.
- R이라는 언어의 특징을 살린 설명을 하고자 하였습니다. 예를들면, R의 가장 큰 특징이라고 할 수 있는 벡터라는 개념을 활용하여 원리 중심의 설명을 제공합니다.
- 통계와 전산에 관련된 전문용어는 쉽게 풀어쓰거나, 용어에 대한 이해를 돕기 위한 참고 자료를 제공합니다.

본 문서와 관련한 프로젝트의 시작일은 2013년 4월 10일이며, iHELP Working Group 관리자에 의하여 수시로 갱신되고 있습니다. 따라서, 이 문서는 어떤 특정한 기간을 두고 완성되으며, 오로지 업데이트된 버전만이 존재합니다.

이 문서는 2005년 이래로 R의 사용에 대한 본 문서의 지은이 개인의 경험과 R Documentation 만을 토대로 하여 작성되었으므로 아직 경험하지 못하여 다루지 못하는 부분이 있습니다. 이 문서의 지은이는 통계학에 대한 배경지식을 가지고 있으므로 문서의 내용이 다소 통계 및 수학 분야에 치중하였을 수 있습니다. 문서가 보다 다양한 계층의 분들에게 R을 사용하는데 도움이 되고자, 이 문서를 읽고 있는 독자가 문서 내용에 대한 추가, 수정, 및 제안이 있다면 이를 ihelp-urquestion@lists.r-forge.r-project.org 주소로 이메일을 보내주신다면 감사하겠습니다.

또한, 아래에 기재된 분들의 도움이 없었다면 이 문서가 발전될 수 없었기에 그 감사의 말씀을 올리고 싶습니다.

- 이부일 박사님 (충남대학교 정보통계학과)
- 신종화 교수님 (서울종합과학대학원 사회학과)

이 문서를 읽는 방법:

- "완전초보에요"라는 챕터와 "기초프로그래밍과 운영체제" 챕터는 이 문서를 읽기 전에 반드시 먼저 읽으셔야 합니다.

Chapter 2

완전 초보에요

이 문서에서 "초보"라는 의미는 아래에 나열된 사항들중 두 가지 이상에 해당되시는 분들을 의미합니다.

- R 이라는 프로그래밍 언어 이전에 다른 프로그래밍 언어에 대한 경험이 전무하신 분,
- 유닉스와 리눅스 시스템에 익숙하지 않으신 분,
- 기초 통계 분석에 대한 도움이 필요하신 분,
- 그냥 무엇을 해야할지 막막함에 쌓여 계신분

이에 해당하시는 분들은 꼭 "사용전 반드시 알아야 할 7가지 숙지사항" 섹션을 꼭 읽어주시길 부탁드립니다. R을 사용하시는데 있어 숙지사항의 내용을 기억하신다면, 매우 도움이 될 것입니다.

2.1 꼭 먼저 알아야 할 7가지

초보라고 생각하시는 분들께서는 아래의 내용들에 대한 개념적 숙지해주시길 부탁드립니다.

1. **데이터 입력과 처리:** R 에서는 이용되는 모든 데이터들에 대한 처리는 열방향으로 이루어집니다. 이 말의 뜻은 데이터의 입력 및 변형, 그리고 연산에 사용되는 데이터들에 대한 처리순서는 열방향으로 나열된 후에 이루어지는 것을 말합니다. 예를들어, 1부터 12까지 12개의 정수로 이루어진 수열 (즉, $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$)을 수학적인 표현으로 4행 3열의 행렬은 R은 아래와 같이 이해합니다.

   ```
   1  5   9
   2  6  10
   3  7  11
   4  8  12
   ```

   따라서, 이 행렬에서 5라는 숫자값은 행렬의 1행 2열에 위치하고 있다고 할 수 있으며, 행렬의 5번째의 값이라고도 할 수 있습니다.

2. **대화식 사용과 결과를 보는 방법에 대해서:** R은 사용자가 주어지는 업무를 시키는 대로만 수행하는 유용한 프로그램일뿐, 그 이상의 내용은 수행하지 않는다는 점을 반드시 명심하셔야 합니다. 따라서, R 프로그램을 시작하게 되면 아래와 같은 기호로 표시되는 프롬프트 (즉, 사용자의 명령어를 기다리는 기호)를 보여주게 됩니다.

   ```
   >
   ```

모든 명령어는 > 기호 뒤에 작성하게 됩니다.

가장 중요한 것은 분석자가 머릿속으로 상상하거나 기대하는 업무가 있다면, 분석자는 반드시 그 업무가 이루어지는 프로세스에 대해서 잘 알고 있어야 합니다. 따라서, R은 다른 통계 소프트웨어들과 같이 분석된 결과를 미리 보여주거나 혹은 분석이 된 모든 결과를 한 번에 다 보여주지 않습니다. 분석자가 확인하고자 하는 결과를 R에서 제공하는 함수를 통하여 중간 결과물들을 확인할 수 있습니다. 우리는 분석자가 작업을 어떻게 진행하고 확인할 것인가에 대한 프로세스 차트를 먼저 작성한뒤 분석을 수행하기를 권장합니다.

3. **통계분석의 절차:** 분석이라 것은 데이터에 대한 이해를 통하여 데이터가 가진 특징을 수학적 표현으로서 설명하기 위한 과정입니다. 따라서, 데이터에 대한 직관적인 이해를 위해서 시각화 작업과 통계모형이 요구하는 데이터 형식을 만드는 것이 중요합니다. 이를 전처리 과정이라고 하며, 통계분석은 크게 아래와 같은 절차를 밟아 이루어집니다.

   - 데이터 입출력과 클리닝, 그리고 분석 전처리 관련 테크닉들
   - 분석전 탐색적 시각화 작업
   - 통계모형의 결정 및 적용
   - 모형 적용 후의 보고서 생성 및 시각화 작업
   - 통계 모형 자체의 개발 또는 자동화 시스템 구축

   따라서, 이 문서는 위에서 설명한 과정에 해당하는 순서대로 챕터들을 구성하였습니다.

4. **한글 표현과 인코딩에 관련하여:**

   R은 한국어 사용자를 위한 한국어 인터페이스를 지원하고 있습니다. **그러나, 사용자는 이러한 한국어 지원이 단순히 사용자의 편의를 위한 선택적 사항이라는 점을 반드시 알고 있어야 합니다**. 또한, 분석시 본래의 정교한 표현은 번역된 한국어 보다는 원래의 영문이라는 점도 잊지 마셔야 합니다.

   현재 한국어에 관련된 작업은 UTF-8이라는 인코딩에 기반하여 한국어 품질관리 프로그램 (http://ihelp.r-forge.r-
   을 통하여 이루어 지고 있으나, R을 한국어가 아닌 영문으로 설정하기 위해서는 다음의 주소를 눌러 그 내용을 확인해주시길 부탁드립니다. 이 내용은 버전에 관계없이 일반적으로 통용되는 방법이나 윈도우즈 사용자에게 맞추어 작성되었습니다.

   http://lists.r-forge.r-project.org/pipermail/ihelp-urquestion/2013-April/000003.html

   본래 데이터를 자국의 언어로 표현하는 방법은 UTF-8 이라는 인코딩을 이용하므로, 간혹 데이터가 올바르게 보이지 않을 경우가 있습니다. 이러한 문제를 해결하기 위해서는 "R을 지원하는 인코딩 중 올바르게 한국어를 표현해주는 코드를 찾는 방법" (http://lists.r-forge.r-project.org/pipermail/ihelp-urquestio
   를 읽어보시길 바랍니다.

   간혹, 잘못된 한글처리가 시스템 에러를 야기하는 경우가 있으므로, 이러한 경우에 한국어로 R을 사용하시는 분들께서는 다소 번거로우실지라도 보다 안정적이고 나은 R을 제공하기 위하여 그 내용을 ihelp-urquestion@lists.r-forge.r-project.org 주소로 이메일을 보내주신다면 감사하겠습니다.

   단, 이러한 한글처리는 분석에 연관된 수치연산과는 아무런 관계가 없음을 반드시 알아주시길 부탁드립니다.

5. **대소문자 구분에 관련하여:** 많은 분들이 이전에 SAS 소프트웨어를 사용하여 분석을 수행하셨을 것입니다. SAS에서는 대소문자를 구분하지 않고 프로그램을 작성하게 되지만, R에서는 대소문자를 구분하므로 A 라는 변수와 a 라는 변수는 서로 다른 것임을 명심하시길 부탁드립니다.

6. **운영체제와 관련하여:** R은 기본적으로 Unix (유닉스)와 같은 환경에서 작성되었습니다.

7. **패키지와 관련하여:** R은 Add-on이라는 패키지 시스템을 이용합니다. 이것은 기본 베이스 시스템에 추가로 필요한 기능들을 추가하는 의미입니다. 이러한 내용을 잘 모르는 상태에서 초보자가 가장 많이 겪는 실수는 어떤 함수를 사용하고자 할 때 "xxx 함수가 없습니다" 또는 "xxx 함수를 찾을 수 없습니다"입니다. 이는 사용하고자 하는 함수가 R 기본 배포판에 포함되어 있지 않은 어떤 사용자에 의해서 제공된 특정한 패키지내에서 존재하기 때문입니다. 이런 경우에는 먼저 사용하고자 하는 함수가 어떤

패키지에 존재하는지 알아야 합니다. 그리고, 해당 패키지를 설치했을 때에는 설치된 패키지를 사용할 수 있도록 로딩하는 과정을 거쳐야 합니다.

```
> library(pkg_name)
```

패키지의 설치, 확인에 관련된 사항은 "통계모형의 선택과 적용"이라는 챕터에 기록해두었습니다.

2.2 왜 R을 사용하나요?

R을 사용하는 이유는 아마도 아래와 같은 이유이기 때문입니다. (R Documentation에는 없는 이 문서의 지은이 개인의 생각임을 명심하시길 바랍니다)

1. R은 매우 다양한 분야에서 개발되고 적용되는 최신 통계기법을 적용할 수 있는 자유소프트웨어이기 때문입니다.
2. 행렬기반의 객체지향적 프로그래밍 언어이기 때문입니다.
3. 다른 소프트웨어들에 비교하여 문법적 사용의 자유롭기 때문일 것입니다.

2.3 통계소프트웨어의 종류

R이라는 통계소프트웨어를 대체할 수 있는 다른 소프트웨어들은 다음과 같습니다.

- S-PLUS (상업용 버전의 S 언어 소프트웨어)
- MATLAB (R과 같은 행렬기반의 언어)
- SPSS
- Octave (MATLAB의 GNU 버전)
- Python (프로그래밍 언어)
- SAS
- STATA

Chapter 3

사용환경에 익숙해지기

R은 연구활동의 수행을 위한 도구일 뿐입니다. 이 도구를 잘 활용한다는 것은 그만큼 사용환경에 익숙해져 있다는 의미와 같습니다. R을 처음에 접하는 사용자들은 본 챕터에서 단순히 R에 대한 사용환경을 어떻게 사용하는지에 대해서 익숙해지는데 촛점을 맞춰주시길 바랍니다.

3.1 시작전 "묻지마" 종류의 개념들

먼저, R에 대해서 왜 그렇게 작동하는가에 대한 질문보다는 단순히 R은 단순히 이렇게 사용해야 한다는 부분부터 설명합니다. 이곳에 적혀 있는 내용은 "자동차"를 왜 자동차라고 부르는가와 같은 질문과 같은 맥락으로 생각해주시길 바랍니다.

프롬프트와 대화식 사용: R을 열자마자 사용자의 입력을 기다리는 기호를 프롬프트라고 합니다.

```
>
```

이 프롬프트 기호뒤에 명령어를 작성합니다. 사용자의 명령이 끝났음을 알려주는 것이 엔터키입니다. R이 입력된 명령어를 처리하여 사용자에게 명령어의 수행 결과를 보여주는 것은 아래와 같습니다. 이는 항상 [1] 이라는 기호로 시작합니다. 실제로 [1] 이라는 것은 추후에 설명하겠지만 첫번째 객체를 의미하는 것입니다.

```
> 1
[1] 1
> 1+1
[1] 2
```

이렇게 사용하는 방식을 R 소트트웨어와 대화식으로 사용한다고 하며, 이러한 대화가 이루어지는 공간(즉, 사용자가 R에게 질의응답을 하는 곳)을 콘솔이라고 합니다.

작업디렉토리와 세션: 컴퓨터상에서 현재 작업하고 있는 디렉토리를 워킹디렉토리라고 합니다. 세션은 현재 R 콘솔상에서 작업하는 것을 의미합니다.

변수와 대입

```
x <- 7
7 -> x1
x = 7
```

스크립트 스크립트란 일련의 작업을 순서대로 나열하여 놓은 순서대로 실행되어지는 명령어의 집합이라고 할 수 있습니다. 대화형으로 R을 사용할 수 있으나, 보다 손 쉽게 작업을 할 수 있는 장점이 있습니다.

3.2 벡터 기반의 R에 익숙해지기

R이라는 새로운 도구에 익숙해지는 것은 손가락을 이용하여 자주 키보드를 눌러보는 것입니다. 먼저, 계산기로서 R은 어떤 기능을 가지고 있는지 살펴봅니다.

3.2.1 단순한 산술연산

먼저 R은 아래와 같은 사칙연산 (더하기, 빼기, 곱하기, 나누기)이 가능합니다.

$$1 + 2 - 3 + 4 * 5 - 6/3 \tag{3.1}$$

이를 R로 수행하기 위해서는 다음과 같이 입력합니다.

```
> 1 + 2 - 3 + 4*5 - 6/3
[1] 18
```

연산자라는 것은 어떤 특정 역할을 수행하는 기호이며, 산술연산에 대한 연산자 우선순위는 수학적 연산순서와 동일합니다. 다음과 같은 다양한 수학적인 연산이 가능합니다.

```
> # 제곱
> 3^2
[1] 9
>
> # 지수
> exp(3)
[1] 20.08554
>
> # 로그
> log(3)
[1] 1.098612
>
> # 파이상수 값
> pi
[1] 3.141593
>
> # 삼각함수 사인, 코사인, 탄젠트
> sin(0)
[1] 0
>
> cos(0)
[1] 1
>
> tan(45)
[1] 1.619775
>
> # 몫과 나머지 구하기
>
> 15/5
[1] 3
> 15/4
[1] 3.75
> 15%/%4
[1] 3
```

```
> 15 %%4
[1] 3
> 15 %% 3
[1] 0
> 15 %% 2
[1] 1
>
```

이러한 산술연산은 벡터를 기반으로 작동하게 됩니다. R의 가장 큰 특징은 벡터를 기반으로 작동한다는 것입니다.

```
> x <- c(1,2,3,4)
> x
[1] 1 2 3 4
> x^2
[1]  1  4  9 16
> x+3
[1] 4 5 6 7
> x/10
[1] 0.1 0.2 0.3 0.4
>
```

3.2.2 논리연산

```
< <= > >= == !=
```

3.2.3 집합연산

3.2.4 함수의 사용

위와 같은 사칙연산을 하던중 반올림을 해야 할 경우가 있을 것입니다. 이럴때 R에서 제공하는 함수(즉, 내장함수)를 이용합니다.

```
> x <- c(0.3823, 0.2353, 0.34321)
> x
[1] 0.38230 0.23530 0.34321
> round(x)
[1] 0 0 0
> round(x,2)
[1] 0.38 0.24 0.34
> round(x,3)
[1] 0.382 0.235 0.343
>
```

다음은 벡터를 생성하는 다양한 방법입니다.
시퀀스는 수열을 의미합니다. 수열이 곧 벡터입니다.

```
> seq(0, 1, 0.1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

만약 by 라는 인자를 넣지 않는다면 다음과 같습니다.

```
> seq(0, 10)
 [1]  0  1  2  3  4  5  6  7  8  9 10

> seq(1,10,length.out=4)
[1]  1  4  7 10
```

이러한 수열은 아래와 같은 방법으로도 생성이 가능합니다.

```
> 0:5
[1] 0 1 2 3 4 5
> 5:0
[1] 5 4 3 2 1 0
```

수열을 생성하는 또 다른 방법은 rep을 활용하는 것입니다.

```
> rep(1,5)
[1] 1 1 1 1 1
> rep(1:3, 3)
[1] 1 2 3 1 2 3 1 2 3
```

조금 더 응용해보면 아래와 같습니다.

```
> c(rep(0,1), rep(1,2), rep(2,3))
[1] 0 1 1 2 2 2
> rep(c("x", 0), 3)
[1] "x" "0" "x" "0" "x" "0"
>
```

3.3 행렬연산

행렬 연산에 대한 이해를 돕기 위해서 아래와 같은 간단한 행렬을 생각해 봅니다.

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \tag{3.2}$$

여기에서 행렬의 구성요소는 $1, 2, 3, 4$ 이며, 크기는 2행 2열입니다.

행렬의 생성: 위에서 수학적으로 표현된 A 라는 행렬을 R에 입력하기 위해서는 아래와 같이 합니다. 제일 처음의 숙지사항에서 언급된 바와 같이 입력된 행렬의 요소들을 R은 열방향으로 나열합니다.

```
> M <- matrix(c(1,2,3,4), ncol=2)
> M
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
```

그러나, 사용자는 때때로 행렬의 요소들을 행방향으로 나열하고자 할 수도 있습니다.

```
> M1 <- matrix(c(1,2,3,4), ncol=2, byrow=TRUE)
> M1
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
```

생성된 행렬의 크기를 확인하고자 한다면 dim()함수를 이용합니다. 첫번째 요소는 행의 개수이고, 두번째 요소는 열의 개수입니다.

```
> dim(M)
[1] 2 2
>
```

행과 열에 라벨링: 행렬의 행과 열에 각각 라벨링을 하고 싶다면 rownames()와 colnames()라는 함수를 이용합니다.

```
> rownames(M) <- c("R1", "R2")
> colnames(M) <- c("C1", "C2")
> M
   C1 C2
R1  1  3
R2  2  4
```

행렬의 전치: 2행 2열인 정방행렬을 열방향으로 나열하는 것을 행방향으로 나열하게 된다면, 이는 행렬의 전치를 의미하게 됩니다. 따라서, 행렬의 전치를 수행했을대 동일한 결과를 가지게 될 것입니다.

```
> t(M)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
```

행렬을 다시 벡터로 전환하기: 어떤 경우는 다시 행렬을 벡터의 형식으로 불러와야 할 경우도 있습니다.

```
> c(M)
[1] 1 2 3 4
>
```

역행렬 찾기: 역행렬은 solve()라는 함수를 통해 얻을 수 있는데, 이는 선형 연립방정식의 해를 구하는데 사용됩니다. $AX = b$라는 형식을 가지며, solve(M)의 결과는 x의 해를 의미합니다.

$$3x + 4y - 2z = 5 \tag{3.3}$$
$$-4x + 3y - z = 22 \tag{3.4}$$
$$x + y + z = 6 \tag{3.5}$$

와 같은 선형방정식을 손으로 풀면 $x = 1, y = 2, z = 3$ 이라는 값이 나오게 됩니다. 이를 R로 하기위해서는 다음과 같이 할 수 있습니다.

```
> solve(M)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
>
```

텐서 프로닥트 tensor product (혹은 Kronecker product)란 아래와 같이 행렬의 연산을 수행합니다. 이를 수행하기 위한 R 함수는 아래와 같습니다.

대각행렬 기능활용: 행렬 A의 대각행렬의 원소들은 1과 4인데, 이를 얻는 방법은 아래와 같이 diag() 함수를 사용하는 것입니다.

```
> diag(M)
[1] 1 4
```

만약, 대각행렬의 원소가 c(3,4)를 가지는 정방대각행렬을 생성하고자 한다면 아래와 같이 사용할 수 있습니다 .

```
> diag(c(3,4))
     [,1] [,2]
[1,]    3    0
[2,]    0    4
>
```

또한, I 행렬을 생성하는데 사용할 수 있습니다.

```
> diag(2)
     [,1] [,2]
[1,]    1    0
[2,]    0    1
>
```

고유값과 고유벡터: M 행렬의 고유값과 고유벡터는 아래와 같은 방법으로 구할 수 있습니다.

```
> eM <- eigen(M)
> names(eM)
[1] "values"  "vectors"
> eM$values
[1]  5.3722813 -0.3722813
> eM$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
>
```

행렬값 구하기

```
> M <- matrix(c(1,2,3,4), ncol=2)
> val <- det(M)
> M
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> val
[1] -2
> 1*4-3*2
[1] -2
```

특이값 분해 행렬 A의 Singular value decomposition (특이값 분해)는 아래와 같은 수학적 의미를 가집니다.

$$A = UD(Q)V^T \tag{3.6}$$

여기에서 $U^TU = V^TV = VV^T = I$ 이며 Q는 특이값 벡터입니다.

```
> svdM <- svd(M)
> names(svdM)
[1] "d" "u" "v"
> svdM$d
[1] 5.4649857 0.3659662
> svdM$u
           [,1]       [,2]
[1,] -0.5760484 -0.8174156
[2,] -0.8174156  0.5760484
> svdM$v
           [,1]       [,2]
[1,] -0.4045536  0.9145143
[2,] -0.9145143 -0.4045536
> $
```

행렬의 부분선택

```
output

colnames <- c("C1", "C2", "C3")
rownames <- c("R1", "R2", "R3")
dimnames(M) <- list(colnames, rownames)
M["C1",]
M[,"R2"]
```

행렬가지고 또 할 수 있는 기본적인 무엇이 있나요?

```
output

which(x==4)

y <- c(6,7,8,9,10)
cbind(x,y)
rbind(x,y)

typos <- c(11,12,13,14,15,16,17,18,19,20)
typos[c(1,3,5)]
typos[-c(1,5)]


x <- c(1,2,3)
names(x) <- c("A", "B", "C")
names(x) <- NULL
```

Chapter 4

기초 프로그래밍과 운영체제

우리는 독자가 R을 과학적 분석을 위해 필요한 일련의 프로세스를 진행하는 프로그래밍 언어로서 이해하기를 권장합니다. 이러한 프로세스를 수행함에 있어서 이 챕터에 있는 내용들을 먼저 숙지하시면 R을 사용하시는데 보다 수월함을 느끼시게 될 것입니다. 따라서, 이 챕터에서는 원리 중심으로 R이 어떻게 동작하는가에 촛점을 맞추고 이들을 잘 활용할 수 있는 예제를 넣어 이해하는데 중점을 둘 것입니다.

우리는 이러한 내용들을 설명하기 전에 에러와 경고에 대한 내용을 먼저 다룰 것입니다. 알려져 있는 많은 교재들이 프로그래밍의 문법적 오류를 잡아내는 디버거라는 도구의 설명에 중점을 두고 있으나, 이 문서는 문법적 오류가 아닌 논리적 오류를 찾는 방법을 중심으로 내용을 전개할 것입니다. 그 이유는 분석을 위한 프로그래밍은 논리적인 절차를 어떻게 잘 구성하는가에 따라서 그 효율성과 프로그램의 가독성이 달라지기 때문입니다.

4.1 에러에 관하여

output

4.1.1 에러의 종류와 관련 메시지

output

문법적 오류

output

논리적 오류

output

오류와 경고의 다른점

output

디버거를 사용하기전에

output

분석작업에서는 주로 논리적 오류를 찾는것이 중요하지만, 패키지 개발에서는 디버거를 잘 다루는 기술적 요소가 더욱 요구됩니다. 이러한 디버거의 사용에 대해서는 “패키지 제작”이라는 챕터에서 다루도록 하겠습니다.

4.1.2 에러 핸들링

만약, 메시지 역시 설명해줄려면 stop(), warning()도 함께 설명해주면 최고임.

try()

output

tryCatch()

```
result <- tryCatch(
{
        수행하고자 하는 표현식
},
warning = function(w) {
        위에서 수행한 표현식이 경고를 발생시킬때 어떻게 처리하고자 하는지에 대한 표현식
},
error = function(e) {
        위에서 수행한 표현식이 에러를 발생시킬때 어떻게 처리하고자 하는지에 대한 표현식
}, finally {
        위에서 수행한 표현식에 대한 최종적 처리를 위한 표현식
}
```

4.2 주석문의 사용

4.3 조건문:

repeat, while에 대해서 간단히 보여주고, break과 next 를 알려주면 좋음.

output

분기:

output

switch

4.4 함수의 정의와 사용

output

내장함수

output

사용자 정의 함수 함수를 정의한 뒤 사용할 때,

output

do.call()

output

스코프

인바이런먼트

output

4.5 벡터라이제이션과 반복문

output

반복문:

output

벡터라이제이션: apply(), lapply(), sapply(), mapply()의 사용방법

output

4.6 객체와 속성

속성:

output

객체:

output

4.7 제네릭 함수와 클래스

메소드:

output

제네릭 함수:

output

클래스:

output

4.8 스크립트 작성하기

일괄처리

source()
dump()
save()
load()

실행하기

```
output
```

4.9 운영체제와 소통하기

```
output
```

4.10 파일과 디렉토리 유틸리티

파일관리와 관계된 여러가지 유용한 유틸리티가 존재합니다.
`edit()`,

```
output
```

```
   file.edit()
```

```
output
```

```
   fix()
```

```
output
```

```
   file.show()
```

```
output
```

```
   file.path()
```

```
output
```

```
   list.files()
```

```
output
```

```
   dir.create()
```

```
output
```

```
   file.access()
```

```
output
```

```
   file.exists()
```

```
output
```

```
   file.copy()
```

```
output
```

```
   data.entry()
```

```
output
```

시간과 관련된 명령어들

```
output
```

자주 찾는 질문들

```
rm(list=ls())
search()
install.packages()
library(MASS)
data()
```

4.11 프로그래밍 스타일 가이드

output

Chapter 5

데이터 조작과 관련하여

5.1 데이터 파일 입출력

데이터 입력과 출력은 R을 이용한 분석에서의 첫번째 단계와 마지막 단계라고도 할 수 있습니다. 데이터가 입력된 형식은 매우 다양하지만, 일반적으로 R을 이용하여 데이터의 입출력을 하는데 있어 안전한 방법은 .csv이라는 파일확장자를 가진 파일을 이용하는 것입니다. 따라서, 가급적이면 다른종류의 파일확장자를 .csv로 먼저 변경한 뒤에 사용하는 것이 좋습니다.

5.1.1 입력

```
output
```

다른 형식들:

```
output
```

고정형식

```
output
```

복잡한 구조를 읽어올때

```
output
```

.SAS

```
output
```

.SPSS

```
output
```

.URL

```
output
```

XML

```
output
```

.xls 또는 .xlsx 마이크로소프트 엑셀을 사용합니다.

```
output
```

.csv

```
output
```

`read.table()`이라는 함수를 아래와 같은 방법으로 이용하여 데이터를 읽어옵니다.

```
mydata <- read.table(file="./filename.csv", header=TRUE, sep=",")
```

여기에서 filename.csv 은 파일명입니다.

입력과 관련된 문제해결법 `read.table()` 함수를 이용하여 데이터를 불러오는데 있어서 많이 발생되는 오류는 "데이터 파일을 작업디렉토리로부터 찾을 수 없다" 또는 "데이터 파일이 존재한 파일경로가 올바르지 않다" 라는 것입니다.

파일을 입력받을 때 R은 일반적으로 첫번째 인자에 주어진 파일명과 현재 작업디렉토리의 파일경로를 함께 묶어 절대경로를 생성한 뒤, 이 절대경로를 이용하여 파일명을 찾습니다. 이러한 원리때문에 운영체제가 영어가 아닌 컴퓨터의 경우, 이 절대경로를 올바르게 생성하지 못할 경우가 있습니다. 또한, 파일경로명에 띄어쓰기가 있는 경우 및 특수문자가 포함된 경우에 이러한 문제가 발생할 경우가 있습니다. 따라서, 사용자는 간혹 문법에서 틀린 점도 없고, 불러오고자 하는 데이터 파일도 올바른 파일경로에 위치하고 있음에도 불구하고, 데이터를 찾을 수 없다는 에러 메시지를 보게 되는 경우가 있습니다. 이러한 경우에 보다 안전한 방법으로 `read.table()` 사용하고자 한다면 아래와 같이 `file.choose()` 함수 또는 `file.path()` 함수를 이용하시길 바랍니다.

```
mydata <- read.table(file.choose(), header=TRUE, sep=",")
```

`file.choose()`는 탐색기를 띄워 사용자가 원하고자 하는 파일을 찾을 수 있도록 도와줍니다.

```
mydata <- read.table(file.path(), header=TRUE, sep=",")
mydata1 <- read.table(file=url(site_address), header=TRUE, sep=",")
```

`file.path()`는 절대경로를 보다 안전하게 R이 이해할 수 있도록 도와줍니다.

```
age <- scan()
32 33 39 28 20 20
```

5.1.2 출력

저장하기

```
output
```

.RData

```
output
```

.CSV

```
output
```

.HTML

```
output
```

.XML

```
write(t(mydata), file="./where/should/be/saved", ncolumns)
```

5.1.3 메타데이터 처리

원 데이터 소스에 데이터 구조 대한 이해와 데이터 엔트리:

```
output
```

데이터셋 또는 변수에 주석첨가하기

```
output
```

5.2 데이터형에 대한 이해

요소 ∈ 벡터 ∈ 행렬 ∈ 배열

5.2.1 벡터

```
output
```

5.2.2 요인과 수준

```
output
```

5.2.3 행렬

```
output
```

5.2.4 데이터프레임

```
output
```

5.2.5 리스트

리스트와 데이터 프레임 관계:

```
ls()

names(mydata)

str(mydata)


dim(object)

class(obj)

mydata

head(mydata, n=10)
```

```
tail(mydata, n=5)

length(obj)
str(obj)
class(obj)
names(obj)

c(obj,obj,...)
cbind(obj, obj, ...)
rbind(obj, obj, ...)

obj

ls()
rm(obj)

newobject <- edit(obj)
fix(obj)
```

5.2.6 배열

output

배열과 행렬과 벡터와의 관계

```
cube <- array(1:27, dim=c(3,3,3))
cube[1,,]
cube[,1,]
cube[,,1]
```

5.2.7 결측치

NA 와 NaN을 데이터로부터 찾고 싶어요. `is.na()`와 `is.nan()` 함수 사용법을 알려주면 좋음. 추가로 `is.null()`도 알려주면 is관련 함수들에 설명해주면 짱임.

```
x <- c(1,3,NA,7,9)
is.na(x)
y <- x[!is.na(x)]

x <- c(3,9,8,2,3,9,1,4,5)
trt <- c(rep("A", 3), rep("B", 3), rep("C", 3))
x[trt=="A"]
x[trt=="A"|trt=="B"]
split(x, trt)
```

5.3 데이터 클리닝 및 전처리 테크닉

output

5.3.1 데이터셋에 관련하여

output

변수명 변경하기

output

조건에 부합하는 데이터셋 골라내기

output

주어진 데이터셋으로부터 랜덤샘플 추출하기

output

정렬하기

output

데이터셋 합치기

output

변수 추가 또는 제거

output

종횡데이터를 횡형으로 변형하기

output

횡형데이터를 종형으로 변형하기

output

관측치의 개수 알아보기

output

중복되는 값 찾아보기

output

결측치에 대해서

output

수치형으로 변환

```
as.numeric()
as.numeric(as.character(variable))
as.numeric(gsub(",", "", variable))
```

5.3.2 문자형 변수들과 연관하여

output

수치형 변수로 강제형변환 하기

output

빈공간 모두 제거하기

output

특정 문자열 뽑아내기

output

변수의 길이 파악하기

output

두 문자형 변수 결합하기

output

대소문자 전환

output

요인과 관계하여

output

라벨링 생성 및 변경하기

output

5.3.3 시간과 날짜에 관련하여

날짜 데이터 생성

output

년/월/일 따로 분리하기

output

시간 데이터 생성

output

5.4 연산자

논리연산자

```
<, >, <=, >=, ==
with(data.frame, expression)
with(mydata, age>34)
attach()
```

5.5 유용한 클리닝 테크닉들

분석자가 보통 얻게 되는 데이터는 분석에 사용되는 통계모형에 적합한 경우는 드물기 때문에 분석자 스스로가 이러한 데이터를 형성하는 것은 필요한 기술중에 하나라고 할 수 있습니다.

결측치를 바로 윗값으로 채워넣기: 아래와 같이 주어진 데이터에 변수 ID는 결측값 없이 모든 값이 완전하게 잘 들어가 있는데, Week 변수에는 각 ID의 첫번째 레코드에만 해당하는 부분에 값이 들어가 있고 나머지부분에는 NA값이 들어가 있습니다.

```
mydata <- data.frame(ID=c(rep(1,4), rep(2,4), rep(3,2)), Week=c(15, NA, NA, NA, 18, NA, NA, NA, 20, NA))
```

```
> mydata

   ID Week
1   1   15
2   1   NA
3   1   NA
4   1   NA
5   2   18
6   2   NA
7   2   NA
8   2   NA
9   3   20
10  3   NA
```

이와 같은 데이터를 아래와 같이 자동으로 채워주려면 어떻게 해야 할까요?

```
   ID Week
1   1   15
2   1   15
3   1   15
4   1   15
5   2   18
6   2   18
7   2   18
8   2   18
9   3   20
10  3   20
```

이를 수행하는데에는 여러 가지 종류의 함수들이 다양한 패지키 안에 존재합니다. 그러나, 이를 수행하는 기본 알고리즘은 동일하며, R 기본시스템만으로 작성이 가능합니다. 아래의 함수를 복사하여 사용하시면 됩니다.

```
fill <- function(x, first, last){
        n <- last-first+1
        for(i in c(1:length(first))) x[first[i]:last[i]] <- rep(x[first[i]], n[i])
        return(x)
}
```

각 아디이별로 첫번째와 마지막 레코드 찾아보기: 위에서 주어진 데이터에서 ID 변수에서 보이는 것처럼 같은 관측치가 여러번 반복 측정되어 ID가 반복적으로 입력이 되었을 때, SAS에서처럼 각 아이디별로 첫번째와 마지막 레코드를 알수 있는 .FIRST 와 .LAST 같은 기능이 R에서는 어떻게 해야 하나요?

```
mydata$first <- !duplicated(mydata$ID)
mydata$last <- !duplicated(mydata$ID, fromLast=TRUE)
```

```
> mydata
   ID Week first  last
1   1   15  TRUE FALSE
2   1   NA FALSE FALSE
3   1   NA FALSE FALSE
4   1   NA FALSE  TRUE
5   2   18  TRUE FALSE
6   2   NA FALSE FALSE
7   2   NA FALSE FALSE
8   2   NA FALSE  TRUE
9   3   20  TRUE FALSE
10  3   NA FALSE  TRUE
```

조건에 맞게 데이터 선택하기: 데이터의 일부분만 골라 내고 싶어요. 예를들면, 위에서 사용된 예제에서 ID 가 1과 2인 데이터만 골라내고 싶다면 아래와 같이 할 수 있습니다.

```
# 데이터 생성하기
mydata <- data.frame(ID=c(rep(1,4), rep(2,4), rep(3,2)), Week=c(15, NA, NA, NA, 18, NA, NA, NA, 20, NA))

# ID 변수에 있는 ID를 기준으로 첫번째와 마지막 레코드의 위치 알아내기
idx.first <- which(!duplicated(mydata$ID))
idx.last <- which(!duplicated(mydata$ID, fromLast=TRUE))

# ID에 있는 NA값을 채워넣기
mydata$Week <- fill(x=mydata$Week, first=idx.first, last=idx.last)

# 개별 ID에 대한 첫번째와 마지막 레코드에 대한 논리값을 추가하여 데이터 확장하기
mydata$first <- !duplicated(mydata$ID)
mydata$last <- !duplicated(mydata$ID, fromLast=TRUE)

# 조건에 맞는 데이터 골라내기
select <- subset(x=mydata, subset=(ID %in% c(1,2)))
> select
  ID Week first  last
1  1   15  TRUE FALSE
2  1   15 FALSE FALSE
3  1   15 FALSE FALSE
4  1   15 FALSE  TRUE
5  2   18  TRUE FALSE
```

```
6  2   18 FALSE FALSE
7  2   18 FALSE FALSE
8  2   18 FALSE  TRUE

# 추 가적인 조 건 부 여 하 기
select.1 <- subset(x=mydata, subset=( (ID %in% c(1,2)) & first==TRUE ))
> select.1
  ID Week first  last
1  1   15  TRUE FALSE
5  2   18  TRUE FALSE
```

여러개의 엑셀시트로 구성된 엑셀파일 하나로 합치기: 여러개의 엑셀시트로 구성되어 있는 엑셀파일을 불러와 하나의 데이터셋으로 합치기

```
output
```

리스트 중첩구조 가끔 리스트형으로 받아진 데이터가 중첩된 구조를 가지고 있어서, 한 번에 이를 불러오기를 해야할 때는 어떻게 해야할지.

```
output
```

Chapter 6

수학/확률/행렬/수치해석과 관련하여

6.1 수학함수들의 사용

일반 수학함수들

삼각함수들

output

집합과 관련된 함수들

output

기타 유용한 함수들 combn() 함수를 이용하여 모든 조합을 찾기

6.2 확률의 사용

6.2.1 밀도/누적 확률분포

output

퍼센타일 값 찾기

output

6.2.2 표준 난수생성 함수

output

6.2.3 비표준 난수생성 알고리즘

output

Multinomial random variables

output

Correlated binary random variables

output

6.3 수치해석

output

6.3.1 미분

output

6.3.2 적분

output

Laplace Approximation 알고리즘을 구현하는 방법 – 적분하는 방법에 많이 쓰임 (특히, 베이지안 컴퓨테이션)

output

6.3.3 최적화 문제

output

Newton-Raphson 알고리즘을 구현하는 방법 – optimization 에 관련된 일종의 설명도 추가해주면 좋을 것 같음

output

6.4 시뮬레이션

6.4.1 Metropolis-Hastings

알고리즘을 구현하는 프레임 워크 - 이것은 그냥 사용가능하게 바로 소스코드 붙여주기 (베이지안 컴퓨테이션에 많이 쓰임)

output

6.4.2 Bootstrap

방법 – 요건 아주 좋은 패키지가 있음

output

Chapter 7

탐색적 데이터 분석

여기에서 말하는 탐색적 분석이란 분석 초기에 단순히 데이터의 특징을 보는데 사용됩니다. 또한, 이 과정은 데이터 클리닝과도 연관이 있습니다.

7.1 기술통계량 요약

```
output
```

평균과 분산과 같은 기초요약 함수들

```
output
```

그룹별 평균산출

```
output
```

5분위수 구하기

```
output
```

퀀타일

```
output
```

표준화와 스케일링

```
output
```

신뢰구간

```
output
```

7.2 분할표와 카이제곱 검정

```
output
```

7.3 z-검정

```
output
```

7.4 t-검정

```
output
```

7.5

```
output
```

Part II

R과 통계모형을 위한 분석

Chapter 8

통계모형의 선택 및 적용

각 섹션은 다음과 같은 방법으로 이루어져야 합니다.

- 아래의 모형이 어느 경우에 사용되어야 하는가?
- 모형을 사용하는데 있어서 요구되는 가정들은 무엇인가?
- 모형의 계수에 대한 추정치는 어떻게 구하는가?
- 모형의 진단
- 추정치들에 대한 해석
- 사용법들
- 모형에 대한 한계점

8.1 분산분석 (Analysis of Variance-Covariance)

output

8.2 상관분석 (Correlation Analysis)

output

8.3 회귀분석 (Regression Analysis)

output

8.4 주성분분석 (Principle Component Analysis)

output

8.5 판별분석 (Discriminant Analysis)

output

8.6 군집분석 (Cluster Analysis)

output

8.7 시계열분석 (Time-Series Analysis)

output

8.8 일반선형모델 (GLM)

output

8.9 의사결정 나무(Decision Tree)

output

8.10 Longitudinal data analysis

output

8.11 생존분석 (Survivial analysis)

output

8.12 Mixture and latent class analysis

output

8.13 신경망 분석

output

8.14 기계학습 (Machine Learning)

output

8.15 메타 분석 (Meta Analysis)

output

8.16 패키지 관리

1. 이와 반대로 현재 연결된 라이브러리를 떼어낼 수도 있습니다.

```
> detach(package:pkg_name)
```

2. 패키지를 설치 (분류: 사용자 환경)
   (답변) 설치되는 패키지의 **설치위치**와 **의존성**에 대해서 반드시 알아야 합니다.

```
> install.packages("패키지명", dependencies=TRUE, )
```

3. 설치된 패키지의 목록을 확인하는 방법을 알고 싶습니다.

Part III

그래픽스

Chapter 9

비쥬얼라이제이션

9.1 플랏 커스터마이징

output

포인트와 텍스트 사이즈 변경

output

마진 조절하기

output

한페이지에 다중 그래프 생성

output

축 라벨, 보조선 조절하기

output

선의 종류와 너비조절

output

색상에 대해서

output

9.2 그래픽 요소 추가하기

output

직선 넣기

output

플랏 기호 변경

output

다중 플랏 보여주기

output

제목/부제목 수정

output

라벨 붙이기

output

레전드 넣기

output

수학기호 표현하기

output

텍스트 집어넣기

output

9.3 그래픽 출력하기

output

Postscript

output

PDF

output

JPEG, TIFF, PNG

output

9.4 플랏팅의 종류

output

9.4.1 산점도

output

9.4.2 바플랏

output

다중 바플랏

output

9.4.3 히스토그램

output

9.4.4 줄기-잎 그림

output

9.4.5 Q-Q 플랏

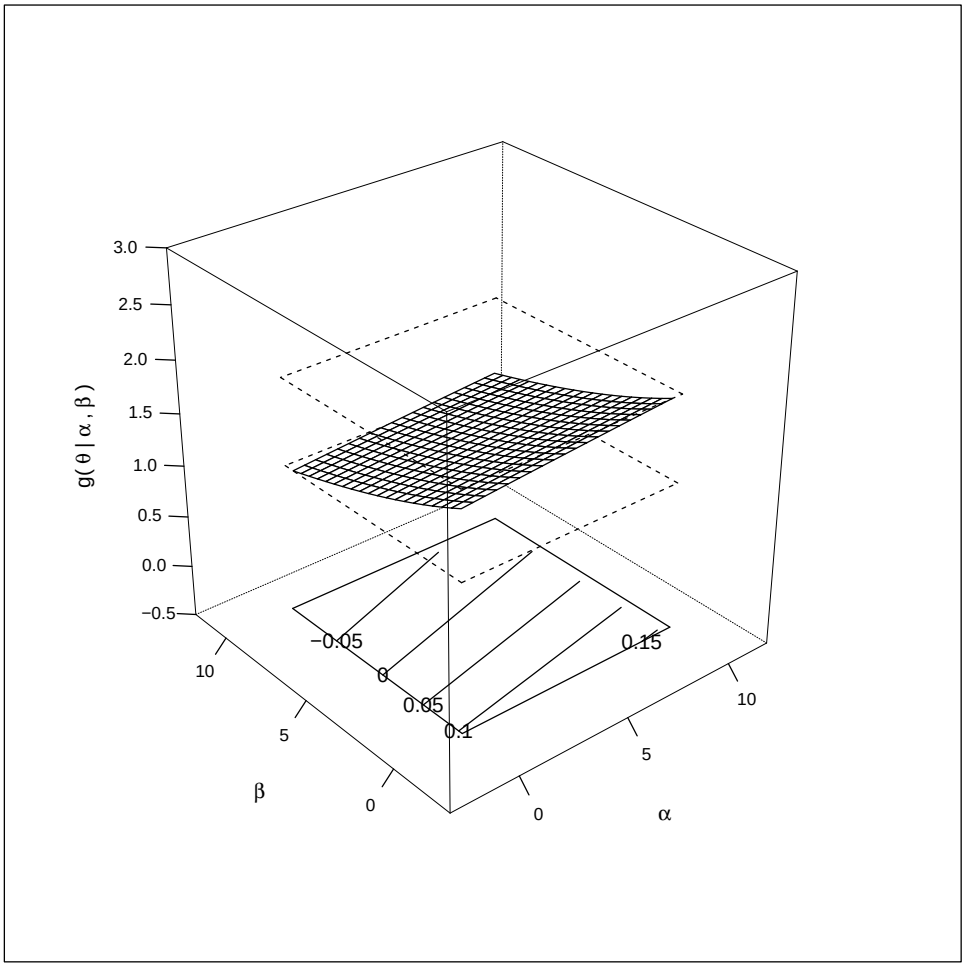
output

9.5 3차원 플랏

output

1. coordinating system을 활용하기

2. Lattice 패키지를 이용하여 아래와 같은 그림을 생성해보기 (가장 단순한 예제임 - 팁 보다는 튜토리얼 형식으로?)



Lattice Example 1

3. LaTeX의 문서에 포함될 `.eps` 그래픽을 R에서 뽑았을 때는 아무런 문제가 없어 보였는데, 정작 `pdf`로 문서를 뽑아 보니까 이 그래픽이 들어간 페이지가 90도로 돌아가 있거나 혹은 그래픽이 90도로 회전되어 있을 경우에는 아래와 같이 하면 됩니다.

```
postscript(file=``filename.eps'', onefile=FALSE, horizontal=FALSE)
```

   이 문제에 대한 출처는 `postscript` 도움말입니다.

   이문제를 다른 방법으로도 해결할 수 있습니다. (대충 서너개 더 있음).

4. 새로운 그래픽 객체를 생성하는 방법을 설명해줘서 사용자가 추후에 독립적인 그래픽을 생성할 수 있게 도와주기

5. (접수: 2013-APR-23) 저는 다각형을 그리고 싶습니다.
   (답변) 이것은 간단히 2차원-랜덤포인트 생성한 뒤에 `polygon()` 함수를 써서 보여주면 됨.

Part IV

분석 후의 패키지 개발

Chapter 10

분석 후 개발과 관련하여

10.1 클래스와 메소드 그리고 패키지 제작

1. 패키지를 만들고 싶어요. (흠.. Generalized Linear Model 프레임워크 흉내내서 똑같이 만들어보기 실습자료로 제공해주기)

output

10.2 간단한 GUI 제작 해보기

1. 다른 언어로 인터페이싱 하는 방법마로, 그냥 R에서 주어지는 패키지를 이용해서 간단한 GUI 환경만들기
2. 아마도... R Commander를 확장하는 방법을 예로 들면 좋을 것 같음
3. 원리도 간단히 설명해주면 더욱 좋을 것 같음.

output

Chapter 11

미분류 질문들

이 섹션에 등록된 질문들은 접수만 되고 아직은 답변되지 않은 상태입니다

1. (접수: 2013-APR-23) 제가 가진 데이터셋이 있는데, 이 데이터를 어떤 특정한 변수들의 값을 이용하여 분류하려고 합니다. 어떻게 해야하나요?

   (답변) 요것은 `split()` 함수를 이용하도록 알려줄 것.

2. (접수: 2013-APR-23) 제가 가진 데이터 프레임에 NA 값들이 있는데, NA 때문에 분석이 이상해지는 것 같아, NA를 가진 데이터 행자체를 없애고 싶습니다. 한번에 해주는게 없나요?

   (답변) 요건 `na.omit()`과 같은 함수를 이용하는 법을 알려줄 것. 흠.. `na.action`이라는 개념을 알려주면 더욱 좋음.

3. (접수: 2013-APR-23) 논리형 벡터가 있는데, 이 벡터의 구성요소가 모두 TRUE 인지 알고 싶습니다.

   (답변) 이건 `isTRUE()`함수와 `all()`함수를 통해 알려주면 매우 좋음.

4. (접수: 2013-APR-22) 선형방정식 $AX = b$의 해 X를 찾으려면 어떻게 해야 하나요?

   (답변) `solve()` 함수의 사용법을 알려줄 것.

5. (접수: 2013-APR-21) R 패키지를 CRAN에 올리는 방법을 알려주세요

   (답변) 이 질문을 대답할 때는 반드시 CRAN Package Submission Guideline에 대해서 알려줘야 함. (이거 번역해 놨는데 당췌 어디에 뒀는지 찾을 수가 없음, 2013-04-20 까지 못 찾으면 새로이 번역할 것)

6. (접수: 2013-APR-19) R은 처음부터 기존의 통계팩키지와는 다른 모습에 약간 두렵기까지 합니다. 기존의 분석은 일반적으로 [프로그램 실행 $->$ 데이터 불러오기 $->$ 분석(메뉴클릭:SPSS 또는 명령어입력:SAS) $->$ 실행]의 절차를 밟아 왔기에 모든 결과를 한 번에 보여주는 식입니다. 그러나 R은 그렇지 않아 이러한 점부터 생소하고 이상합니다. 데이터를 불러오기 하면 바로 데이터시트를 볼 수 있는 것도 아닙니다 (접수날짜: 2013-APR-17).

7. (접수: 2013-04-18, Reproducibility=NO) read.xlsx함수를 이용해 xlsx파일에서 데이터프레임형태로 가져옵니다. 이 때 [3,3] 셀에 있는 텍스트가 "3월" 이라고 할 때 temp[3,3] == "3월" 이렇게 비교하려고 하면 제대로 비교가 안되더군요.. 한글 텍스트로 이루어진 변수값를 비교하는 방법이 어떤게 있는지 궁금합니다.

8. 분석을 하고 나면 결과를 그래프나 그림으로 나타내게 되는데 R에서는 그림을 나타내는 창이 하나만 나타나서 동시에 두 개를 보지 못하는 경우가 허다한데, 이의 해결방법은 없나요? (접수: 2013-APR-13, 분류: 그래픽스 관련)

   (답변) R에서는 그래픽 디바이스가 그래픽 생성시 마다 초기화되어 다시 보여줌으로서 그래픽 창이 하나만 계속 보여지는 것입니다. 새로운 그래프를 또다른 장치를 통해 보여주고자 한다면 `X11()`이라는 명령어를 이용하면 됩니다. 이 명령어는 유닉스환경에 설치된 R의 경우에 해당합니다.

11.1 답변되지 않을 수도 있는 질문들

1. (접수: 2013-04-18, Reproducibility=NA) R의 장점이자 단점이라고 생각되는 것 중에 하나가 엄청난 수의 패키지들임. 즉 어떤 분석을 하고자 할 때 그것에 대해 하나의 패키지가 있는 것이 아니라 대체적으로 사용가능한 패키지들이 존재하는데 이들 중 어느 것을 써야할 지 잘 모름. 다른 분석 프로그램의 경우 이러한 문제가 없는데... 결국엔 어떻게 제일 성능이 좋은? 결과가 신뢰할 만한? 좋은 패키지를 선택하는가를 알려주었으면 좋겠씀돠.

   (답변) 이것은 경험에 해당되며, 해당분야의 전문가로부터의 조언을 받는 것이 안전합니다. 그렇지 않다면, 직접 베이스를 이용하여 작성하면 됩니다.

Part V

알면 도움이 되는

Chapter 12

유닉스 명령어

Chapter 13

LaTeX사용법

Chapter 14

Perl 명령어

Chapter 15

C 언어

Bibliography

Hornik, K. (2013). The R FAQ.

R Core Team (2012). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.