국립환경과학원

경희대학교
KYUNG HEE UNIVERSITY

R을 이용한 자료기반 모델 실습

이대성, 박영석
경희대학교

목차

1. R 입문
2. Random forest model
3. MaxEnt model

R 입문

이대성, 박영석
경희대학교 생물학과

1. R이란?

- **R (R programming language)**은 데이터 분석을 위한 통계 계산 및 그래픽을 위한 프로그래밍 언어입니다

- R은 벨 연구소에서 만들어진 통계 분석 언어인 S를 바탕으로, 뉴질랜드 오클랜드 대학의 로버트 젠틀맨(Robert Gentleman)과 로스 이하카(Ross Ihaka)에 의해 제작되었고, 현재는 R 코어팀(R Development Core Team)에서 유지 및 보수, 개발을 진행하고 있습니다.

- R은 최근 데이터 분석을 위한 도구로 각광받고 있으며, 전기전자 및 최신 기술로 유명한 IEEE Spectrum (https://spectrum.ieee.org)에서 선정한 The Top Programming Languages 2019에서 5위를 차지하였습니다. (2017년 6위 → 2019년 5위)

Rank	Language	Type	Score
1	Python		100.0
2	Java		96.3
3	C		94.4
4	C++		87.5
5	R		81.5
6	JavaScript		79.4
7	C#		74.5
8	Matlab		70.6
9	Swift		69.1
10	Go		68.0

※ Rank of Programming Languages (→)
https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019

1. R이란?

○ R의 장점:

1. R은 Interpreter language입니다.

→ 소스 프로그램을 한번에 기계어로 변환시키는 컴파일러와는 달리 프로그램을 한 단계씩 기계어로 해석하여 실행하는 '언어처리 프로그램'이다. 즉, 순차 번역 및 실행 형식의 처리 방법을 가지므로, 사용자가 실시간으로 처리과정을 확인 할 수 있습니다. ※ 기존 프로그래밍 언어(C, Phython 등)와의 차이점

2. R은 무료이고, 오픈 소스 프로그램입니다.

→ SPSS와 같은 기존의 다른 통계 툴들은 비싸기도 하고, 개발사에서 정한 기능 이상을 원할 수 없었습니다.
하지만 R은 소프트웨어의 설계도에 해당하는 소스코드가 무료로 공개되어 있기 때문에 전세계 누구나 R의 기능을 향상시킬 수 있습니다.

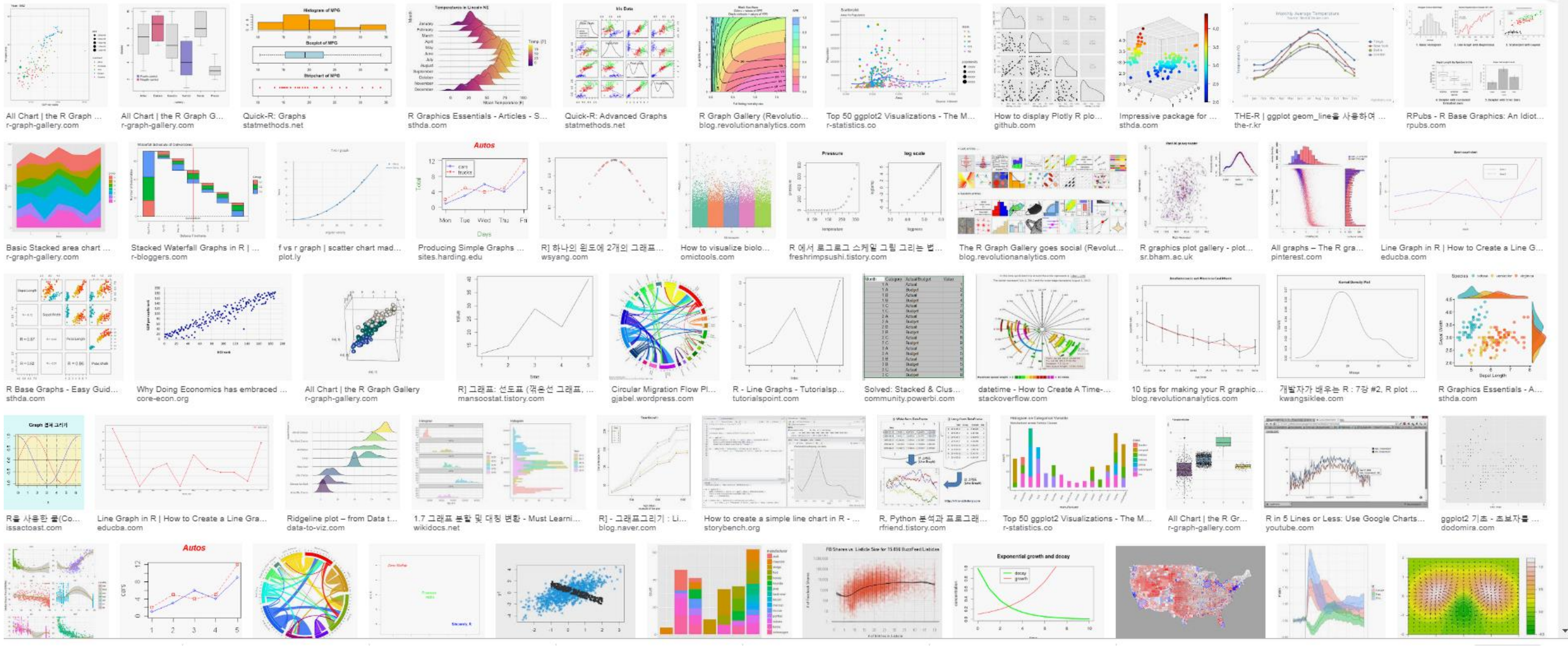
3. 다양한 통계 기법과 수치 해석 기법을 제공합니다.

→ R은 사용자가 제작한 패키지(Package)를 추가하여 기능을 확장할 수 있습니다. 핵심적인 패키지는 R와 함께 설치되며, CRAN(the Comprehensive R Archive Network; https://cran.r-project.org/)을 통해 2019년 현재 15,365 개 이상의 패키지를 사용할 수 있습니다.

4. 뛰어나고 다양한 그래픽 기능을 제공합니다.

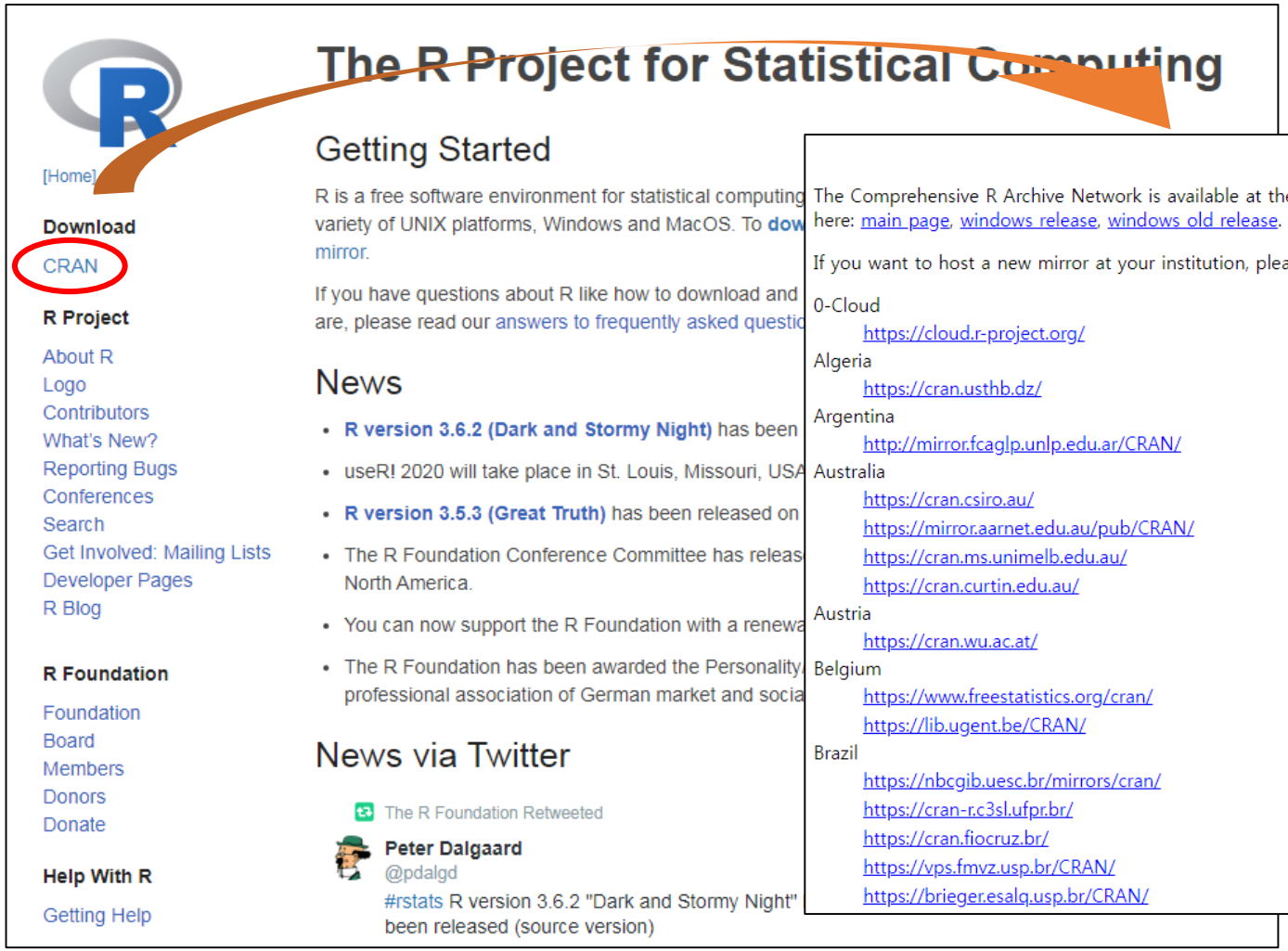
1. R이란?

※ R graphics (Keywords : R graph)

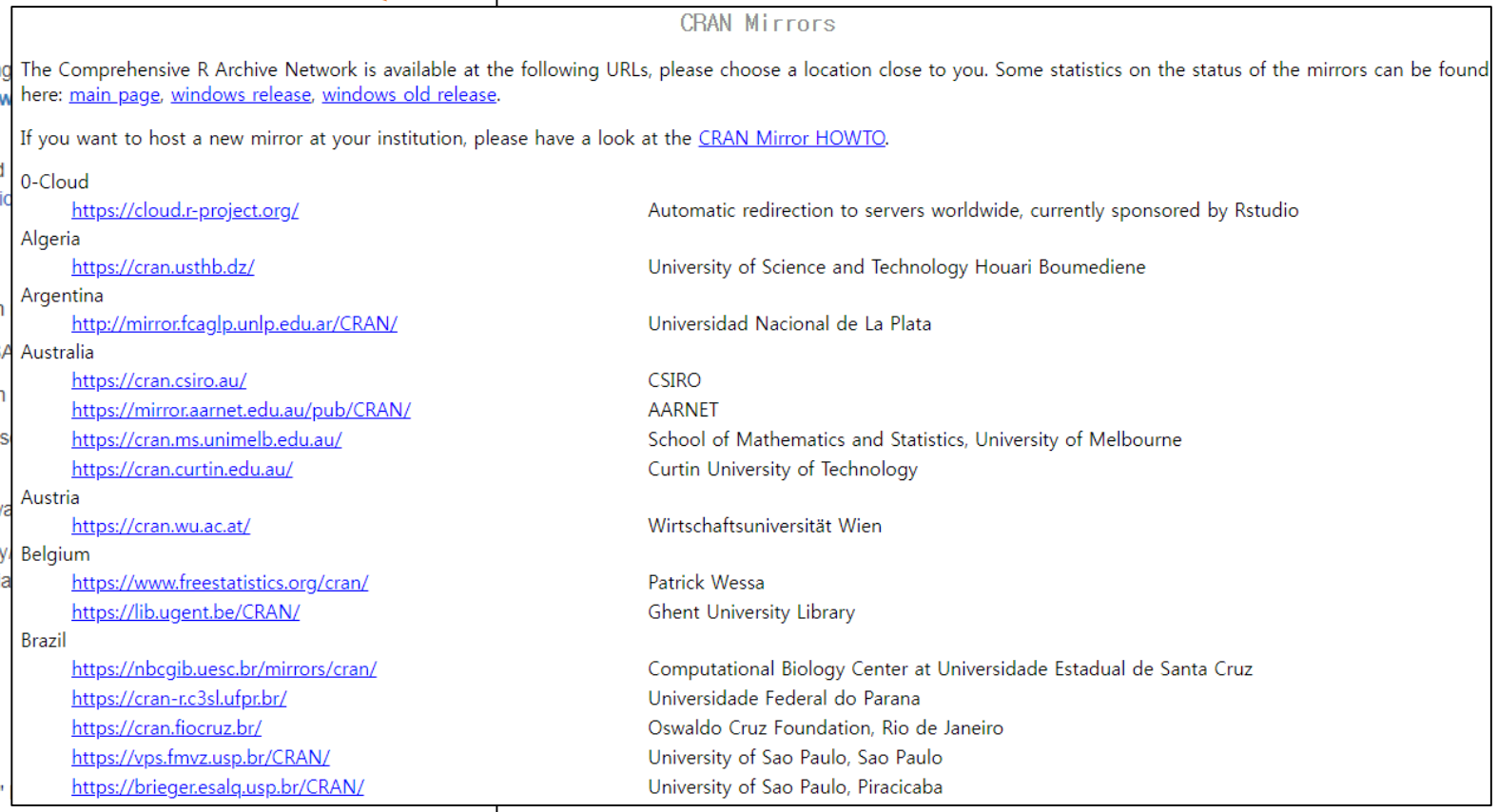


2. R설치

○ R 설치 (https://www.r-project.org/)



※ 국가 서버 선택(Korea)



2. R설치

○ R 설치 (https://www.r-project.org/)

Korea

https://ftp.harukasan.org/CRAN/ — Information and Database Systems Laboratory, Pukyong National University

https://cran.yu.ac.kr/ — Yeungnam University

https://cran.seoul.go.kr/

http://healthstat.snu.ac.kr/CRAN/

https://cran.biodisk.org/

The Comprehensive R Archive Network

Download and install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

※ 운영 체제 선택

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-12-12, Dark and Stormy Night) R-3.6.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

2. R설치

○ R 설치 (https://www.r-project.org/)

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

2. R설치

○ R 설치 (https://www.r-project.org/)

R-3.6.2 for Windows (32/64 bit)

Download R 3.6.2 for Windows (83 megabytes, 32/64 bit) ※ 최신 버전 다운로드

Installation and other instructions

New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

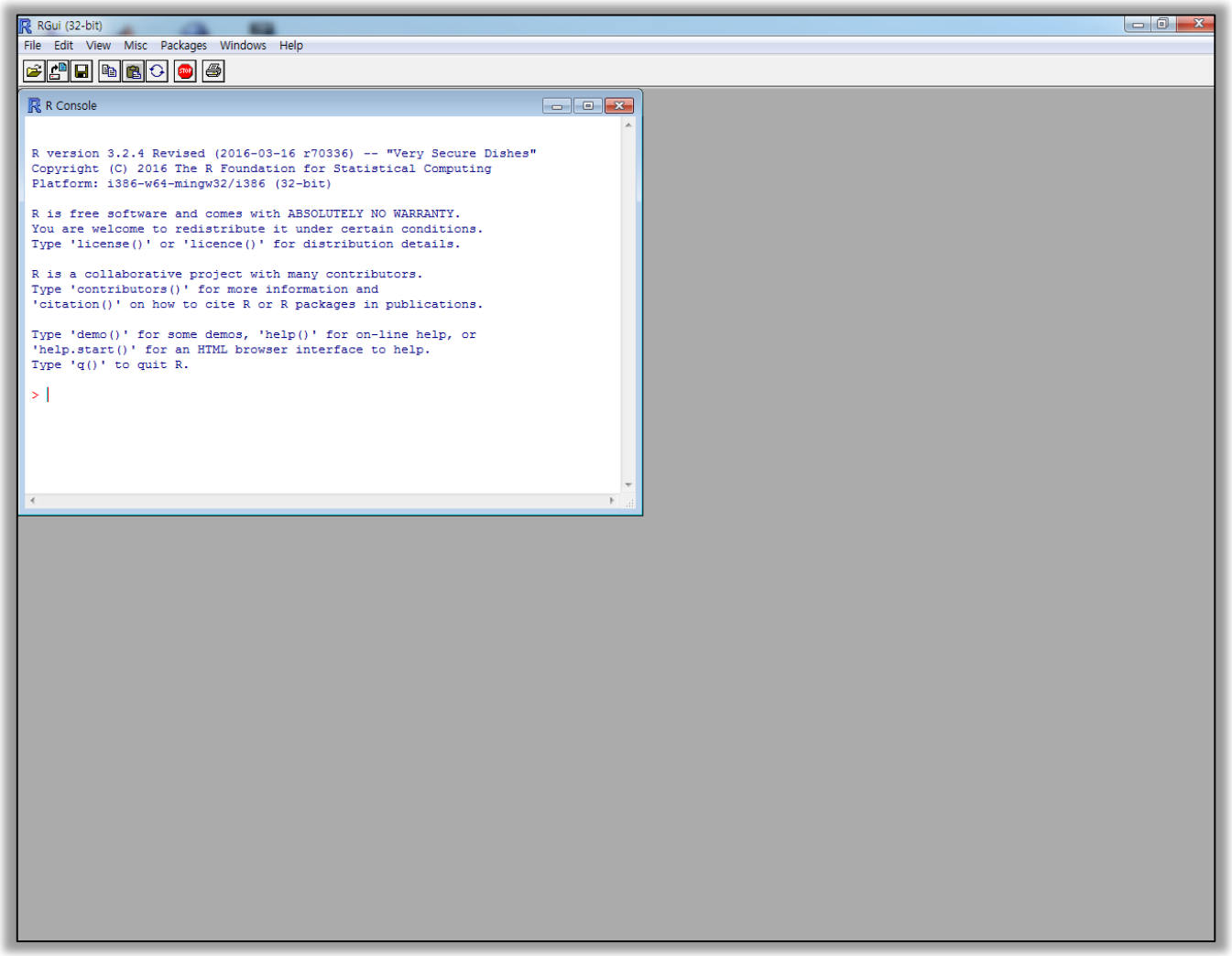
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.htm.

Last change: 2019-12-12

2. R설치

○ R 설치 (https://www.r-project.org/)



2. R설치 + R지원 프로그램

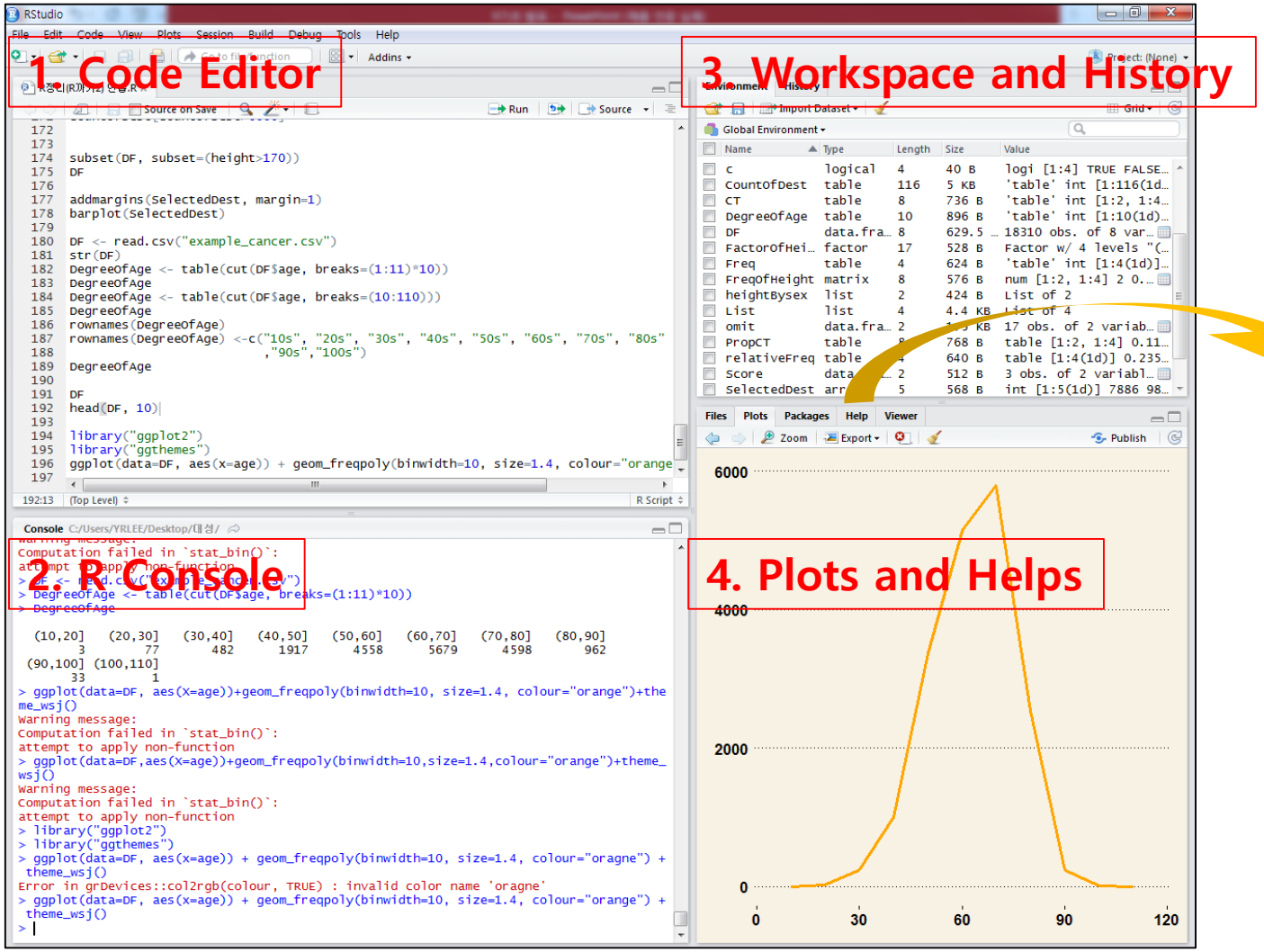
○ **R Studio (Using R through R Studio)**

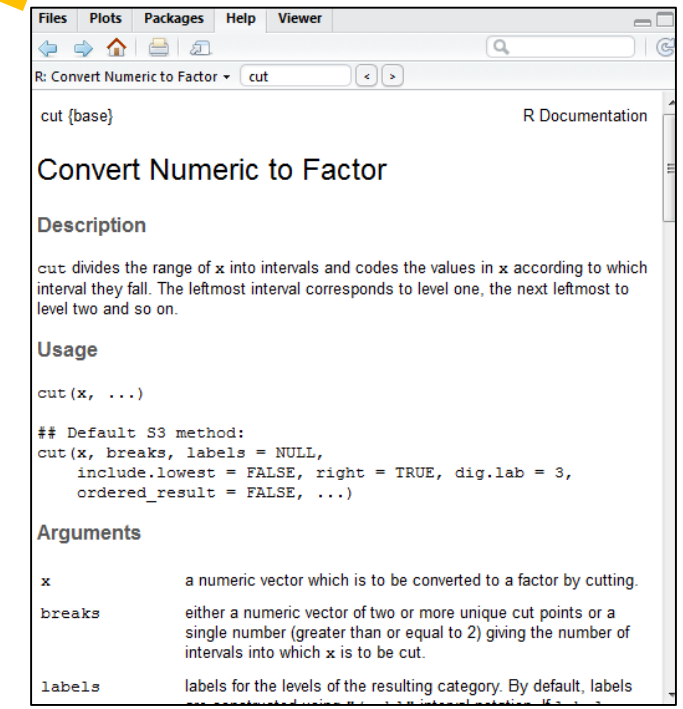
- **R Studio**는 통계 컴퓨팅, 그래픽스를 위한 프로그래밍 언어인 R을 위한
무료이며, 오픈 소스인 통합 개발 환경(integrated development environment)입니다.

※ Rstudio를 사용할려면 R이 먼저 설치되어있어야 합니다.
R studio자체에 R이 내장되어 있지 않습니다.

https://www.rstudio.com/

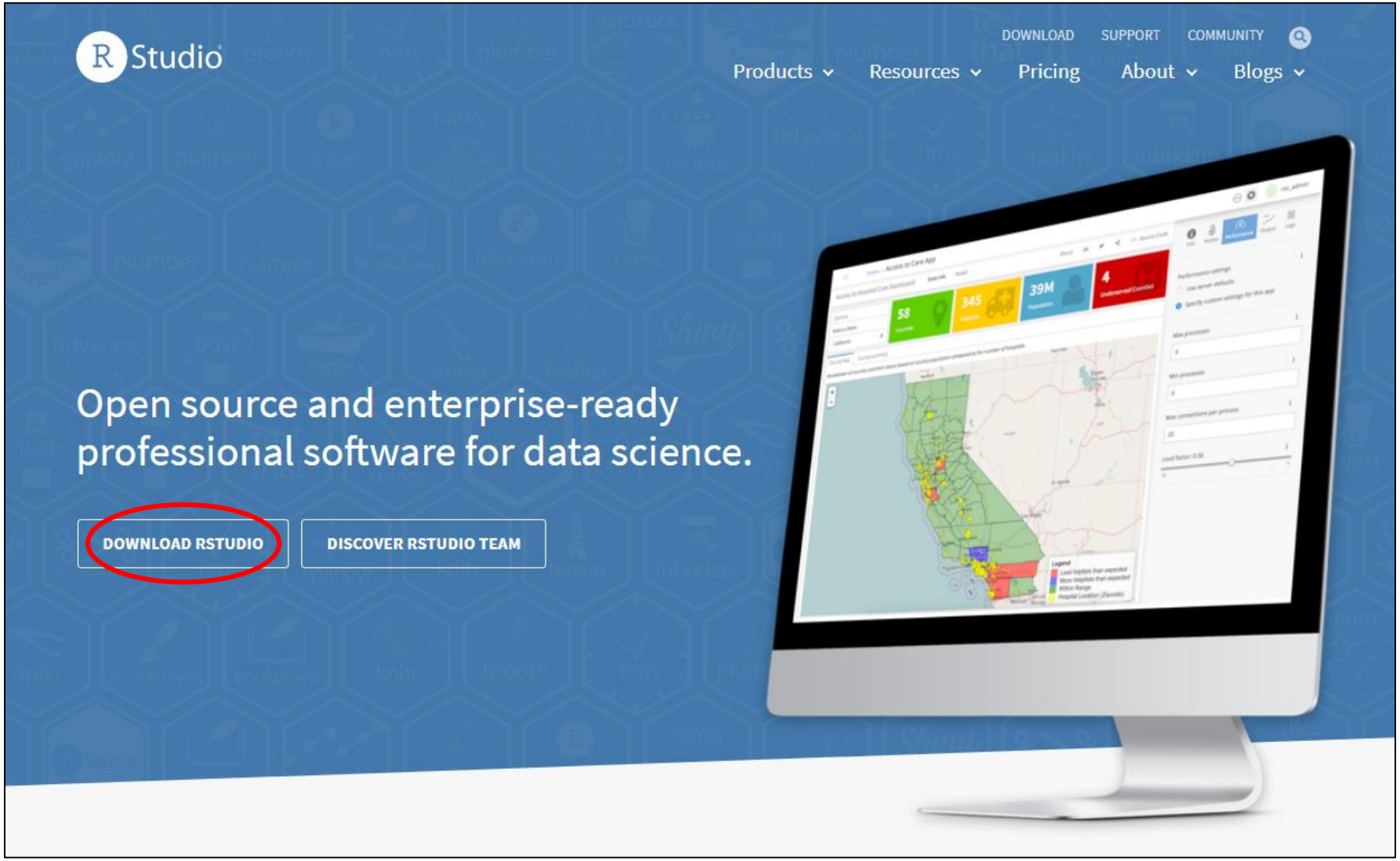
2. R설치 + R지원 프로그램





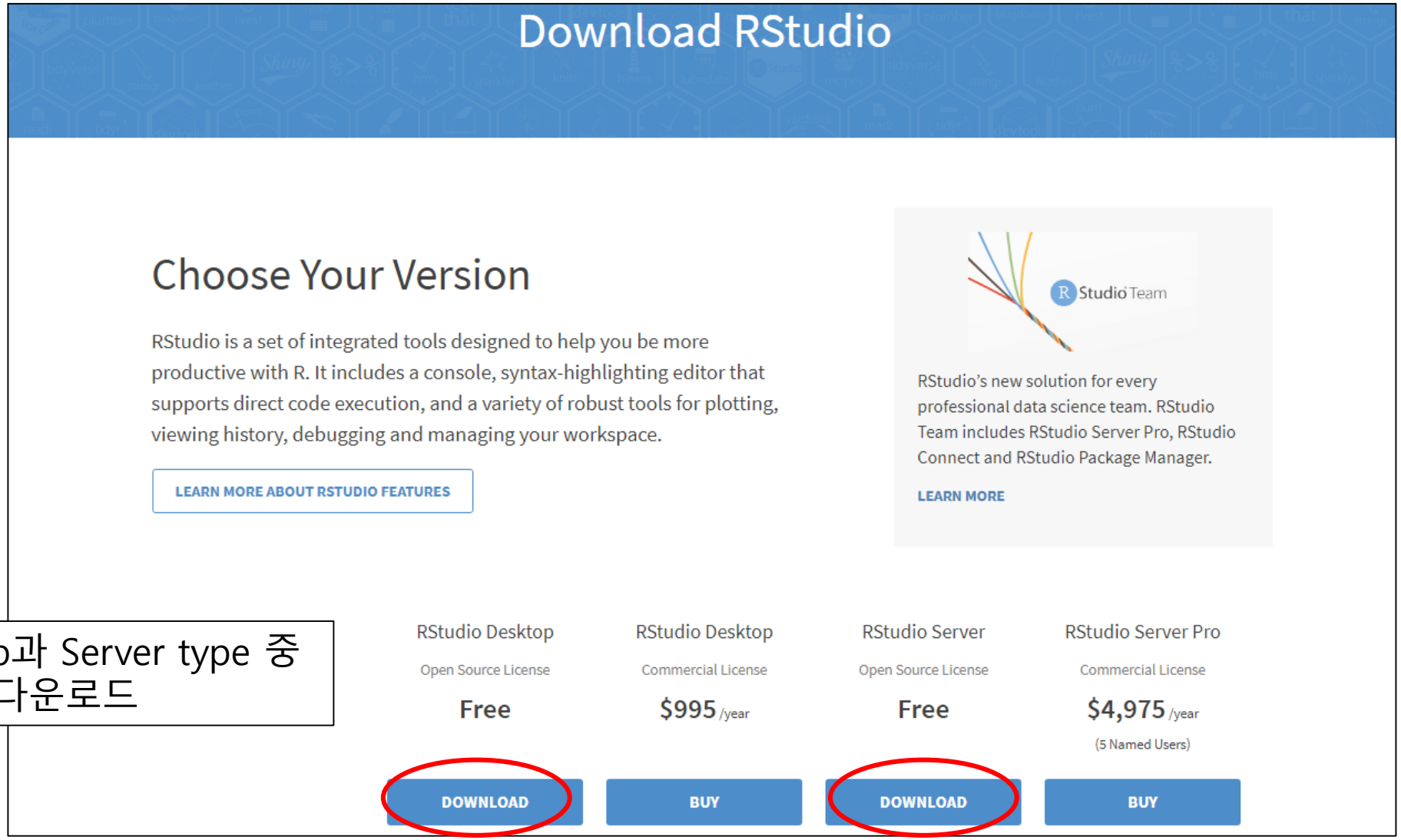
2. R설치 + R지원 프로그램

○ R studio 설치 (https://rstudio.com/)



2. R설치 + R지원 프로그램

○ R studio 설치 (https://rstudio.com/)



※ Desktop과 Server type 중 선택하여 다운로드

2. R설치 + R지원 프로그램

○ R studio 설치 (https://rstudio.com/)

All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

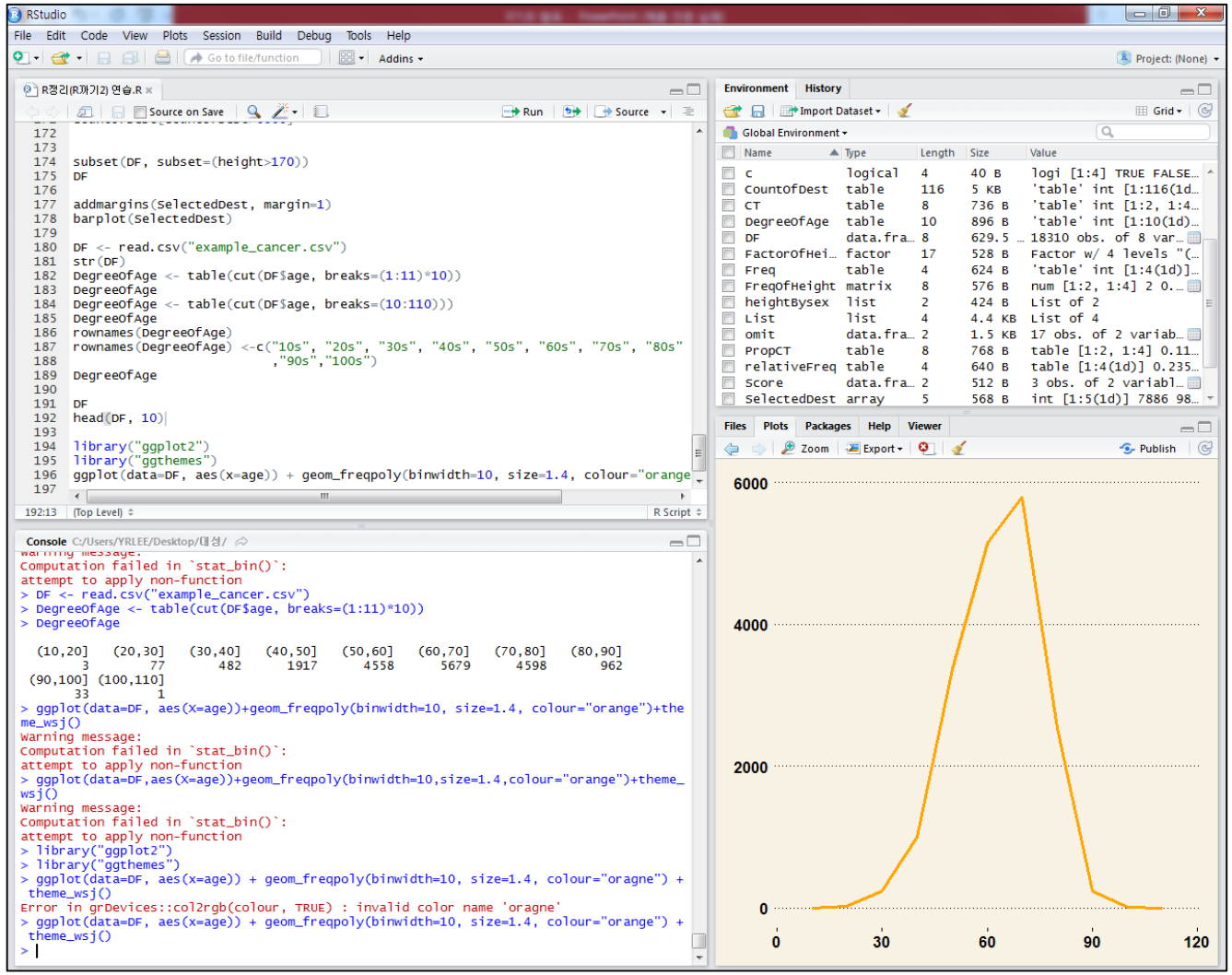
RStudio 1.2 requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.2.5019.exe	149.82 MB	7c6a943c
macOS 10.12+	RStudio-1.2.5019.dmg	126.88 MB	00cf7d64
Ubuntu 14/Debian 8	rstudio-1.2.5019-amd64.deb	96.93 MB	a0f43062
Ubuntu 16	rstudio-1.2.5019-amd64.deb	104.91 MB	24fad367
Ubuntu 18/Debian 10	rstudio-1.2.5019-amd64.deb	106.04 MB	e819293c
Fedora 19/Red Hat 7	rstudio-1.2.5019-x86_64.rpm	120.26 MB	c4fb97ce
Fedora 28/Red Hat 8	rstudio-1.2.5019-x86_64.rpm	120.89 MB	06ed9379
Debian 9	rstudio-1.2.5019-amd64.deb	106.39 MB	cd8a2413
SLES/OpenSUSE 12	rstudio-1.2.5019-x86_64.rpm	99.04 MB	87190f72
OpenSUSE 15	rstudio-1.2.5019-x86_64.rpm	107.09 MB	e4929a16

3. R 기본 요소

- R은 하나의 거대한 계산기입니다.

- 사용자가 R언어로 코드(명령어)를 입력하면, 그에 대한 결과를 보여줍니다.



3. R 기본 요소

○ R 객체(object)

Script (입력)

```
A <- 3
B <- 4

A+B
A*B

A <- c(3, 4)
D <- A
```

실행

Console (결과)

```
> A <- 3
> B <- 4
> A+B
[1] 7
> A*B
[1] 12
> A <- c(3, 4)
> D <- A
```

Envrionment (환경)

Environment | History

Import Dataset | Grid

Global Environment

Name	Type	Length	Size	Value
A	numeric	2	40 B	num [1:2] 3 4
B	numeric	1	32 B	4
D	numeric	2	40 B	num [1:2] 3 4

- R은 객체 안에 자료(데이터)를 저장합니다.

- R에서 다루는 모든 데이터는 객체에 담을 수 있습니다.

- 만들어진 객체 및 자료는 환경 창에서 확인 가능합니다.

= 객체 ≒ 변수(variable)

3. R 기본 요소

○ R 벡터(Vector)

1	4.157	A	Informatics
정수	실수	문자	문자열

= **스칼라(Scala)**

{1, 4.157, 4, A, B }

= **벡터(Vector)**

A <- 1 ⟶ {1} ※ 벡터(Vector)

3. R 기본 요소

○ R Datatype (자료 형태)

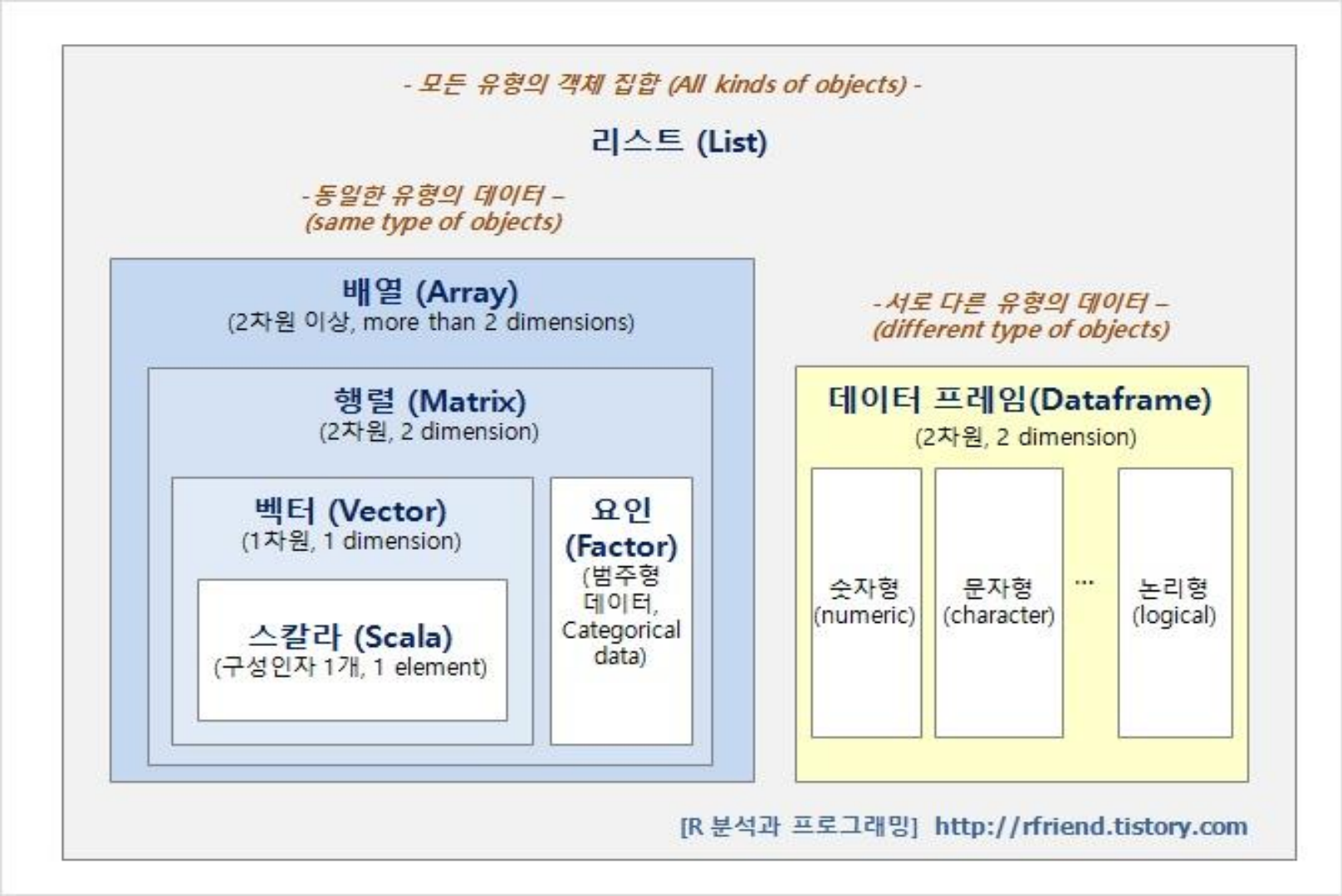
numeric	integer	character	logical
실수형, 숫자형	정수형	문자형	논리형
숫자(양수, 음수, 소수)	소수점 이하 값 없는 숫자	텍스트	TRUE/FALSE
1, 3.24	1, 2, 3	“Hello”	3>4 -> FALSE

○ R Vector class (유형)

- Matrix, Data.frame, List, etc.

3. R 기본 요소

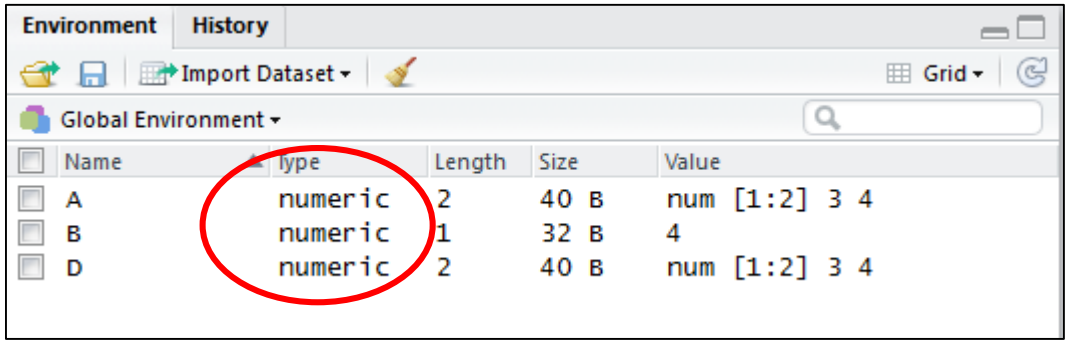
○ R Datastructure



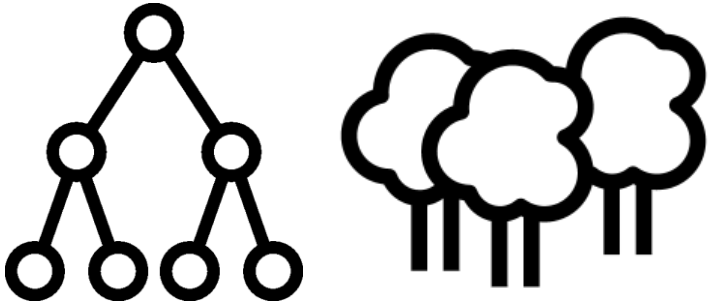
3. R 기본 요소

○ R Datastructure

numeric	integer	character	logical
실수형, 숫자형	정수형	문자형	논리형
숫자(양수, 음수, 소수)	소수점 이하 값 없는 숫자	텍스트	TRUE/FALSE
1, 3.24	1, 2, 3	“Hello”	3>4 -> FALSE



Random Forest 모델 사용 설명서

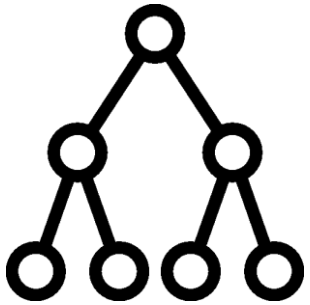


이대성, 박영석
경희대학교 생물학과
dleotjd520@naver.com, parkys@khu.ac.kr

순서

1. 의사결정나무(decision tree)
2. Random forest
3. Random forest model 적용

1. 의사결정나무란?



1. 의사 결정 나무 (Decision tree)

- 자료의 특징에 대한 질문을 하면서, 응답에 따라 자료를 분류해가는 알고리즘입니다.

(예시)

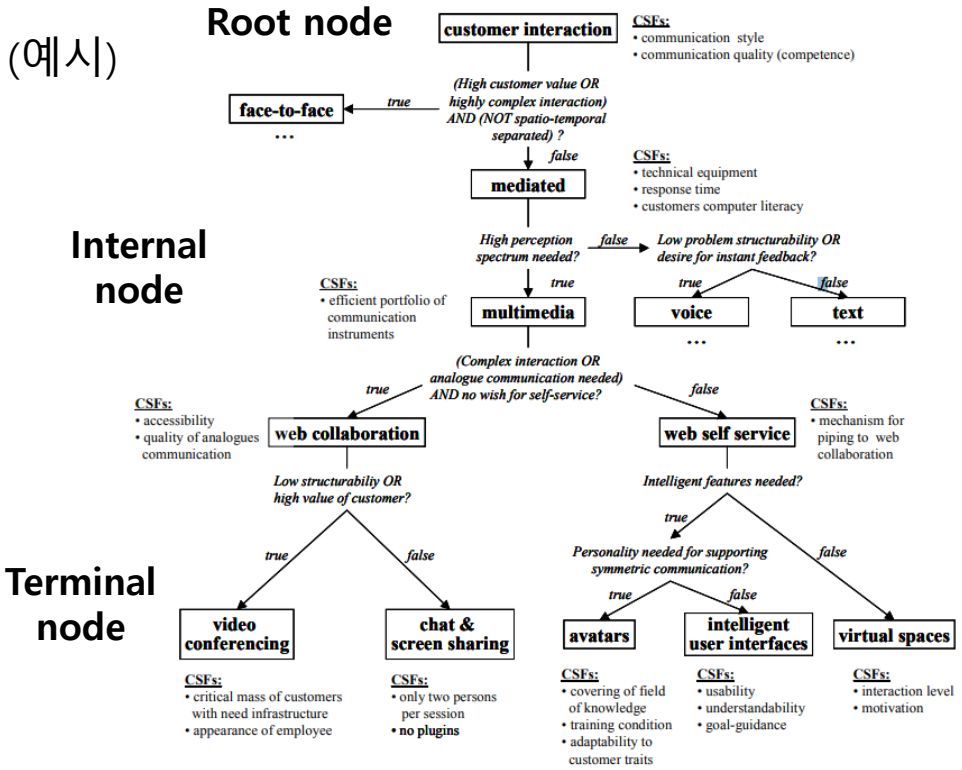


Figure 1. Customer Web Interaction Decision Tree

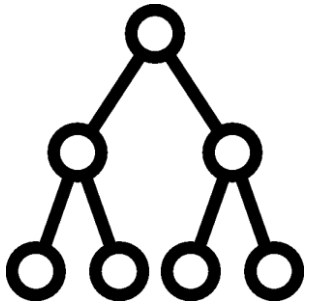
Senger et al. (2002). Customer web interaction: Fundamentals and decision tree. *AMCIS 2002 Proceedings*, 270.

- 의사결정 규칙(Rule)을 나무구조로 도표화하여, 분류(classification)와 예측(prediction)을 수행하는 분석방법입니다.

- 의사 결정 분석에서 의사 결정 나무는, 시각적으로 매우 명시적인 방법으로 ,의사 결정 과정과 결정된 의사를 보여주는데 사용됩니다.

- 데이터 마이닝 분야에서 의사 결정 나무는 결정된 의사보다도 자료 자체를 표현하는데 주로 이용됩니다.

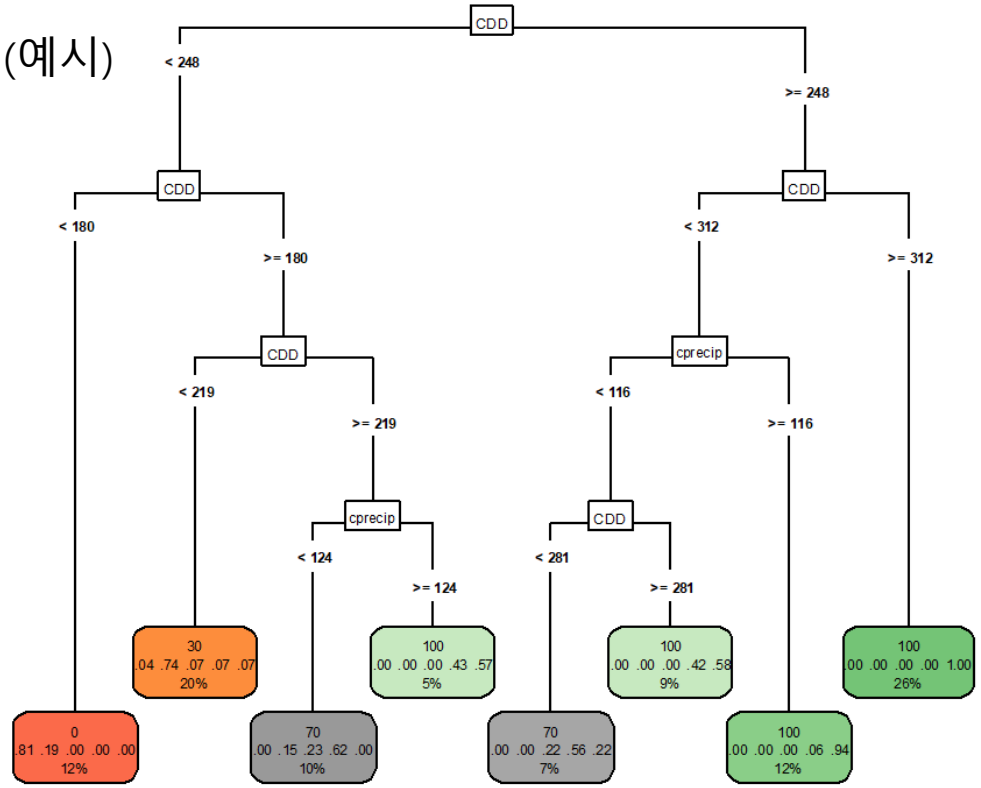
1. 의사결정나무란?



2. 분류 회귀 나무 (CART, Classification And Regression Trees)

- 의사결정나무 학습법(Learning)의 한 방법으로 분류(Classification) 및 회귀(Regresstion) 나무를 함께 일컫는 용어입니다.

(예시)

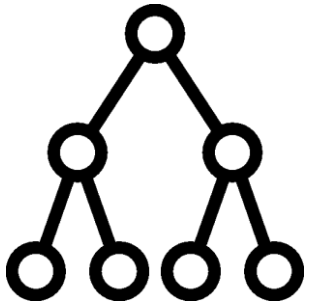


※ 분류나무(classification tree):
→ 목표 변수가 이산형인 경우, 목표변수의 각 범주(Class)에 속하는 빈도(frequency)에 기초하여 Node에서 분할이 일어납니다.

※ 회귀나무(regression tree):
→ 목표변수가 연속형(구간형)인 경우, 목표변수의 평균(mean)과 표준편차(standard deviation)에 기초하여 마디의 분리가 일어납니다.

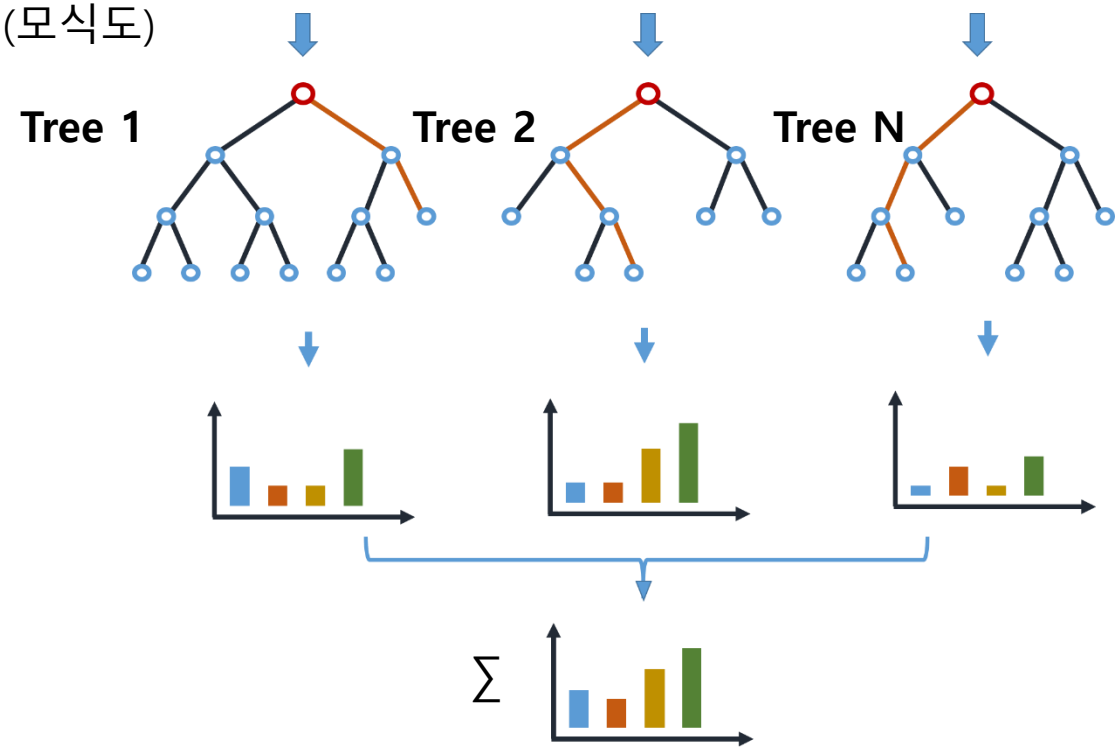
- 상위 Node에서 가지(Branch)가 분할 될 때, 지니 지수(Gini index), 엔트로피 지수(Entropy index) 등의 기준을 사용합니다..
→ 지니 지수, 엔트로피 지수 값 ∝ 하위 Node의 이질성
이질성이 작아지는 방향으로 가지 분할을 수행합니다.

2. Random forest란?



- Random forest (RF)는 분류(Classification)와 회귀(Regression)를 위해 다수의 의사결정나무 (Decision trees)를 구성 및 종합하는 앙상블 방법이다.

(모식도)



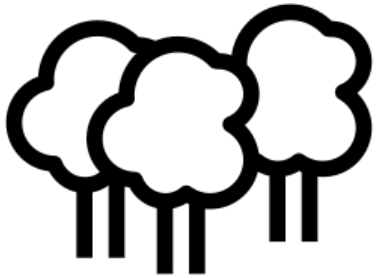
- 앙상블(Ensemble)학습 기법을 사용한 모델

→ 주어진 자료로부터 여러 개의 모델을 학습한 다음,
여러 모델의 분석 결과를 종합하여 정확도를 높이는 기법입니다.

- 분류와 회귀 문제 모두 적용가능하며,
입력 자료에 할당된 상대적인 중요성을 확인할 수 있습니다.

-의사 결정 나무를 하나가 아닌 여러 개를 사용해
과적합 문제를 방지할 수 있습니다.

2. Random forest란?



○ 절차

1. Bagging(bootstrap aggregating)을 이용하여 의사결정나무 숲 구성

- Bootstrap을 통해 무작위 T개의 훈련 자료를 구성
- T개의 개별 의사결정나무를 구성함(훈련과정)

2. 노드(Node) 개수 최적화

- 각 나무의 Node(v)마다 분할 함수(h)를 가짐, $h(v, \theta_i) \in (0, 1)$
- 분할함수는 매개변수 $\theta = (\phi, \psi, \tau)$에 의해 결정됨

※ ϕ: Node에서의 특징을 의미
※ ψ: 분할 함수의 기하학적 특성을 의미(초평면, 일반평면 등)
※ τ: 매개변수의 임계값을 의미

※ 선형적 분할함수: $h(v, \theta_i) = [\tau_1 > \phi(v) * \psi > \tau_2]$
※ 비선형적 분할함수: $h(v, \theta_i) = [\tau_1 > \phi(v)^{\mathsf{T}} * \psi * \phi(v) > \tau_2]$

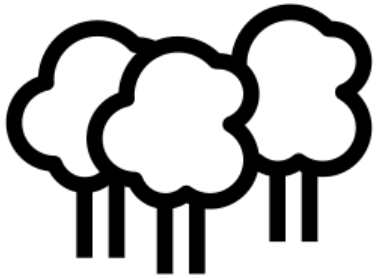
3. 앙상블(Ensemble) 모델

- 개별 의사결정나무를 종합하여 하나의 결과를 도출함

$$p(c|\theta_i) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|v)$$

- 검증 자료(Test dataset)를 가지고 훈련된 모델을 검증함

3. Random forest Model



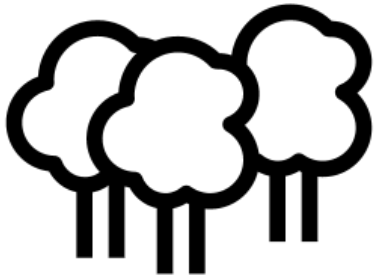
(1) 자료 확인

1. 어류 대표종 개체수 자료(Abundance)

- 어류 각 대표종 자료를 사용하여 작성합니다('수생태_어류_생물'파일).
- 앞 1개열은 조사 지점정보, 그 뒷열부터는 각 대표 생물 종에 해당하며, 행은 각 지점을 의미합니다.
- 각 대표종 열에 들어간 값은 각 어류 대표종의 개체수를 의미합니다.
- 입력에 필요한 대상 생물종수의 제한은 없으며, 2번째 열 이후로만 지정해주면 됩니다.

	A	B	C	D	E	F	G	H	I
1	Total_Code	피라미	참갈겨니	돌고기	배스	블루길	감돌고기	묵납자루	어름치
2	1001G002	34.5	41.9375	3.625	0	0	0	0	0
3	1001G004	27.75	42.75	3.0625	0	0	0	0	0
4	1001G006	0	48	4.5	0	0	0	0	0
5	1001G008	91.5	145	64	0	0	0	22.5	5
6	1001G012	0	69	16.5	0	0	0	0	0
7	1001G014	16.875	52.625	11.5	0	0	0	4.625	1.6875

3. Random forest Model



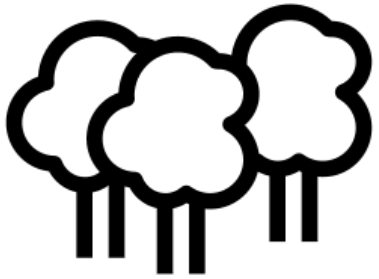
(1) 자료 확인

2. 환경자료

- 모델에 사용할 환경자료입니다. 행은 지점, 열은 환경 변수를 의미합니다.
- 환경 변수의 종류는 상관없으나, 자료의 형태는 수치형자료(예, 1, 2.1, 120.2 등)이어야 합니다.
- 입력한 환경 변수 중 실제 모델 제작에 사용할 환경 변수는 차후 사용자가 직접 설정합니다.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Total_Cod	하천차수	수온	DO	pH	Conductiv	Turbidity	BOD	NH3-N	NO3-N	T-N	PO4-P	T-P	Chl-a	1시가화건	2농업지역
2	1001G002	5	15.4	11.8	8.4	214.1	0.6	0.7	0.032	1.697	2.52	0.008	0.02	0.6	1.072	24.129
3	1001G004	5	15.2	9.8	8.1	57.3	0.8	0.7	0.011	0.771	1.44	0.004	0.01	0.8	2.353	18.593
4	1001G006	4	17.4	17.6	7.7	33.5	1.8	1.3	0.106	1.303	1.736	0.026	0.038	2.1	0.372	5.597
5	1001G008	6	23.1	6.7	8.1	263.5	2.6	1.9	0.041	2.599	2.828	0.015	0.024	1.3	3.941	63.812
6	1001G012	4	16.6	13.2	7.7	194	66	1.3	0.124	1.061	3.704	0.045	0.09	2.5	3.214	52.682
7	1001G014	6	19.4	11.1	8.4	170.9	1.7	0.8	0.02	1.337	2.315	0.007	0.017	1.2	17.749	35.215
8	1001G024	5	17.9	10.7	8.4	116.1	2.5	0.8	0.032	1.52	2.356	0.008	0.02	1.1	3.673	41.396
9	1001G026	6	17.6	10.2	8	151.4	2.4	0.9	0.03	1.339	2.263	0.006	0.015	1	8.641	47.352
10	1001G036	5	18.2	11.2	8.3	120.3	3	0.9	0.029	1.144	2.144	0.006	0.018	1.1	7.199	32.818
11	1001G038	6	18.9	11.8	8.7	151.3	6.3	0.9	0.022	1.089	2.056	0.004	0.012	1	47.169	0.138

3. Random forest Model



(2) 자료 불러오기

※ **'검정색'**: R script창 내 코드

※ **'>파란색'**: R console 내 결과물

```
#######################################################################
################################### 1. 자료 불러오기
## 풍부도자료(Abundance): 개체수
sp.DB <- read.csv("수생태_어류_생물.csv", stringsAsFactors=F)
str(sp.DB)

## 환경 자료
Env.DB <- read.csv("수생태_어류_환경.csv", stringsAsFactors=F)
str(Env.DB)
```

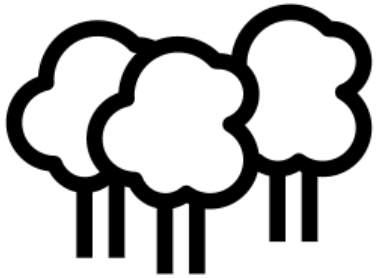
→ 문자가 함께 혼용되어 있는 자료 불러오기

→ 자료 확인 함수: str(), head(), tail()

```
> str(sp.DB)
'data.frame':   459 obs. of  9 variables:
 $ Total_Code: chr  "1001G002" "1001G004" "1001G006" "1001G008" ...
 $ 피라미    : num  34.5 27.8 0 91.5 0 ...
 $ 참갈겨니  : num  41.9 42.8 48 145 69 ...
 $ 돌고기    : num  3.62 3.06 4.5 64 16.5 ...
 $ 배스      : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 블루길    : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 감돌고기  : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 묵납자루  : num  0 0 0 22.5 0 ...
 $ 어름치    : num  0 0 0 5 0 ...
```

```
> str(Env.DB)
'data.frame':   459 obs. of  32 variables:
 $ Total_Code      : chr  "1001G002" "1001G004" "1001G006" "1001G008" ...
 $ 하천차수        : int  5 5 4 6 4 6 5 6 5 6 ...
 $ 수온            : num  15.4 15.2 17.4 23.1 16.6 19.4 17.9 17.6 18.2 18.9 ...
 $ DO              : num  11.8 9.8 17.6 6.7 13.2 11.1 10.7 10.2 11.2 11.8 ...
 $ pH              : num  8.4 8.1 7.7 8.1 7.7 8.4 8.4 8 8.3 8.7 ...
 $ Conductivity    : num  214.1 57.3 33.5 263.5 194 ...
 $ Turbidity       : num  0.6 0.8 1.8 2.6 66 1.7 2.5 2.4 3 6.3 ...
 $ BOD             : num  0.7 0.7 1.3 1.9 1.3 0.8 0.8 0.9 0.9 0.9 ...
 $ NH3.N           : num  0.032 0.011 0.106 0.041 0.124 0.02 0.032 0.03 0.029 0.022 ...
 $ NO3.N           : num  1.697 0.771 1.303 2.599 1.061 ...
 $ T.N             : num  2.52 1.44 1.74 2.83 3.7 ...
 $ PO4.P           : num  0.008 0.004 0.026 0.015 0.045 0.007 0.008 0.006 0.006 0.004 ...
 $ T.P             : num  0.02 0.01 0.038 0.024 0.09 0.017 0.02 0.015 0.018 0.012 ...
 $ Chl.a           : num  0.6 0.8 2.1 1.3 2.5 1.2 1.1 1 1.1 1 ...
 $ X1시가화건조지역: num  1.072 2.353 0.372 3.941 3.214 ...
 $ X2농업지역      : num  24.1 18.6 5.6 63.8 52.7 ...
 $ X3산림지역      : num  69.04 37.53 86.03 9.41 30.63 ...
 $ X4초지지역      : num  0.412 11.575 0 5.892 0.288 ...
 $ X5습지          : num  1.826 0.151 5.36 5.514 10.217 ...
 $ X6나지          : num  0.714 1.632 1.092 0 0 ...
 $ X7수역          : num  2.81 28.16 1.55 11.43 2.97 ...
 $ 비율            : num  100 100 100 100 100 ...
 $ slope_R         : num  0.324 6.204 6.066 0 0 ...
 $ aspect_gra      : int  8 1 6 0 0 0 0 0 7 5 ...
 $ tavg_R          : num  6.55 7.29 6.75 7.83 7.46 ...
 $ tmax_7m_R       : num  24.3 24.3 24.3 24.8 24.3 ...
 $ tmin_1m_R       : num  -13.1 -12.1 -13 -12.3 -12.4 ...
 $ prcp_R          : num  1222 1186 1203 1160 1172 ...
 $ godo_R          : num  720 680 680 500 520 ...
 $ 수폭m           : num  14.5 10.1 8.3 15 6.3 ...
 $ 수심cm          : num  17 20.3 33.9 25 31.9 21.8 20.9 25.7 19.4 22.7 ...
 $ 유속cm.s        : num  48.2 68.5 72.4 14 91.2 55.4 72.9 78.1 60.7 60.5 ...
```

3. Random forest Model



(2) 자료 정리하기

※ **'검정색'**: R script창 내 코드
※ **'>파란색'**: R console 내 결과물

```
##### 생물 자료(풍부도) 확인 및 DB생성
names(sp.DB)
```

```
> names(sp.DB)
[1] "Total_Code" "피라미"     "참갈겨니"    "돌고기"     "배스"       "블루길"     "감돌고기"    "묵납자루"    "어름치"
```

※ 지점 + 8종

```
names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
```

```
> names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
[1] "피라미"   "참갈겨니" "돌고기"   "배스"     "블루길"   "감돌고기" "묵납자루" "어름치"
```

```
used.sp.DB <- sp.DB[2:length(names(sp.DB))]
```

→ 생물 종 DB 생성 (피라미~어름치)

```
# 지점 명칭 부여(DB 행)
rownames(used.sp.DB)
rownames(used.sp.DB) <- sp.DB[,1]
```

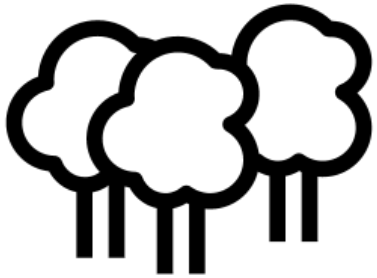
```
> rownames(used.sp.DB)
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"
```

→ 'rowname' = DB 행 의미(지점)
만들어진 생물 종 DB에 지점정보(sp.DB[,1])를 반영해줌

```
> used.sp.DB
        피라미     참갈겨니
1   34.5000000  41.9375000
2   27.7500000  42.7500000
3    0.0000000  48.0000000
4   91.5000000 145.0000000
5    0.0000000  69.0000000
6   16.8750000  52.6250000
```

```
rownames(used.sp.DB)
[1] "1001G002" "1001G004" "1001G006" "1001G008" "1001G012"
```

3. Random forest Model

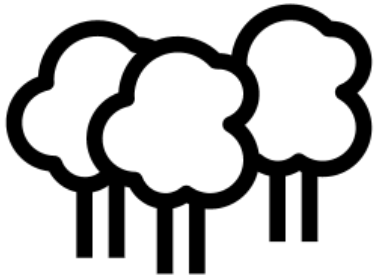


(2) 자료 정리하기

```
##### 환경 자료확인
names(Env.DB)
 > names(Env.DB)
  [1] "Total_Code"      "하천차수"        "수온"            "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"
  [9] "NH3.N"           "NO3.N"           "T.N"             "PO4.P"           "T.P"             "Chl.a"           "X1시가화건조지역" "X2농업지역"
 [17] "X3산림지역"      "X4초지지역"      "X5습지"          "X6나지"          "X7수역"          "비율"            "slope_R"         "aspect_gra"
 [25] "tavg_R"         "tmax_7m_R"       "tmin_1m_R"       "prcp_R"          "godo_R"          "수폭m"           "수심cm"          "유속cm.s"

names(Env.DB)[1] #조사 지점 정보
names(Env.DB)[2] #하천차수
names(Env.DB)[3] #수온
names(Env.DB)[4:14] #수질1
names(Env.DB)[15:21] #토지피복
names(Env.DB)[23] #경사각
names(Env.DB)[29] #고도
names(Env.DB)[25:28] #기상조건
names(Env.DB)[30:32] #수리수문

##### 사용할 환경변수 지정
var.DB <- Env.DB[c(2,#하천차수
                   4:8,11,13:14,#수질- DO, pH, Conductivity, Turbidity, BOD, / T-N, / T-P, Chl-a
                   15,17,#토지피복- 1시가화건조지역, 3산림지역
                   29,#고도
                   25:28,#기상조건
                   30:32#수리수문- 수폭,수심,유속
)]
```

→ 환경 변수 중 일부만 사용

3. Random forest Model



(2) 자료 정리하기

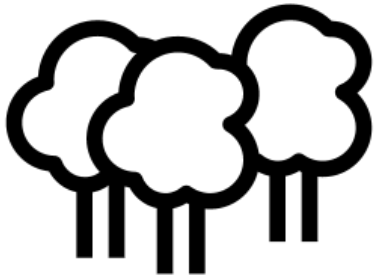
```
names(var.DB) #사용 환경변수 명칭
length(names(var.DB)) #사용 환경변수 개수
```

```
> names(var.DB) #사용 환경변수 명칭
 [1] "하천차수"        "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"             "T.N"             "T.P"
 [9] "Chl.a"           "X1시가화건조지역" "X3산림지역"      "godo_R"          "tavg_R"          "tmax_7m_R"       "tmin_1m_R"       "prcp_R"
[17] "수폭m"           "수심cm"          "유속cm.s"
> length(names(var.DB)) #사용 환경변수 개수
[1] 19
```

```
Total.DB <- cbind(used.sp.DB, var.DB)
```

→ 환경 + 생물 DB

3. Random forest Model



(3) 모델 설정

```
#######################################################################
################################### 2. 모델 설정
##### 어류
# 사용모델: RF
# 자료형태: P/A

##### 필요 package
# install.packages(dismo)
# install.packages(randomForest)
# install.packages(ROCR)
# install.packages(caret)
# install.packages(mgcv)
library(dismo)
library(randomForest)
library(ROCR)
library(caret)
library(mgcv)
```

※ Random forest model용 package

→ randomForest, dismo

※ 모델 평가용 package

→ ROCR, caret

※ 부분의존성 그림(PDP)용 package

→ mgcv

```
###### 모의 생물 종 선택
#모델을 제작하고자 하는 생물 종 선택
#형성된 자료의 열을 기준으로 생성
target.sp = 1 #

names(used.sp.DB)[target.sp] #대상 생물 확인
```

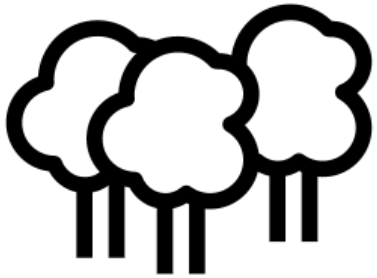
※ 8종의 생물 중 모의 대상 종 선택

```
> names(used.sp.DB)[target.sp] #대상 생물 확인
[1] "피라미"
```

```
> names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
[1] "피라미"   "참갈겨니" "돌고기"   "배스"     "블루길"   "감돌고기" "묵납자루" "어름치"
```

(1) ~ **(8)**

3. Random forest Model



(4) 자료의 전처리

```
#######################################################################
#################################### 3. 자료의 전처리 과정
targetDB <- Total.DB
total.num.sp <- dim(used.sp.DB)[2] # DB 내 포함된 생물종 수
```

= 8종

생물

환경

```
> names(targetDB)
 [1] "피라미"          "참갈겨니"        "돌고기"          "배스"            "블루길"          "감돌고기"        "묵납자루"        "어름치"
 [9] "하천차수"        "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"             "T.N"             "T.P"
[17] "Chl.a"           "X1시가화건조지역" "X3산림지역"      "godo_R"          "tavg_R"          "tmax_7m_R"       "tmin_1m_R"       "prcp_R"
[25] "수폭m"           "수심cm"          "유속cm.s"
```

```
### 전처리 과정
#1. 사용 Data만 선택
m1 <- targetDB[,c(target.sp,(1+total.num.sp):dim(targetDB)[2])]
print(colnames(m1)[1]) #모의 생물 종 명칭
```

1 1+8 27

※ DB 중 선택 생물 열(target.sp) + 환경 열

→ 피라미

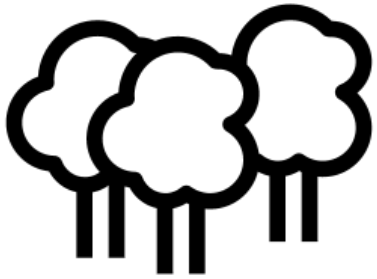
```
#2. 자료 내 NA 제거
m2 <- na.omit(m1)
names(m2)[1] <- "pop" #모의를 위한 종명칭 변경
```

→ 환경자료 내 자료 NA 지점 제거

```
#3. 생물 자료 정수화
m3 <- m2
m3[,1] <- round(m3[,1],0) # 정수화
```

→ 정수화 예시: 피라미 1.1마리 -> 1마리

3. Random forest Model



(4) 자료의 전처리

```
#4. 출현여부(P/A) 변환 (1/0)
m4 <- m3
for(i in 1:length(m4[,1])){
  if(m3[i,1] > 0){
    m4[i,1] <- 1
  } else
    m4[i,1] <- 0
}# for i
```

→변환(생물): 생물 개체수(정수) → 출현여부(0, 1)

※ [,1]: 1번 열 = 생물(≒피라미)열

※ length(m4[,1]): 1번 열(생물)의 전체 행 개수(≒ 지점 수)

```
#5. 환경 변수 표준화(scale))
m5 <- m4
#str(e1)
m5[,2:(dim(m4)[2])] <- scale(m4[,2:(dim(m4)[2])]) # 표준화: z= (Mean - X)/sd
```

→변환(환경): 환경열을 표준화(Standadization)함

e.g. DB[1] ≒ dim(DB)[2]

length(DB[,1]) ≒ dim(DB)[1]

(No)	피라미
1	1
2	0
3	1
4	1

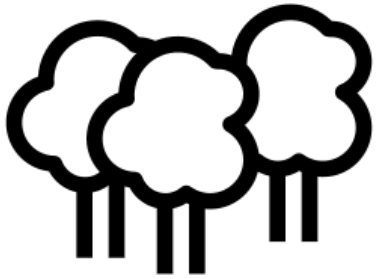
```
#6. 환경 변수 정보(var.info) 저장
target.env.DB <- m4[,2:(dim(m4)[2])] # scale 하기 바로 직전 DB
total.var.name <- names(target.env.DB)
```

= m4

```
### 표준화 정보
var.info <- as.data.frame(matrix(NA, nrow=length(total.var.name), ncol=3))
names(var.info) <- c("name", "mean", "sd")
for( i in 1:length(total.var.name)){
  var.info[i,1] <- total.var.name[i]
  var.info[i,2] <-mean(target.env.DB[,i], na.rm=T)
  var.info[i,3] <-sd(target.env.DB[,i], na.rm=T)
}
```

```
> var.info
                name          mean           sd
1           하천차수    5.51034483 1.983052e+00
2                DO    9.41471264 1.862191e+00
3                pH    7.89172414 5.201697e-01
4      Conductivity  298.82321839 1.131712e+03
5         Turbidity   10.09839080 2.004259e+01
6               BOD    1.93471264 1.110015e+00
7               T.N    2.38296552 1.217586e+00
8               T.P    0.06201379 9.026808e-02
9             Chl.a    3.72850575 6.416491e+00
10 X1시가화건조지역   11.03603908 1.325349e+01
11       X3산림지역   26.55498621 2.354068e+01
12           godo_R  104.70679977 1.168564e+02
13           tavg_R   10.95712149 1.350842e+00
14        tmax_7m_R   27.84787563 9.843470e-01
15        tmin_1m_R   -8.42283251 2.617238e+00
16           prcp_R 1232.35022299 1.018252e+02
17            수폭m   85.40620690 1.182695e+02
18           수심cm   30.60091954 1.783805e+01
19         유속cm.s   25.47241379 2.329885e+01
```

3. Random forest Model



(5) 모델링

```
#######################################################################
################################### 4. 모델링
# 모델용 DB
model.DB <-  m5
```

= 변환 완료 DB

```
# model.DB 구성
dim(model.DB) # 20개 변수(pop + 19개), 435개 지점(site)
```

```
> dim(model.DB) # 20개 변수(pop + 19개), 435개 지점(site)
[1] 435  20
```

```
# 사용 DB 내 환경변수 열 지정
var.num <- c(2:dim(m5)[2])
names(model.DB)[var.num] # 사용 환경 변수명
```

→ 2:20

```
> names(model.DB)[var.num] # 사용 환경 변수명
 [1] "하천차수"          "DO"                "pH"                "Conductivity"      "Turbidity"         "BOD"               "T.N"               "T.P"
 [9] "Chl.a"             "X1시가화건조지역"  "X3산림지역"        "godo_R"            "tavg_R"            "tmax_7m_R"         "tmin_1m_R"         "prcp_R"
[17] "수폭m"             "수심cm"            "유속cm.s"
```

```
##### 훈련(Training) 및 검증(Test) 자료 분할
DB_P <- subset(model.DB, model.DB$pop > 0) #출현 DB
DB_A <- subset(model.DB, model.DB$pop == 0) #비출현 DB

# 분할(훈련:검증 = 8:2)
set.seed(6808)
no.p <- sample(1:dim(DB_P)[1], dim(DB_P)[1]*0.8, replace=F)
no.a <- sample(1:dim(DB_A)[1], dim(DB_A)[1]*0.8, replace=F)

train_DB <- rbind(DB_P[no.p,], DB_A[no.a,])
test_DB <-  rbind(DB_P[-no.p,], DB_A[-no.a,])
dim(train_DB)
dim(test_DB)
```

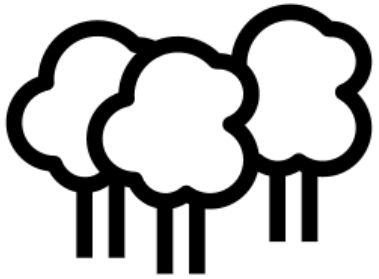
→ 80%

→ 20%

※ set.seed(): random sample 패턴 고정 → 재현 가능

※ replace=F: 복원추출 방지

3. Random forest Model



(5) 모델링: 훈련

```
##### RF model 제작
RF.DB <- train_DB[c(1,var.num)]
RF.test_DB <- test_DB

set.seed(6808)
RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
```

※ 도움말 (?randomForest)

Usage

```
# S3 method for formula
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
# S3 method for default
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
# S3 method for randomForest
print(x, ...)
```

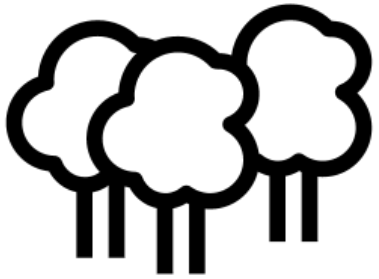
※ formula 형태: 종속변수 ~ 독립변수

→ pop(=생물) ~ . (=모든 변수; All)

(다른 예시) pop ~ 하천차수 + DO

x, formula a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the `print` method, an `randomForest` object).

3. Random forest Model



(5) 모델링: 훈련

```
##### RF model 제작
RF.DB <- train_DB[c(1,var.num)]
RF.test_DB <- test_DB

set.seed(6808)
RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
```

※ 도움말 (?randomForest)

Usage

```
# S3 method for formula
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
# S3 method for default
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
# S3 method for randomForest
print(x, ...)
```

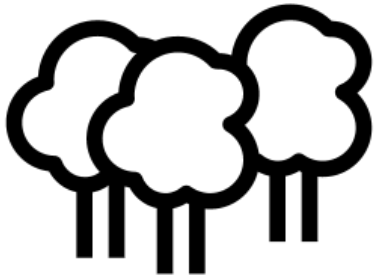
※ RF 모델 내 인자

- ntree: RF 모델 내 의사결정나무 모형 개수
- importance: 변수의 중요도 표시 여부
- mtry: 각 노드 분할에서 사용하는 변수의 개수

(초기값: regression의 경우 변수갯수/3, classification의 경우 전체변수 개수)

- na.action: 입력 자료 내 결측치(NA)가 있는 경우의 처리 방법 설정

3. Random forest Model



(5) 모델링: 훈련

```
##### RF model 제작
RF.DB <- train_DB[c(1,var.num)]
RF.test_DB <- test_DB

set.seed(6808)
RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
```

```
> RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
경고메시지(들):
In randomForest.default(m, y, ...) :
  The response has five or fewer unique values.  Are you sure you want to do regression?
> RF.model

Call:
 randomForest(formula = pop ~ ., data = RF.DB, importance = TRUE,      ntree = 1000)
               Type of random forest: regression
                     Number of trees: 1000
No. of variables tried at each split: 6

          Mean of squared residuals: 0.1227525
                    % Var explained: 27.55
```

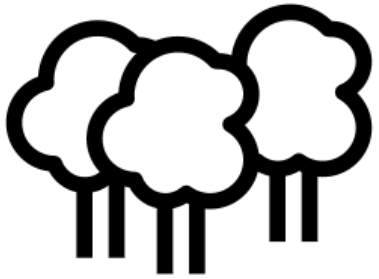
→ **정상임**

※ **Warning massage**와 **Error massage**는 다름

→ Warning: 결과 도출 후 경고,

→ Error: 결과 미도출 및 오류 표시

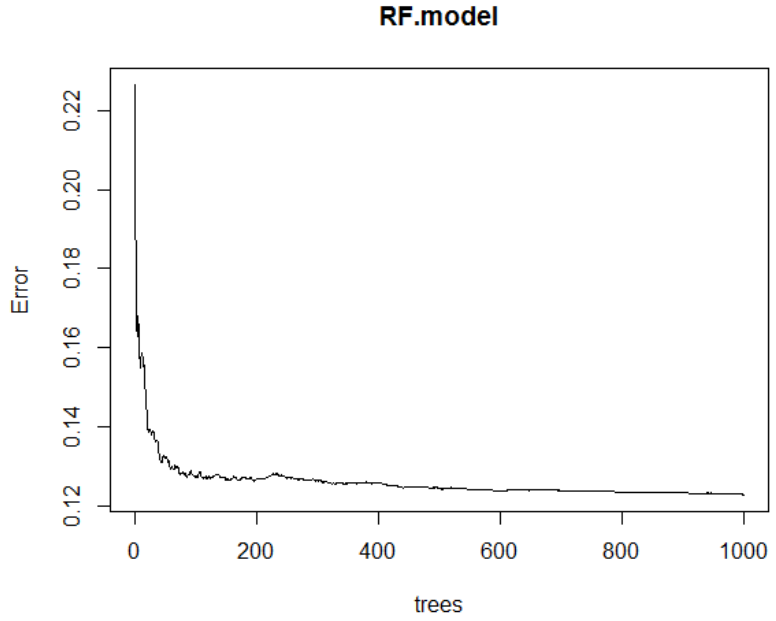
3. Random forest Model



(5) 모델링: 훈련

```
##### RF model 제작
RF.DB <- train_DB[c(1,var.num)]
RF.test_DB <- test_DB

set.seed(6808)
RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
```

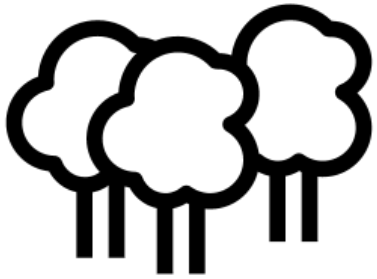
```
> plot(RF.model)
```



→ 회귀모델 내 오차(Error)의 변화를 확인할 수 있음

※ x축 'Trees'는 'ntree'를 의미

3. Random forest Model



(5) 모델링: 훈련

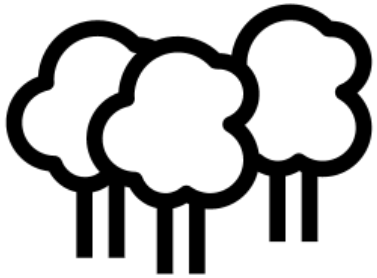
```
##### RF model 제작
RF.DB <- train_DB[c(1,var.num)]
RF.test_DB <- test_DB

set.seed(6808)
RF.model <- randomForest(pop ~. ,data=RF.DB, importance=TRUE, ntree=1000)
e.rf<- evaluate(subset(test_DB[c(1,var.num)], test_DB$pop > 0), subset(test_DB[c(1,var.num)], test_DB$pop == 0), RF.model)
t.rf <- threshold(e.rf, 'spec_sens')
```

※ evaluate(): Model evaluation function

※ threshold(): Find threshold (cut-off) between P and A

→ Model에서 제시해주는 값임

3. Random forest Model



(5) 모델링: 검증 (ACC, AUC index)

※ predict(): 훈련된 모델에 새로운 자료 적용

```
##### 모델 성능 평가
# 모델 검증 결과(test)
RF.test <- predict(RF.model, RF.test_DB, type='class')
```

```
> predict(RF.model, RF.test_DB)
 1001G004  1001G026  1001G038  1001G044  1001G060  1003G002  1003G044
0.5531000 0.9557000 0.9429833 0.7978167 0.5727667 0.9731333 0.9163000
```

※ 0 ~ 1 사이 값 → Probability

```
### 1) AUC
RF_p1 <- ROCR::prediction(as.numeric(as.vector(RF.test)), RF.test_DB$pop)
RF_pp1 <- performance(RF_p1, 'tpr', 'fpr')
RF_auc1 <- performance(RF_p1, "auc")
RF_auc <- RF_auc1@y.values
print(paste0("AUC : ",round(RF_auc[[1]],3)))
```

```
### 2) ACC
test_matrix <- RF.test_DB[,1]
for(i in 1:length(test_matrix)){
  if(RF.test[i] > t.rf){test_matrix[i] <- 1} else {test_matrix[i] <- 0}
}
RF_test_ACC <- confusionMatrix(factor(RF.test_DB[,"pop"]),factor(test_matrix))$overall[1]
print(paste0("ACC : ",round(RF_test_ACC,3)))
```

```
> print(paste0("AUC : ",round(RF_auc[[1]],3)))
[1] "AUC : 0.818"
> print(paste0("ACC : ",round(RF_test_ACC,3)))
[1] "ACC : 0.773"
```

종합

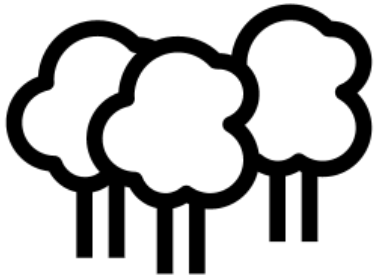
```
# AUC, ACC 모음
RF.mvalue.df <- data.frame(matrix(NA, nrow=1,ncol=2))
names(RF.mvalue.df) <- c("AUC", "ACC")
RF.mvalue.df[1,1] <- round(RF_auc[[1]],3) #AUC
RF.mvalue.df[1,2] <- round(RF_test_ACC,3) #ACC
# RF.mvalue.df
```

```
> RF.mvalue.df
    AUC   ACC
1 0.818 0.773
```

```
##### 입력자료에 대한 최종 서식확률 계산
RF.result <- predict(RF.model, model.DB[-1])
```

※ Model DB = Train + Test data

3. Random forest Model



※ AUC(AUROC)와 ACC

- Confusion matrix

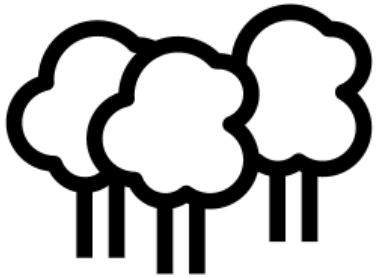
- ACC (Accuracy): 정확도

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

※ https://en.wikipedia.org/wiki/Confusion_matrix

※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

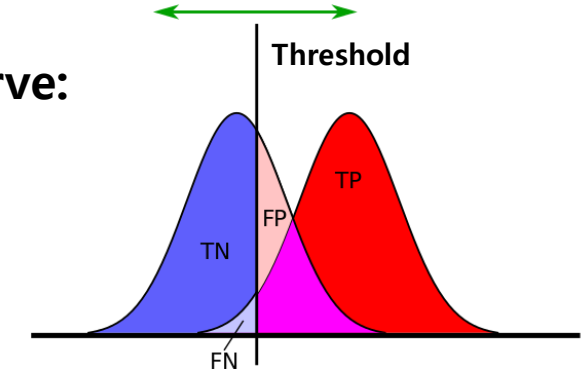
3. Random forest Model



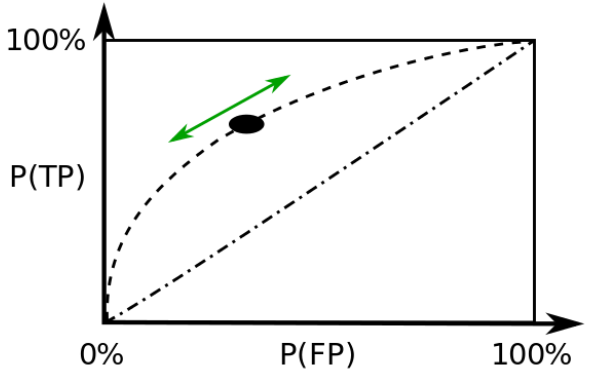
※ AUC(AUROC)와 ACC

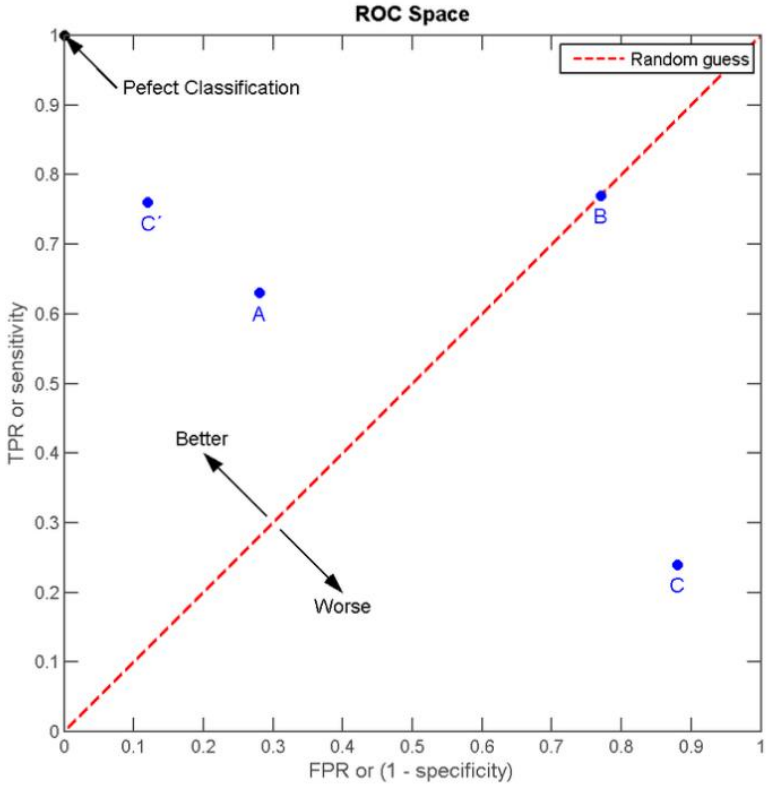
- Area Under a ROC Curve (AUROC; AUC)

ROC curve:



TP	FP
FN	TN

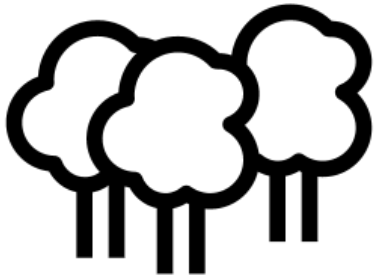




※ 50%

※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

3. Random forest Model



※ AUC(AUROC)와 ACC

- Confusion matrix

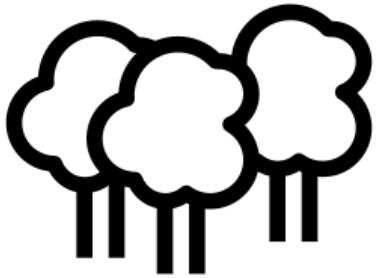
		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

- AUC

※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

※ https://en.wikipedia.org/wiki/Confusion_matrix

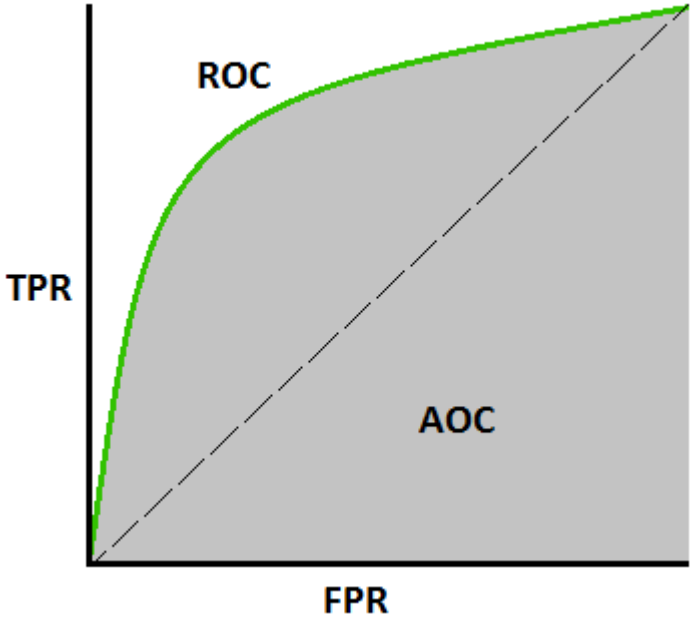
3. Random forest Model

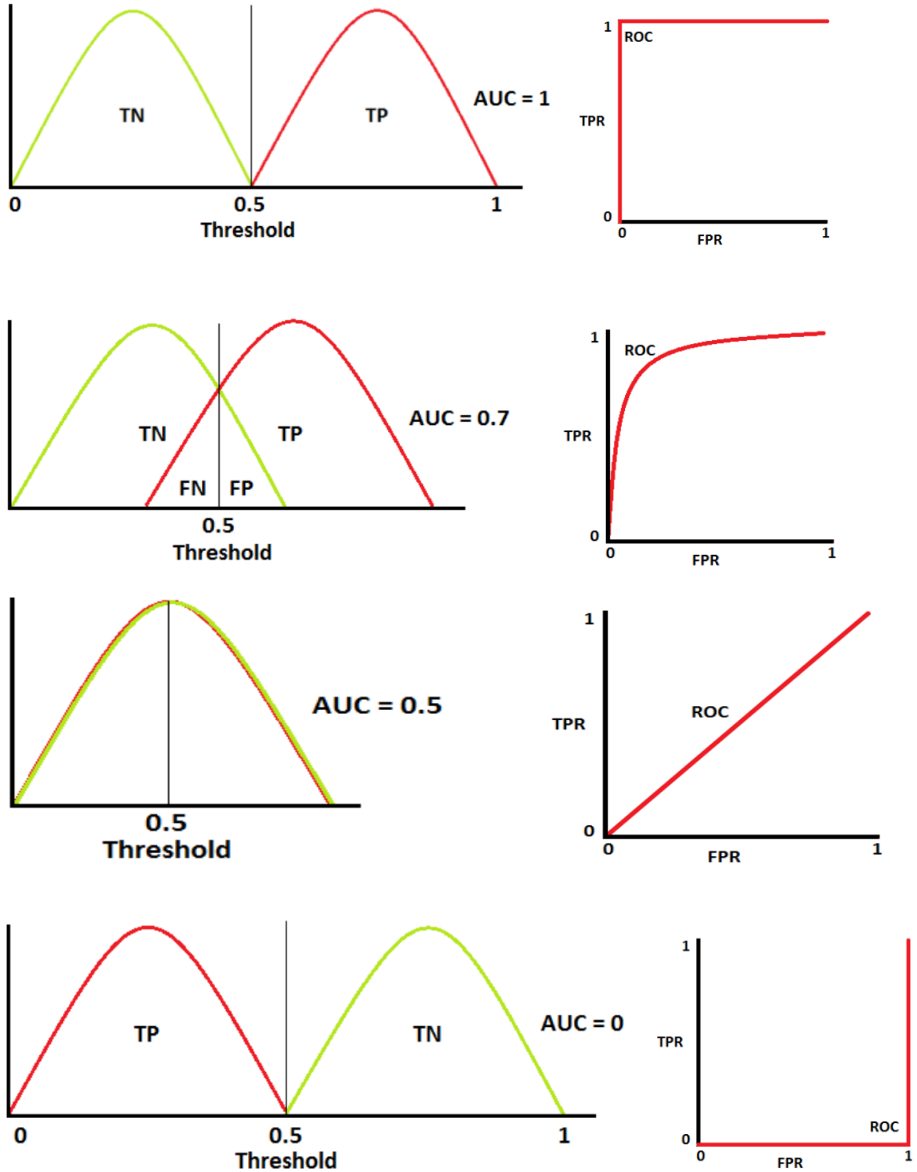


※ AUC(AUROC)와 ACC

- Area Under a ROC Curve (AUROC; AUC)

AUC:





※ **Writer: Sarang Narkhede**
※ https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

3. Random forest Model

(5) 모델링 결과 정리 및 저장

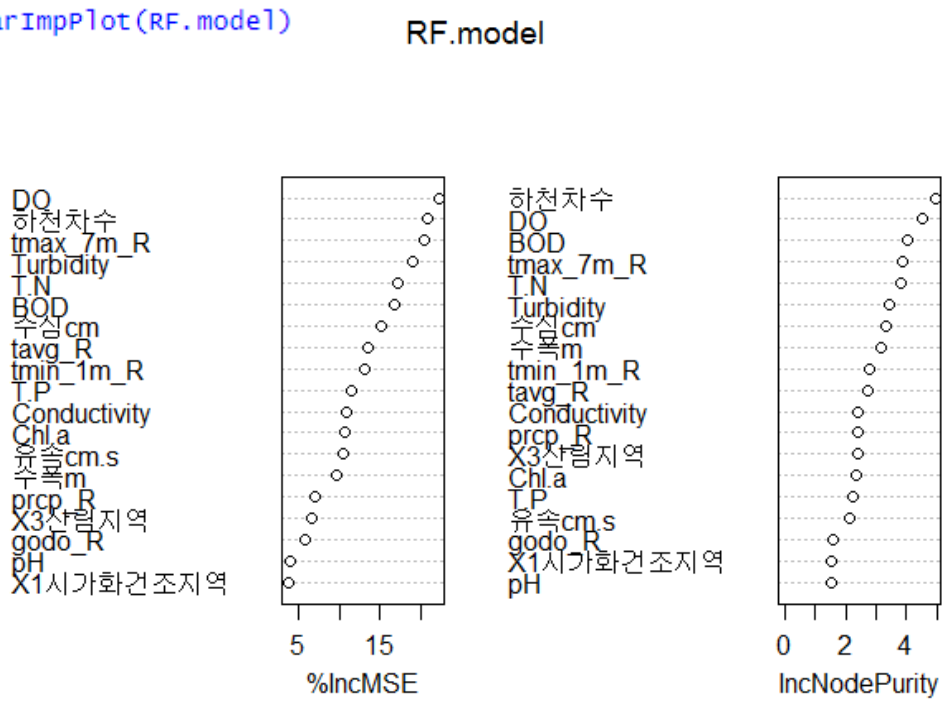
```
#######################################################################
################################### 5. 결과 정리 및 저장하기
#정리하기

# RF.result #서식확률 예측결과
# RF.model$importance[,2] #모델 내 종합된 변수 중요도
# RF.mvalue.df #모델 평가값
```

```
> RF.result
 1001G002  1001G004  1001G006  1001G008  1001G012  1001G014
0.6934833 0.5531000 0.1713833 0.8118667 0.1898500 0.9627833
```

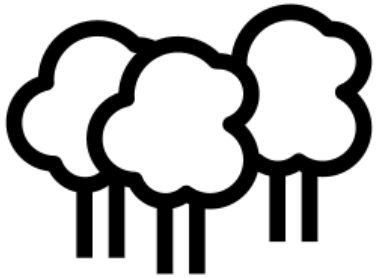
```
> RF.model$importance
                    %IncMSE IncNodePurity
하천차수         0.022804221       4.950085
DO               0.015055230       4.538380
pH               0.001396497       1.509881
Conductivity     0.005857177       2.414623
Turbidity        0.012299074       3.415312
BOD              0.011022229       4.044355
```

```
> RF.mvalue.df
    AUC   ACC
1 0.818 0.773
```

```
> varImpPlot(RF.model)
```



3. Random forest Model



(5) 모델링 결과 정리 및 저장

```
#######################################################################
################################### 5. 결과 정리 및 저장하기
#정리하기

# RF.result #서식확률 예측결과
# RF.model$importance[,2] #모델 내 종합된 변수 중요도
# RF.mvalue.df #모델 평가값
```

```
> RF.model$importance
                    %IncMSE IncNodePurity
하천차수        0.022804221     4.950085
DO              0.015055230     4.538380
pH              0.001396497     1.509881
Conductivity    0.005857177     2.414623
Turbidity       0.012299074     3.415312
BOD             0.011022229     4.044355
```

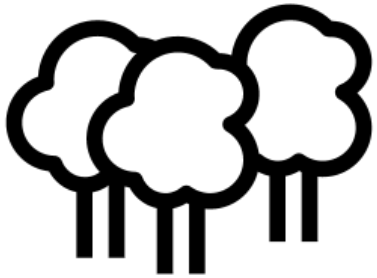
※ RF 모델 내 변수 중요도 평가 방법:

1.정확도(MeanDecreaseAccuracy)

2.노드 불순도 개선(MeanDecreaseGini)

→ 본 모델에서는 '노드 불순도 개선'값을 사용(RF.model$importance[,2]).

3. Random forest Model



(5) 모델링 결과 정리 및 저장

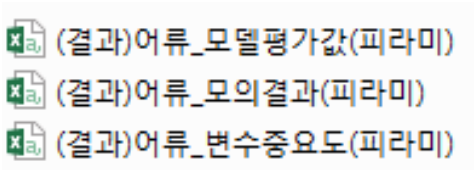
```
#######################################################################
################################### 5. 결과 정리 및 저장하기
#정리하기

# RF.result #서식확률 예측결과
# RF.model$importance[,2] #모델 내 종합된 변수 중요도
# RF.mvalue.df #모델 평가값

#평가 대상 생물
names(Total.DB)[target.sp]
```

```
> #평가 대상 생물
> names(Total.DB)[target.sp]
[1] "피라미"
```

```
#저장하기
temp.result <- as.data.frame(RF.result)
names(temp.result) <- c("출현확률")
write.csv(temp.result, paste0("(결과)어류_모의결과(",names(Total.DB)[target.sp],").csv"))

temp.varimp <- as.data.frame(RF.model$importance[,2])
names(temp.varimp) <- names(Total.DB)[target.sp]
write.csv(temp.varimp, paste0("(결과)어류_변수중요도(",names(Total.DB)[target.sp],").csv"))

temp.eval <- as.data.frame(t(RF.mvalue.df))
names(temp.eval) <- names(Total.DB)[target.sp]
write.csv(temp.eval, paste0("(결과)어류_모델평가값(",names(Total.DB)[target.sp],").csv"))
```

→ write.csv(): csv파일로 저장하기

※ 파일 저장 형태: CSV파일



3. Random forest Model

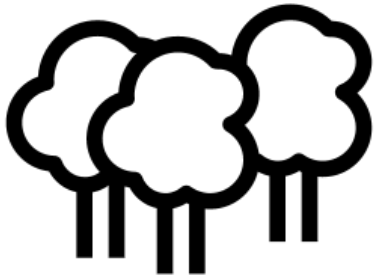
(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
###### 부분의존성 그림 설정
# 부분의존성 그림 내 대상 환경 변수 설정
names(Env.DB) #사용환경명과 동일하게 선택
PDP.variables = c("godo_R","tavg_R","Conductivity", "Turbidity", "BOD", "T.N", "T.P","수폭m", "수심cm", "유속cm.s")

##### PDP 옵션
RF.model
PDP.variables

##### 그림 크기 설정
par(mar=c(4,3,1,1))
par(mfrow=c(3, 4)) # 3x4로 그림 배열
```

```
> names(Env.DB) #사용환경명과 동일하게 선택
 [1] "Total_Code"      "하천차수"        "수온"            "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"
 [9] "NH3.N"           "NO3.N"           "T.N"             "PO4.P"           "T.P"             "Chl.a"           "X1시가화건조지역" "X2농업지역"
[17] "X3산림지역"      "X4초지지역"      "X5습지"          "X6나지"          "X7수역"          "비율"            "slope_R"         "aspect_gra"
[25] "tavg_R"         "tmax_7m_R"       "tmin_1m_R"       "prcp_R"          "godo_R"          "수폭m"           "수심cm"          "유속cm.s"
```

```
> PDP.variables
 [1] "godo_R"       "tavg_R"       "Conductivity" "Turbidity"    "BOD"          "T.N"          "T.P"          "수폭m"        "수심cm"       "유속cm.s"
```

3. Random forest Model



(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

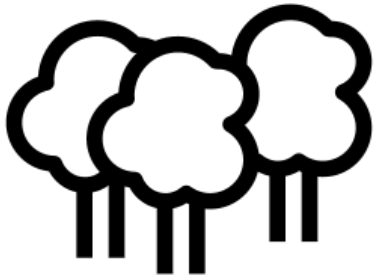
```
##### 주요 환경 변수에 대해 모델 DB(RF.DB) 내 위치(열) 찾기
for(i in 1:length(PDP.variables)){
  if(i == 1){
    PDP.var.num <- (1:length(names(RF.DB[-1])))[names(RF.DB[-1]) == PDP.variables[i]]
  } else {
    PDP.var.num <- c(PDP.var.num, (1:length(names(RF.DB[-1])))[names(RF.DB[-1]) == PDP.variables[i]])
  }
} # for i
PDP.var.num
#변수 확인
names(var.DB)[PDP.var.num] == PDP.variables # 모두 True
```

```
> PDP.var.num
 [1] 12 13  4  5  6  7  8 17 18 19
> #변수 확인
> names(var.DB)[PDP.var.num] == PDP.variables # 모두 True
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

→ DB 내 보고자 하는 변수(PDP.variables)에 해당하는 열 찾기

→ 확인

3. Random forest Model



(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### 그림 내 x축 범위 정하기 (Min, Max)
length(PDP.var.num) #주요 환경 변수 개수
names(var.DB)[PDP.var.num]
x.limit <- list()
range(na.omit(var.DB[,PDP.var.num[1]]))
x.limit[[1]] <- c(0, 800)
range(na.omit(var.DB[,PDP.var.num[2]]))
x.limit[[2]] <- c(5, 15)
#
range(na.omit(var.DB[,PDP.var.num[3]]))
x.limit[[3]] <- c(0,25000)
range(na.omit(var.DB[,PDP.var.num[4]]))
x.limit[[4]] <- c(0,350)
range(na.omit(var.DB[,PDP.var.num[5]]))
x.limit[[5]] <- c(0,12)
range(na.omit(var.DB[,PDP.var.num[6]]))
x.limit[[6]] <- c(0,12)
range(na.omit(var.DB[,PDP.var.num[7]]))
x.limit[[7]] <- c(0,1.2)
range(na.omit(var.DB[,PDP.var.num[8]]))
x.limit[[8]] <- c(0,1200)
range(na.omit(var.DB[,PDP.var.num[9]]))
x.limit[[9]] <- c(0,120)
range(na.omit(var.DB[,PDP.var.num[10]]))
x.limit[[10]] <- c(0,120)
```

```
> names(var.DB)[PDP.var.num]
 [1] "godo_R"       "tavg_R"       "Conductivity" "Turbidity"
> x.limit <- list()
> range(na.omit(var.DB[,PDP.var.num[1]]))
[1]   0 720
> x.limit[[1]] <- c(0, 800)
> range(na.omit(var.DB[,PDP.var.num[2]]))
[1]  6.54779 13.43050
> x.limit[[2]] <- c(5, 15)
```

※ 1번 변수: 고도(godo)

고도 범위: 0 ~ 720 (m)

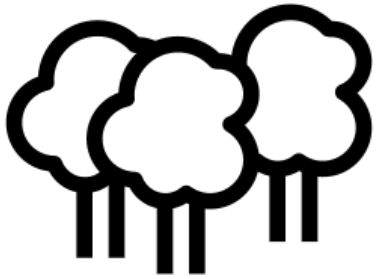
X축 범위 설정: 0 ~ 800 (m)

※ 2번 변수: 평년 평균기온(tavg)

고도 범위: 6.5 ~ 13.4 (℃)

X축 범위 설정: 5 ~ 15 (℃)

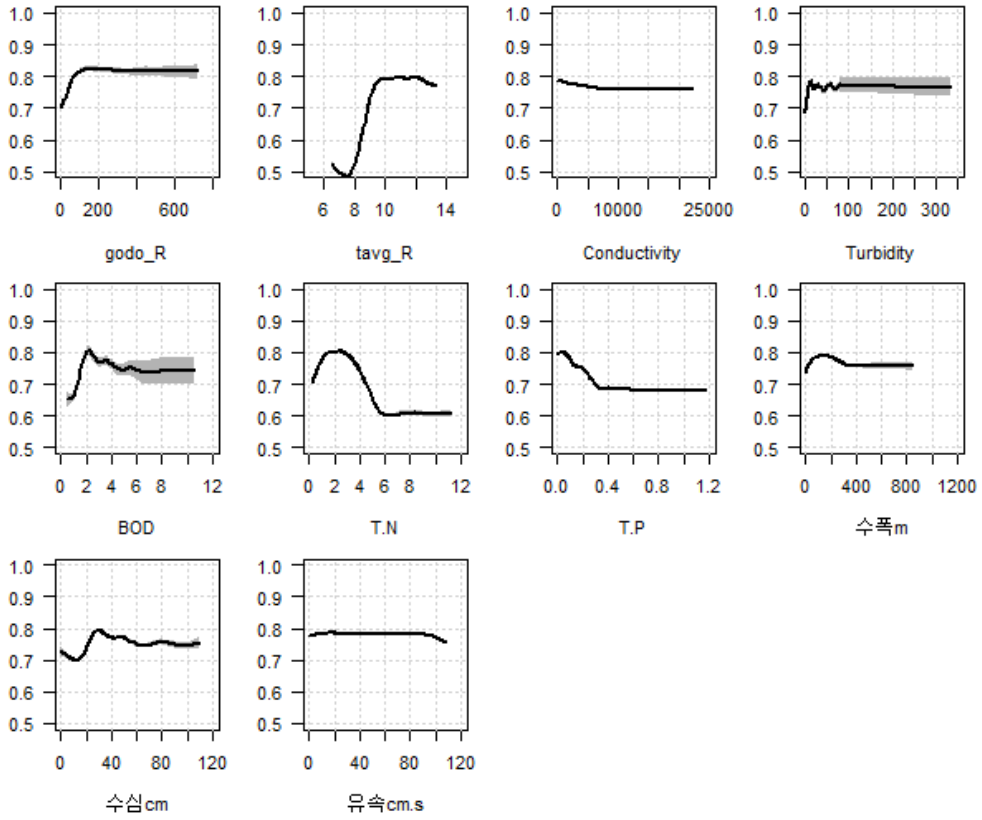
3. Random forest Model



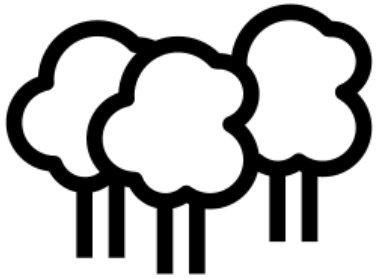
(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### PDP 그리기
for(j in 1:length(PDP.var.num)){ # 1) PD값 구하기
  temp <- model.DB[-1]
  xx <- unique(temp[,PDP.var.num[j]])
  yy <- numeric(length(xx))

  for(num in 1:length(xx)){
    temp[,j] <- xx[num]
    preds <- predict(RF.model, temp)
    yy[num] <- mean(preds)
  }

  # 2) GAM모델 용 DB 제작
  gam.iv.DB <- data.frame("y"=yy, "x"=(xx*var.info[PDP.var.num[j],3]+var.info[PDP.var.num[j],2] ))

  if(length(unique(gam.iv.DB$x))>10){
    gam.iv <- predict(gam(y ~ s(x), data=gam.iv.DB), se=T)
  } else {
    gam.iv <- predict(gam(y ~ s(x, k=length(unique(gam.iv.DB$x))), data=gam.iv.DB), se=T)
  }

  fit <- gam.iv$fit
  se <- gam.iv$se.fit # 신뢰구간
  lcl <- fit - 1.96*se
  ucl <- fit + 1.96*se

  lin.bind <- data.frame("x"=gam.iv.DB$x, fit)
  pol.bind <- data.frame("x"=gam.iv.DB$x, ucl, lcl)

  x.pol <- c(gam.iv.DB[order(gam.iv.DB$x),2],gam.iv.DB[order(gam.iv.DB$x, decreasing=T),2])
  y.pol <- c(pol.bind[order(pol.bind$x),2], pol.bind[order(pol.bind$x, decreasing=T),3])

  ### y축 범위 재설정
  ylimit <- c(0.2, 0.8)

  # 3) PDP 그리기
  plot(c(0.5,0.5), xlim=c(x.limit[[j]][1],
                          x.limit[[j]][2]),
       ylim=ylimit, type="n", yaxt="n", xlab=PDP.variables[j], ylab="")
  grid()
  polygon(x.pol, y.pol, col="gray70",border=NA) #"black")
  lines(lin.bind[order(lin.bind$x),], col="black", lwd=2)
  axis(2, las=2)
}# for j
```

→ for 함수를 통해 대상 변수 plot을 한 번에 그려줌



3. Random forest Model



(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### PDP 그리기
for(j in 1:length(PDP.var.num)){ # 1) PD값 구하기
  temp <- model.DB[-1]
  xx <- unique(temp[,PDP.var.num[j]])
  yy <- numeric(length(xx))

  for(num in 1:length(xx)){
    temp[,j] <- xx[num]
    preds <- predict(RF.model, temp)
    yy[num] <- mean(preds)
  }

  # 2) GAM모델 용 DB 제작
  gam.iv.DB <- data.frame("y"=yy, "x"=(xx*var.info[PDP.var.num[j],3]+var.info[PDP.var.num[j],2] ))

  if(length(unique(gam.iv.DB$x))>10){
    gam.iv <- predict(gam(y ~ s(x), data=gam.iv.DB), se=T)
  } else {
    gam.iv <- predict(gam(y ~ s(x, k=length(unique(gam.iv.DB$x))), data=gam.iv.DB), se=T)
  }
```

※ Algorithm :

모델 DB에서 대상 변수(PDP.variables)의 값을
대상 변수의 고유값(unique)으로 반복적으로 치환한 후
(for num), 모델 예측 결과값의 변화를 확인함

→ 'GAM 모델'을 통해 부분의존성 효과에 대한 모델 결과값을 fitting해줌(추세선)

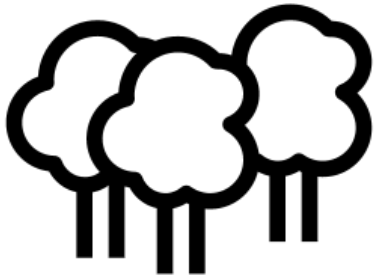
※ 주의사항: 'model.DB'객체는 환경 변수가 표준화(scale)가 된 값이기에,

표준화 이전 값으로 반환해주고 fitting을 진행함 (var.info 이용)

※ 표준화 z = (X – mean)/sd → X = z(≒ Model.DB값) * sd + mean

```
> var.info
              name          mean           sd
1          하천차수    5.51034483 1.983052e+00
2               DO    9.41471264 1.862191e+00
3               pH    7.89172414 5.201697e-01
4      Conductivity  298.82321839 1.131712e+03
5         Turbidity   10.09839080 2.004259e+01
6               BOD    1.93471264 1.110015e+00
7               T.N    2.38296552 1.217586e+00
8               T.P    0.06201379 9.026808e-02
9             Chl.a    3.72850575 6.416491e+00
10 X1시가화건조지역   11.03603908 1.325349e+01
11       X3산림지역   26.55498621 2.354068e+01
12           godo_R  104.70679977 1.168564e+02
13           tavg_R   10.95712149 1.350842e+00
14         tmax_7m_R   27.84787563 9.843470e-01
15         tmin_1m_R   -8.42283251 2.617238e+00
16            prcp_R 1232.35022299 1.018252e+02
17            수폭m   85.40620690 1.182695e+02
18            수심cm   30.60091954 1.783805e+01
19          유속cm.s   25.47241379 2.329885e+01
```

3. Random forest Model



(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
fit <- gam.iv$fit
se <- gam.iv$se.fit # 신뢰구간
lcl <- fit - 1.96*se
ucl <- fit + 1.96*se

lin.bind <- data.frame("x"=gam.iv.DB$x, fit)
pol.bind <- data.frame("x"=gam.iv.DB$x, ucl, lcl)

x.pol <- c(gam.iv.DB[order(gam.iv.DB$x),2],gam.iv.DB[order(gam.iv.DB$x, decreasing=T),2])
y.pol <- c(pol.bind[order(pol.bind$x),2], pol.bind[order(pol.bind$x, decreasing=T),3])

### y축 범위 재설정
ylimit <- c(0.2, 0.8)

# 3) PDP 그리기
plot(c(0.5,0.5), xlim=c(x.limit[[j]][1],
                        x.limit[[j]][2]),
     ylim=ylimit, type="n", yaxt="n", xlab=PDP.variables[j], ylab="")
grid()
polygon(x.pol, y.pol, col="gray70",border=NA) #"black")
lines(lin.bind[order(lin.bind$x),], col="black", lwd=2)
axis(2, las=2)
}# for j
```

→ 신뢰구간(95%, 1.96)은 다각형(polygon으로 표현함)

→ y축 범위: 설정 가능함 (e.g. 0.5~1.0)

※ 실질적으로 그림을 그리는 함수부분

순서: 1. Plot(): 배경, 범위 지정

(type="n"으로 내용을 표시하지 않음, yaxt="n"로 y축 표시하지 않음)

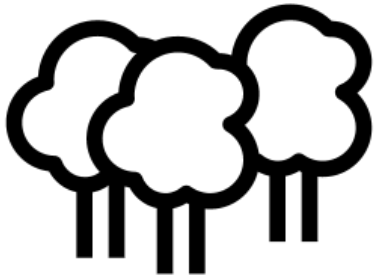
2. grid(): 배경 격자

3. polygon(): 신뢰구간 표시

4. lines(): GAM모델을 이용한 추세선 표시

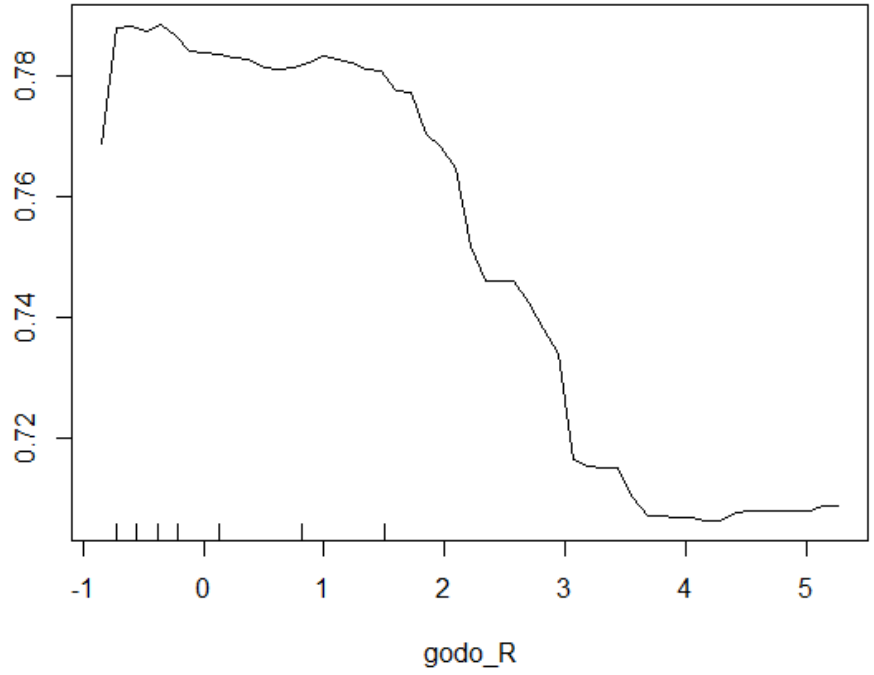
5. axis(2): y축 표시

3. Random forest Model



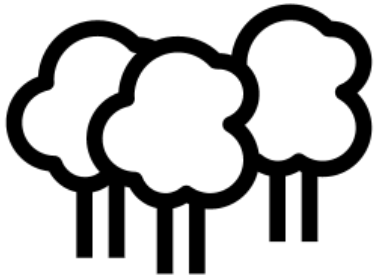
(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
partialPlot(RF.model, train_DB, PDP.variables[1], plot=T)
```



※ 'randomForest' package 내 PDP에 대한 함수가 존재함:
partialPlot()
→단점: 산출 시간이 오래걸리며, scale 재조정이 힘듦

3. Random forest Model



(7) 신규 자료를 이용한 확률 예측

```
#######################################################################
################################### 7. 모델을 이용한 서식 확률 예측 (신규자료 바탕)

##### 신규자료 불러오기
# new.env <- read.csv("수생태_어류_환경.csv", stringsAsFactors=F)
new.env <- na.omit(var.DB)

##### 표준화
## 기존 모델 내 변수 순서와 신규 자료 내 변수 순서 맞춤
for(k in 1:length(names(new.env))){
  new.num <-(1:length(names(new.env)))[var.info[,1]== names(new.env)[k]]
  if(k == 1){
    new.num2 <- new.num
  } else{
    new.num2 <- c(new.num2, new.num)
  }
}
names(new.env)[new.num2] # 모델 내 변수 순서로 정렬
order.env <- new.env[new.num2]

## scale(표준화) : z = (Mean - X)/sd
scaled.env <- order.env
for(j in 1:length(new.num2)){
  scaled.env[,j] <- (order.env[,j] - var.info[j,2])/(var.info[j,3])
}

##### 모델 예측
RF.pred <- predict(RF.model, scaled.env)
pred.result <- data.frame("서식확률"=RF.pred)

# 저장하기
write.csv(pred.result, "(R결과)어류_RF_신규모의결과.csv")
```

※ **신규자료**: 새로운 지점의 환경자료를 의미

※ **모델링과 동일 과정을 거침**

→ NA자료 제거, 환경변수 표준화

※ **모의**: predict()함수 이용

국립환경과학원

경희대학교 KYUNG HEE UNIVERSITY

THE UNIVERSITY OF SEOUL 서울시립대학교

EnFRA

수생태계모델연구 컨소시엄

감사합니다

2018-2019년에 수행된 국립환경과학원 연구 과제 :

차세대 수질. 수생태계 예측모델 개발 및 적용성 평가(II)

- 국내 수질. 수생태계 모니터링 자료를 활용한 수생태 모델 활용성 평가”로 수행된 연구 결과의 일부임

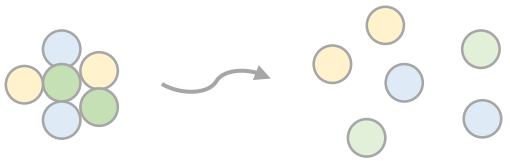
MaxEnt 모델 사용 설명서

이대성, 박영석
경희대학교 생물학과
dleotjd520@naver.com, parkys@khu.ac.kr

순서

1. 최대 엔트로피 모델 (Maximum entropy model)
2. 종분포에 대한 MaxEnt 모델
3. MaxEnt model 적용

1. 최대 엔트로피모델이란?



1. 엔트로피(Entropy)

- 무질서도로 알려진 엔트로피(Entropy)는 어떤 확률 변수의 불확실성을 의미한다(정보량).

※ 정보의 정량화: 사건의 예측이 가능하지 않을수록(≒불확실성이 높을 수록), 정보량이 많아집니다
반대로, 사건이 예측 가능해질 때, 정보의 가치가 없게 됩니다.
→ 즉, 사건의 발생확률이 작을수록 정보의 가치가 높아집니다.

(1) 정보량 $h(x) = -\log(p(xi))$

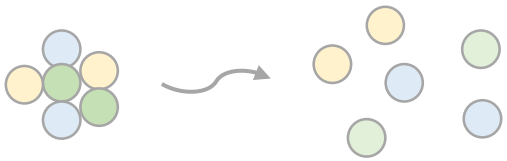
※ N 종류의 사건 i=1, 2, 3 ~ N
※ 사건이 발생할 확률 $p(xi)$

→ 어떤 사건의 발생 확률에 대하여, 정보량은 확률이 1에 가까워질수록 0으로 수렴합니다.

(2) 엔트로피 $H(x) = -\sum_{i=1}^{N} p(xi)\log(p(xi))$

→ 전체 상태(확률 분포)에 대한 정보량의 기대치입니다. N개의 상태가 각각 1/N의 발생확률을 가질 때 최대가 됩니다.
즉, 확률분포에서 특정한 값이 나올 확률에 따라 엔트로피 값이 변화합니다.

1. 최대 엔트로피모델이란?



2. 최대 엔트로피(Maximum Entropy; MaxEnt)

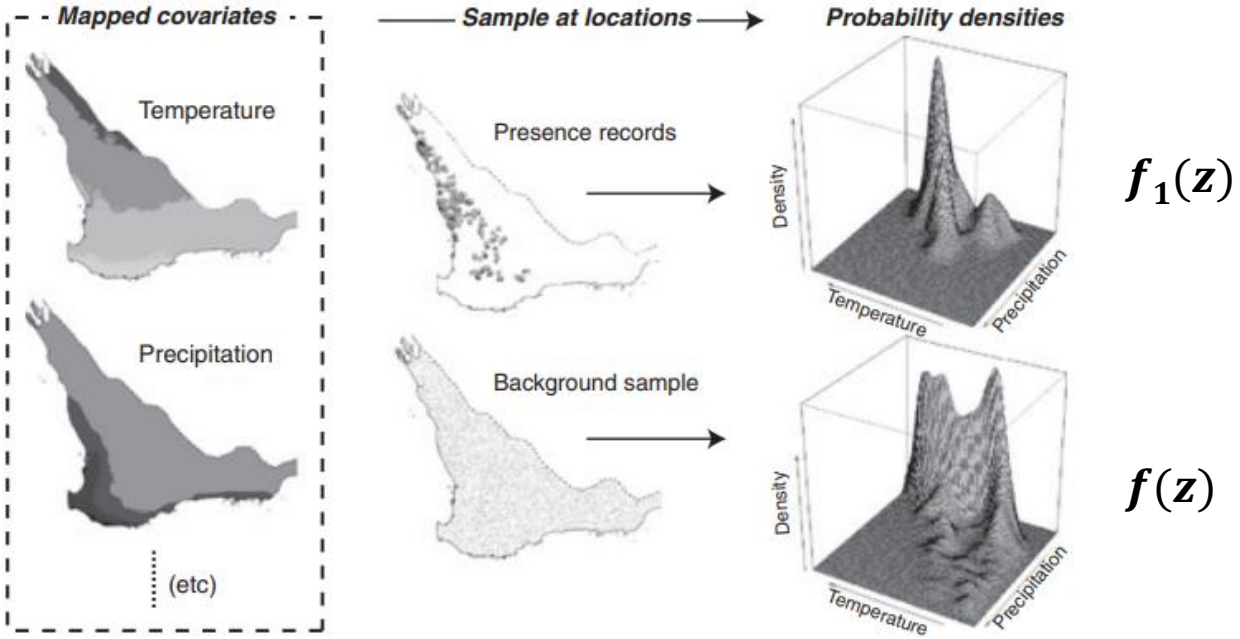
- Maximum likelihood 계산과 유사하게, Entropy가 최대가 될 때, 사건의 최적값을 의미합니다.
라그랑주 승주법(Lagrange Multiplier) 등을 이용하여 최적값을 계산할 수 있습니다.

※ 최대 엔트로피를 가지는 확률분포 $\mathrm{p}(x) = \mathrm{argmax}\, H(x)$

2. 종 분포에 대한 MaxEnt 모델

- 환경 및 지리적 좌표화된(georeferenced) 출현 지점 자료로부터, 최대 엔트로피 방법이 적용된 MaxEnt모델은 지점에 대해 출현 종에 대한 조건의 예측 된 적합성을 갖는 확률 분포를 표현합니다.

- MaxEnt 모델은 알려진 종 출현 위치(Presence site)와 배경 지점(Background location)에 환경 데이터를 사용합니다.



※ $f1(z)/f(z)$ 비율에 대한 공분산 추정 후, 파라미터(τ) 추정을 통해 주어진 환경 내 종 출현 확률($\Pr(y = 1|z)$)을 계산합니다.

Figure 1 A diagrammatic representation of the probability densities relevant to our statistical explanation
Elith et al. (2011). A statistical explanation of MaxEnt for ecologists. *Diversity and distributions*, *17*(1), 43-57.

3. MaxEnt Model

(1) 자료 확인

1. 어류 대표종 개체수 자료(Abundance)

- 어류 각 대표종 자료를 사용하여 작성합니다('수생태_어류_생물'파일).
- 앞 1개열은 조사 지점정보, 그 뒷열부터는 각 대표 생물 종에 해당하며, 행은 각 지점을 의미합니다.
- 각 대표종 열에 들어간 값은 각 어류 대표종의 개체수를 의미합니다.
- 입력에 필요한 대상 생물종수의 제한은 없으며, 2번째 열 이후로만 지정해주면 됩니다.

	A	B	C	D	E	F	G	H	I
1	Total_Code	피라미	참갈겨니	돌고기	배스	블루길	감돌고기	묵납자루	어름치
2	1001G002	34.5	41.9375	3.625	0	0	0	0	0
3	1001G004	27.75	42.75	3.0625	0	0	0	0	0
4	1001G006	0	48	4.5	0	0	0	0	0
5	1001G008	91.5	145	64	0	0	0	22.5	5
6	1001G012	0	69	16.5	0	0	0	0	0
7	1001G014	16.875	52.625	11.5	0	0	0	4.625	1.6875

3. MaxEnt Model

(1) 자료 확인

2. 환경자료

- 모델에 사용할 환경자료입니다. 행은 지점, 열은 환경 변수를 의미합니다.
- 환경 변수의 종류는 상관없으나, 자료의 형태는 수치형자료(예, 1, 2.1, 120.2 등)이어야 합니다.
- 입력한 환경 변수 중 실제 모델 제작에 사용할 환경 변수는 차후 사용자가 직접 설정합니다.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Total_Cod	하천차수	수온	DO	pH	Conductiv	Turbidity	BOD	NH3-N	NO3-N	T-N	PO4-P	T-P	Chl-a	1시가화건	2농업지역
2	1001G002	5	15.4	11.8	8.4	214.1	0.6	0.7	0.032	1.697	2.52	0.008	0.02	0.6	1.072	24.129
3	1001G004	5	15.2	9.8	8.1	57.3	0.8	0.7	0.011	0.771	1.44	0.004	0.01	0.8	2.353	18.593
4	1001G006	4	17.4	17.6	7.7	33.5	1.8	1.3	0.106	1.303	1.736	0.026	0.038	2.1	0.372	5.597
5	1001G008	6	23.1	6.7	8.1	263.5	2.6	1.9	0.041	2.599	2.828	0.015	0.024	1.3	3.941	63.812
6	1001G012	4	16.6	13.2	7.7	194	66	1.3	0.124	1.061	3.704	0.045	0.09	2.5	3.214	52.682
7	1001G014	6	19.4	11.1	8.4	170.9	1.7	0.8	0.02	1.337	2.315	0.007	0.017	1.2	17.749	35.215
8	1001G024	5	17.9	10.7	8.4	116.1	2.5	0.8	0.032	1.52	2.356	0.008	0.02	1.1	3.673	41.396
9	1001G026	6	17.6	10.2	8	151.4	2.4	0.9	0.03	1.339	2.263	0.006	0.015	1	8.641	47.352
10	1001G036	5	18.2	11.2	8.3	120.3	3	0.9	0.029	1.144	2.144	0.006	0.018	1.1	7.199	32.818
11	1001G038	6	18.9	11.8	8.7	151.3	6.3	0.9	0.022	1.089	2.056	0.004	0.012	1	47.169	0.138

3. MaxEnt Model

(1) 자료 확인

3. MaxEnt 모델 Software in R

- MaxEnt 모델을 R에서 사용하기 위해서는 MaxEnt software 홈페이지에서 MaxEnt 파일(.jar)을 받아 C드라이브 내 R 폴더(dismo package 폴더)에 위치시켜야 합니다.
- 파일의 저장 경로는 R에서 ‘system.file(" java " , package= " dismo ")’함수를 실행하여 알 수 있습니다.

※ **홈페이지**: https://biodiversityinformatics.amnh.org/open_source/maxent/

3. MaxEnt Model

(2) 자료 불러오기

※ **'검정색'**: R script창 내 코드

※ **'>파란색'**: R console 내 결과물

```
#######################################################################
################################### 1. 자료 불러오기
## 풍부도자료(Abundance): 개체수
sp.DB <- read.csv("수생태_어류_생물.csv", stringsAsFactors=F)
str(sp.DB)

## 환경 자료
Env.DB <- read.csv("수생태_어류_환경.csv", stringsAsFactors=F)
str(Env.DB)
```

→ 문자가 함께 혼용되어 있는 자료 불러오기

→ 자료 확인 함수: str(), head(), tail()

```
> str(sp.DB)
'data.frame':   459 obs. of  9 variables:
 $ Total_Code: chr  "1001G002" "1001G004" "1001G006" "1001G008" ...
 $ 피라미    : num  34.5 27.8 0 91.5 0 ...
 $ 참갈겨니  : num  41.9 42.8 48 145 69 ...
 $ 돌고기    : num  3.62 3.06 4.5 64 16.5 ...
 $ 배스      : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 블루길    : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 감돌고기  : num  0 0 0 0 0 0 0 0 0 0 0 ...
 $ 묵납자루  : num  0 0 0 22.5 0 ...
 $ 어름치    : num  0 0 0 5 0 ...
```

```
> str(Env.DB)
'data.frame':   459 obs. of  32 variables:
 $ Total_Code      : chr  "1001G002" "1001G004" "1001G006" "1001G008" ...
 $ 하천차수        : int  5 5 4 6 4 6 5 6 5 6 ...
 $ 수온            : num  15.4 15.2 17.4 23.1 16.6 19.4 17.9 17.6 18.2 18.9 ...
 $ DO              : num  11.8 9.8 17.6 6.7 13.2 11.1 10.7 10.2 11.2 11.8 ...
 $ pH              : num  8.4 8.1 7.7 8.1 7.7 8.4 8.4 8 8.3 8.7 ...
 $ Conductivity    : num  214.1 57.3 33.5 263.5 194 ...
 $ Turbidity       : num  0.6 0.8 1.8 2.6 66 1.7 2.5 2.4 3 6.3 ...
 $ BOD             : num  0.7 0.7 1.3 1.9 1.3 0.8 0.8 0.9 0.9 0.9 ...
 $ NH3.N           : num  0.032 0.011 0.106 0.041 0.124 0.02 0.032 0.03 0.029 0.022 ...
 $ NO3.N           : num  1.697 0.771 1.303 2.599 1.061 ...
 $ T.N             : num  2.52 1.44 1.74 2.83 3.7 ...
 $ PO4.P           : num  0.008 0.004 0.026 0.015 0.045 0.007 0.008 0.006 0.006 0.004 ...
 $ T.P             : num  0.02 0.01 0.038 0.024 0.09 0.017 0.02 0.015 0.018 0.012 ...
 $ Chl.a           : num  0.6 0.8 2.1 1.3 2.5 1.2 1.1 1 1.1 1 ...
 $ X1시가화건조지역: num  1.072 2.353 0.372 3.941 3.214 ...
 $ X2농업지역      : num  24.1 18.6 5.6 63.8 52.7 ...
 $ X3산림지역      : num  69.04 37.53 86.03 9.41 30.63 ...
 $ X4초지지역      : num  0.412 11.575 0 5.892 0.288 ...
 $ X5습지          : num  1.826 0.151 5.36 5.514 10.217 ...
 $ X6나지          : num  0.714 1.632 1.092 0 0 ...
 $ X7수역          : num  2.81 28.16 1.55 11.43 2.97 ...
 $ 비율            : num  100 100 100 100 100 ...
 $ slope_R         : num  0.324 6.204 6.066 0 0 ...
 $ aspect_gra      : int  8 1 6 0 0 0 0 0 7 5 ...
 $ tavg_R          : num  6.55 7.29 6.75 7.83 7.46 ...
 $ tmax_7m_R       : num  24.3 24.3 24.3 24.8 24.3 ...
 $ tmin_1m_R       : num  -13.1 -12.1 -13 -12.3 -12.4 ...
 $ prcp_R          : num  1222 1186 1203 1160 1172 ...
 $ godo_R          : num  720 680 680 500 520 ...
 $ 수폭m           : num  14.5 10.1 8.3 15 6.3 ...
 $ 수심cm          : num  17 20.3 33.9 25 31.9 21.8 20.9 25.7 19.4 22.7 ...
 $ 유속cm.s        : num  48.2 68.5 72.4 14 91.2 55.4 72.9 78.1 60.7 60.5 ...
```

3. MaxEnt Model

(2) 자료 정리하기

※ **'검정색'**: R script창 내 코드
※ **'>파란색'**: R console 내 결과물

```
##### 생물 자료(풍부도) 확인 및 DB생성
names(sp.DB)
```

```
> names(sp.DB)
[1] "Total_Code" "피라미"     "참갈겨니"    "돌고기"     "배스"       "블루길"     "감돌고기"    "묵납자루"    "어름치"
```

※ 지점 + 8종

```
names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
```

```
> names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
[1] "피라미"   "참갈겨니" "돌고기"   "배스"     "블루길"   "감돌고기" "묵납자루" "어름치"
```

```
used.sp.DB <- sp.DB[2:length(names(sp.DB))]
```

→ 생물 종 DB 생성 (피라미~어름치)

```
# 지점 명칭 부여(DB 행)
rownames(used.sp.DB)
rownames(used.sp.DB) <- sp.DB[,1]
```

```
> rownames(used.sp.DB)
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"
```

→ 'rowname' = DB 행 의미(지점)
만들어진 생물 종 DB에 지점정보(sp.DB[,1])를 반영해줌

```
> used.sp.DB
        피라미     참갈겨니
1   34.5000000  41.9375000
2   27.7500000  42.7500000
3    0.0000000  48.0000000
4   91.5000000 145.0000000
5    0.0000000  69.0000000
6   16.8750000  52.6250000
```

```
rownames(used.sp.DB)
[1] "1001G002" "1001G004" "1001G006" "1001G008" "1001G012"
```

3. MaxEnt Model

(2) 자료 정리하기

```
##### 환경 자료확인
names(Env.DB)
```

```
> names(Env.DB)
 [1] "Total_Code"      "하천차수"        "수온"            "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"
 [9] "NH3.N"           "NO3.N"           "T.N"             "PO4.P"           "T.P"             "Chl.a"           "X1시가화건조지역" "X2농업지역"
[17] "X3산림지역"      "X4초지지역"      "X5습지"          "X6나지"          "X7수역"          "비율"            "slope_R"         "aspect_gra"
[25] "tavg_R"         "tmax_7m_R"       "tmin_1m_R"       "prcp_R"          "godo_R"          "수폭m"           "수심cm"          "유속cm.s"
```

```
names(Env.DB)[1] #조사 지점 정보
names(Env.DB)[2] #하천차수
names(Env.DB)[3] #수온
names(Env.DB)[4:14] #수질1
names(Env.DB)[15:21] #토지피복
names(Env.DB)[23] #경사각
names(Env.DB)[29] #고도
names(Env.DB)[25:28] #기상조건
names(Env.DB)[30:32] #수리수문
```

```
##### 사용할 환경변수 지정
var.DB <- Env.DB[c(2,#하천차수
                   4:8,11,13:14,#수질- DO, pH, Conductivity, Turbidity, BOD, / T-N, / T-P, Chl-a
                   15,17,#토지피복- 1시가화건조지역, 3산림지역
                   29,#고도
                   25:28,#기상조건
                   30:32#수리수문- 수폭,수심,유속
)]
```

→ 환경 변수 중 일부만 사용

3. MaxEnt Model

(2) 자료 정리하기

```
names(var.DB) #사용 환경변수 명칭
length(names(var.DB)) #사용 환경변수 개수
```

```
> names(var.DB) #사용 환경변수 명칭
 [1] "하천차수"        "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"             "T.N"             "T.P"
 [9] "Chl.a"           "X1시가화건조지역" "X3산림지역"      "godo_R"          "tavg_R"          "tmax_7m_R"       "tmin_1m_R"       "prcp_R"
[17] "수폭m"           "수심cm"          "유속cm.s"
> length(names(var.DB)) #사용 환경변수 개수
[1] 19
```

```
Total.DB <- cbind(used.sp.DB, var.DB)
```

→ 환경 + 생물 DB

3. MaxEnt Model

(3) 모델 설정

```
#######################################################################
################################### 2. 모델 설정
##### 어류
# 사용모델: MaxEnt
# 자료형태: P/A

##### 필요 package
# install.packages(dismo)
# install.packages(ROCR)
# install.packages(caret)
# install.packages(mgcv)
library(dismo)
#system.file("java", package="dismo")
library(ROCR)
library(caret)
library(mgcv)
```

※ MaxEnt model용 package

→ dismo

※ 모델 평가용 package

→ ROCR, caret

※ 부분의존성 그림(PDP)용 package

→ mgcv

```
###### 모의 생물 종 선택
#모델을 제작하고자 하는 생물 종 선택
#형성된 자료의 열을 기준으로 생성
target.sp = 1 #

names(used.sp.DB)[target.sp] #대상 생물 확인
```

※ 8종의 생물 중 모의 대상 종 선택

```
> names(used.sp.DB)[target.sp] #대상 생물 확인
[1] "피라미"
```

```
> names(sp.DB)[2:length(names(sp.DB))] #실제 생물 열
[1] "피라미"   "참갈겨니" "돌고기"   "배스"     "블루길"   "감돌고기" "묵납자루" "어름치"
```

(1) ~ **(8)**

3. MaxEnt Model

(4) 자료의 전처리

```
#######################################################################
#################################### 3. 자료의 전처리 과정
targetDB <- Total.DB
total.num.sp <- dim(used.sp.DB)[2] # DB 내 포함된 생물종 수
```

= 8종

생물

환경

```
> names(targetDB)
 [1] "피라미"          "참갈겨니"        "돌고기"          "배스"            "블루길"          "감돌고기"        "묵납자루"        "어름치"
 [9] "하천차수"        "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"             "T.N"             "T.P"
[17] "Chl.a"           "X1시가화건조지역" "X3산림지역"      "godo_R"          "tavg_R"          "tmax_7m_R"       "tmin_1m_R"       "prcp_R"
[25] "수폭m"           "수심cm"          "유속cm.s"
```

```
### 전처리 과정
#1. 사용 Data만 선택
m1 <- targetDB[,c(target.sp,(1+total.num.sp):dim(targetDB)[2])]
print(colnames(m1)[1]) #모의 생물 종 명칭
```

1 1+8 27

※ DB 중 선택 생물 열(target.sp) + 환경 열

→ 피라미

```
#2. 자료 내 NA 제거
m2 <- na.omit(m1)
names(m2)[1] <- "pop" #모의를 위한 종명칭 변경
```

→ 환경자료 내 자료 NA 지점 제거

```
#3. 생물 자료 정수화
m3 <- m2
m3[,1] <- round(m3[,1],0) # 정수화
```

→ 정수화 예시: 피라미 1.1마리 -> 1마리

3. MaxEnt Model

(4) 자료의 전처리

```
#4. 출현여부(P/A) 변환 (1/0)
m4 <- m3
for(i in 1:length(m4[,1])){
  if(m3[i,1] > 0){
    m4[i,1] <- 1
  } else
    m4[i,1] <- 0
}# for i
```

→변환(생물): 생물 개체수(정수) → 출현여부(0, 1)

※ [,1]: 1번 열 = 생물(≒피라미)열

※ length(m4[,1]): 1번 열(생물)의 전체 행 개수(≒ 지점 수)

```
#5. 환경 변수 표준화(scale))
m5 <- m4
#str(e1)
m5[,2:(dim(m4)[2])] <- scale(m4[,2:(dim(m4)[2])]) # 표준화: z= (Mean - X)/sd
```

→변환(환경): 환경열을 표준화(Standadization)함

e.g. DB[1] ≒ dim(DB)[2]

length(DB[,1]) ≒ dim(DB)[1]

(No)	피라미
1	1
2	0
3	1
4	1

```
#6. 환경 변수 정보(var.info) 저장
target.env.DB <- m4[,2:(dim(m4)[2])] # scale 하기 바로 직전 DB
total.var.name <- names(target.env.DB)
```

= m4

```
### 표준화 정보
var.info <- as.data.frame(matrix(NA, nrow=length(total.var.name), ncol=3))
names(var.info) <- c("name", "mean", "sd")
for( i in 1:length(total.var.name)){
  var.info[i,1] <- total.var.name[i]
  var.info[i,2] <-mean(target.env.DB[,i], na.rm=T)
  var.info[i,3] <-sd(target.env.DB[,i], na.rm=T)
}
```

```
> var.info
                name          mean           sd
1           하천차수    5.51034483 1.983052e+00
2                DO    9.41471264 1.862191e+00
3                pH    7.89172414 5.201697e-01
4      Conductivity  298.82321839 1.131712e+03
5         Turbidity   10.09839080 2.004259e+01
6               BOD    1.93471264 1.110015e+00
7               T.N    2.38296552 1.217586e+00
8               T.P    0.06201379 9.026808e-02
9             Chl.a    3.72850575 6.416491e+00
10 X1시가화건조지역   11.03603908 1.325349e+01
11       X3산림지역   26.55498621 2.354068e+01
12           godo_R  104.70679977 1.168564e+02
13           tavg_R   10.95712149 1.350842e+00
14        tmax_7m_R   27.84787563 9.843470e-01
15        tmin_1m_R   -8.42283251 2.617238e+00
16           prcp_R 1232.35022299 1.018252e+02
17            수폭m   85.40620690 1.182695e+02
18           수심cm   30.60091954 1.783805e+01
19         유속cm.s   25.47241379 2.329885e+01
```

3. MaxEnt Model

(5) 모델링

```
#######################################################################
################################### 4. 모델링
# 모델용 DB
model.DB <-  m5 = 변환 완료 DB

# model.DB 구성
dim(model.DB) # 20개 변수(pop + 19개), 435개 지점(site)

# 사용 DB 내 환경변수 열 지정 → 2:20
var.num <- c(2:dim(m5)[2])
names(model.DB)[var.num] # 사용 환경 변수명

##### 훈련(Training) 및 검증(Test) 자료 분할
DB_P <- subset(model.DB, model.DB$pop > 0) #출현 DB
DB_A <- subset(model.DB, model.DB$pop == 0) #비출현 DB

# 분할(훈련:검증 = 8:2)
set.seed(6808)
no.p <- sample(1:dim(DB_P)[1], dim(DB_P)[1]*0.8, replace=F)
no.a <- sample(1:dim(DB_A)[1], dim(DB_A)[1]*0.8, replace=F)

train_DB <- rbind(DB_P[no.p,], DB_A[no.a,])     → 80%
test_DB <-  rbind(DB_P[-no.p,], DB_A[-no.a,])   → 20%
dim(train_DB)
dim(test_DB)
```

```
> dim(model.DB) # 20개 변수(pop + 19개), 435개 지점(site)
[1] 435  20
```

```
> names(model.DB)[var.num] # 사용 환경 변수명
 [1] "하천차수"        "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"             "T.N"             "T.P"
 [9] "Chl.a"           "X1시가화건조지역" "X3산림지역"       "godo_R"          "tavg_R"          "tmax_7m_R"       "tmin_1m_R"       "prcp_R"
[17] "수폭m"           "수심cm"          "유속cm.s"
```

※ set.seed(): random sample 패턴 고정 → 재현 가능
※ replace=F: 복원추출 방지

3. MaxEnt Model

(5) 모델링: 훈련

```
##### MaxEnt model 제작
Mx.DB <- train_DB[c(1,var.num)]
Mx.test_DB <- test_DB

set.seed(6808)
Mx.model <-  maxent(Mx.DB[c(var.num)],Mx.DB[,1])
```

※ 환경자료(x), 생물자료(P) 순

※ 도움말 (?maxent)

Usage

```
"maxent"(x, p, a=NULL, factors=NULL, removeDuplicates=TRUE, nbg=10000, ...)
"maxent"(x, p, a=NULL, removeDuplicates=TRUE, nbg=10000, ...)
"maxent"(x, p, args=NULL, path, silent=FALSE, ...)
"maxent"(x, p, silent=FALSE, ...)
```

x Predictors. Raster* object or SpatialGridDataFrame, containing grids with predictor variables. These will be used to extract values from for the point locations. `x` can also be a data.frame, in which case each column should be a predictor variable and each row a presence or background record

p Occurrence data. This can be a data.frame, matrix, SpatialPoints* object, or a vector. If `p` is a data.frame or matrix it represents a set of point locations; and it must have two columns with the first being the x-coordinate (longitude) and the second the y-coordinate (latitude). Coordinates can also be specified with a SpatialPoints* object

If `x` is a data.frame, `p` should be a vector with a length equal to `nrow(x)` and contain 0 (background) and 1 (presence) values, to indicate which records (rows) in data.frame `x` are presence records, and which are background records

a Background points. Only used if `p` is and not a vector and not missing

3. MaxEnt Model

(5) 모델링: 훈련

```
##### MaxEnt model 제작
Mx.DB <- train_DB[c(1,var.num)]
Mx.test_DB <- test_DB

set.seed(6808)
Mx.model <-  maxent(Mx.DB[c(var.num)],Mx.DB[,1])
```

※ 환경자료(x), 생물자료(P) 순

```
> Mx.model
class    : MaxEnt
variables: 하천차수 DO pH Conductivity Turbidity BOD T.N T.P Chl.a X1시가화건조지역
 tmax_7m_R tmin_1m_R prcp_R 수폭m 수심cm 유속cm.s
```

→ 해당 Model 실행 시, 외부 홈페이지를 통해 model 정보를 알 수 있음(뒷장)

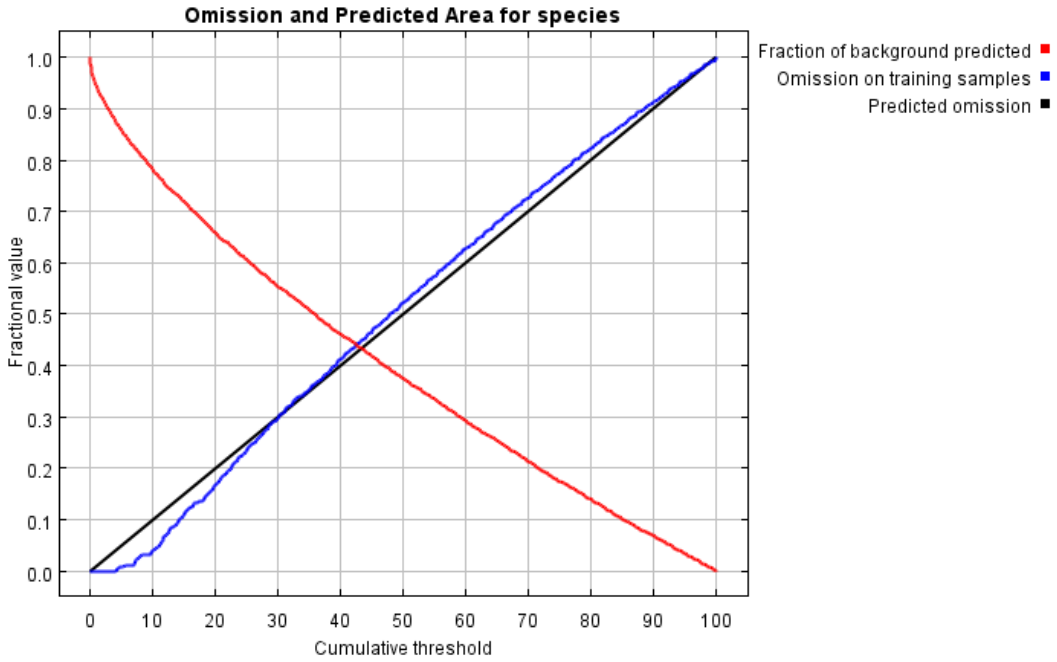
localhost:29965/session/raster/maxent/7189175339/maxent.html

Maxent model

This page contains some analysis of the Maxent model result, created Wed Dec 18 06:06:35 KST 2019 using 'dismo' version 1.1-4 & Maxent version 3.4.1. If you would like to do further analyses, the raw data used here is linked to at the end of this page.

Analysis of omission/commission

The following picture shows the omission rate and predicted area as a function of the cumulative threshold. The omission rate is is calculated both on the training presence records, and (if test data are used) on the test records. The omission rate should be close to the predicted omission, because of the definition of the cumulative threshold.



The next picture is the receiver operating characteristic (ROC) curve for the same data. Note that the specificity is defined using predicted area, rather than true

3. MaxEnt Model

(5) 모델링: 훈련

```
##### MaxEnt model 제작
Mx.DB <- train_DB[c(1,var.num)]
Mx.test_DB <- test_DB

set.seed(6808)
Mx.model <-  maxent(Mx.DB[c(var.num)],Mx.DB[,1])
e.Mx<- evaluate(subset(test_DB[c(1,var.num)], test_DB$pop > 0), subset(test_DB[c(1,var.num)], test_DB$pop == 0), Mx.model)
t.Mx <- threshold(e.Mx, 'spec_sens')
```

※ evaluate(): Model evaluation function

※ threshold(): Find threshold (cut-off) between P and A

→ Model에서 제시해주는 값임

3. MaxEnt Model

(5) 모델링: 검증 (ACC, AUC index)

※ predict(): 훈련된 모델에 새로운 자료 적용

```
##### 모델 성능 평가
# 모델 검증 결과(test)
Mx.test <- predict(Mx.model, Mx.test_DB, type='class')
```

```
> predict(Mx.model, Mx.test_DB, type='class')
 [1] 0.4567523 0.5920558 0.6939507 0.5853251 0.4434561 0.5908368
```

※ 0 ~ 1 사이 값 → Probability

```
### 1) AUC
Mx_p1 <- ROCR::prediction(as.numeric(as.vector(Mx.test)), Mx.test_DB$pop)
Mx_pp1 <- performance(Mx_p1, 'tpr', 'fpr')
Mx_auc1 <- performance(Mx_p1, "auc")
Mx_auc <- Mx_auc1@y.values
print(paste0("AUC : ",round(Mx_auc[[1]],3)))

### 2) ACC
test_matrix <- Mx.test_DB[,1]
for(i in 1:length(test_matrix)){
  if(Mx.test[i] > t.Mx){test_matrix[i] <- 1} else {test_matrix[i] <- 0}
}
Mx_test_ACC <- confusionMatrix(factor(Mx.test_DB[,"pop"]),factor(test_matrix))$overall[1]
print(paste0("ACC : ",round(Mx_test_ACC,3)))
```

```
> print(paste0("AUC : ",round(Mx_auc[[1]],3)))
[1] "AUC : 0.789"
> print(paste0("ACC : ",round(Mx_test_ACC,3)))
[1] "ACC : 0.739"
```

종합

```
# AUC, ACC 모음
Mx.mvalue.df <- data.frame(matrix(NA, nrow=1,ncol=2))
names(Mx.mvalue.df) <- c("AUC", "ACC")
Mx.mvalue.df[1,1] <- round(Mx_auc[[1]],3) #AUC
Mx.mvalue.df[1,2] <- round(Mx_test_ACC,3) #ACC
# Mx.mvalue.df
```

```
> Mx.mvalue.df
    AUC   ACC
1 0.789 0.739
```

```
##### 입력자료에 대한 최종 서식확률 계산
Mx.result <- predict(Mx.model, model.DB[-1])
```

※ Model DB = Train + Test data

3. MaxEnt Model

※ AUC(AUROC)와 ACC

- Confusion matrix

- ACC (Accuracy): 정확도

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

※ https://en.wikipedia.org/wiki/Confusion_matrix

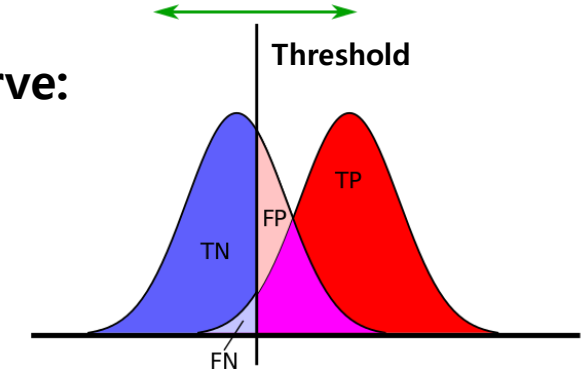
※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

3. MaxEnt Model

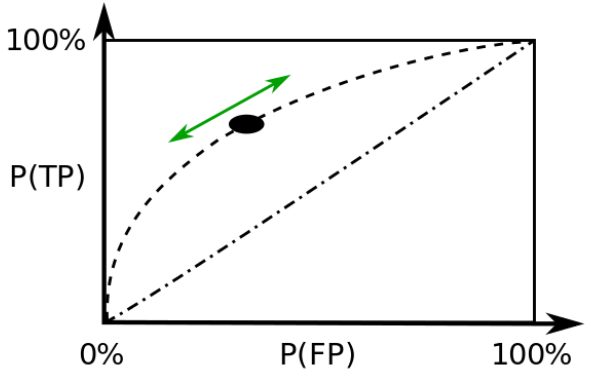
※ AUC(AUROC)와 ACC

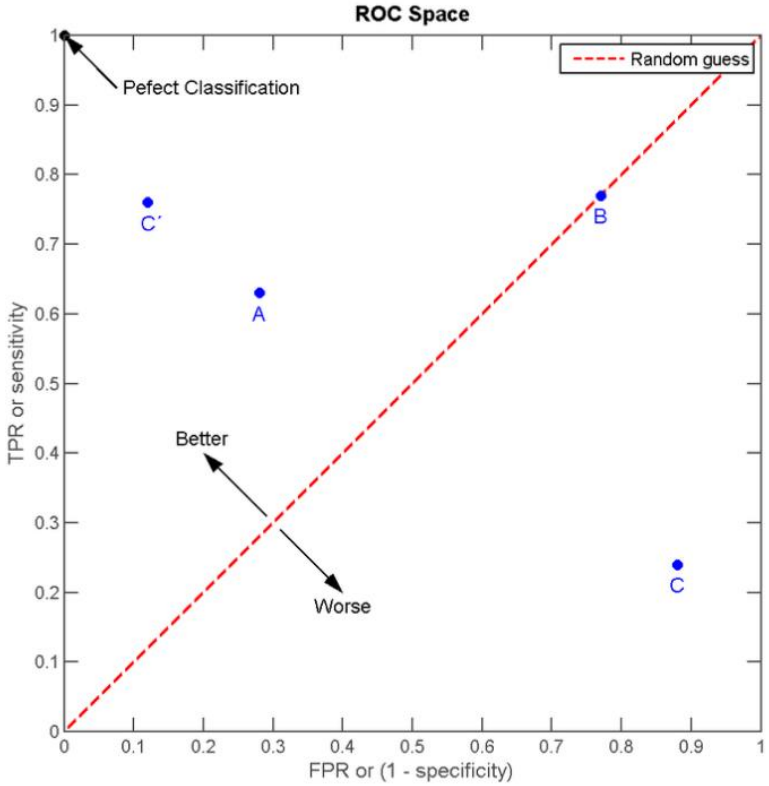
- Area Under a ROC Curve (AUROC; AUC)

ROC curve:



TP	FP
FN	TN





※ 50%

※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

3. MaxEnt Model

※ AUC(AUROC)와 ACC

- Confusion matrix

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

- AUC

※ https://en.wikipedia.org/wiki/Receiver_operating_characteristic

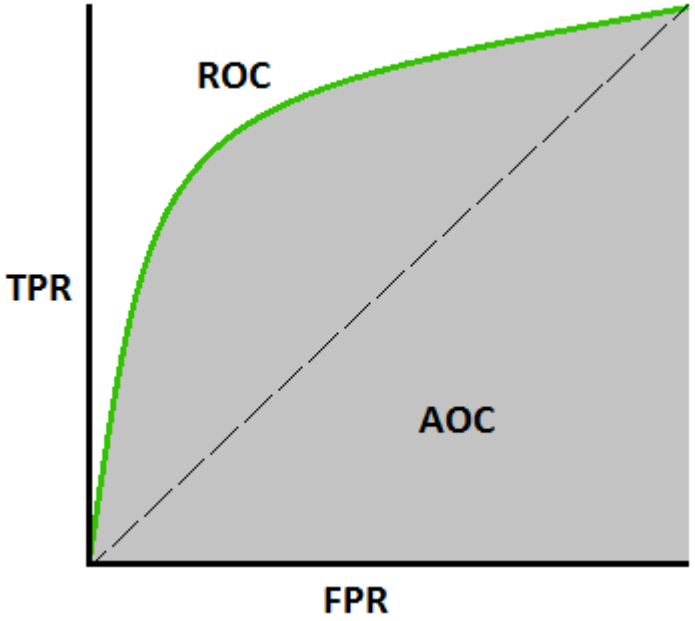
※ https://en.wikipedia.org/wiki/Confusion_matrix

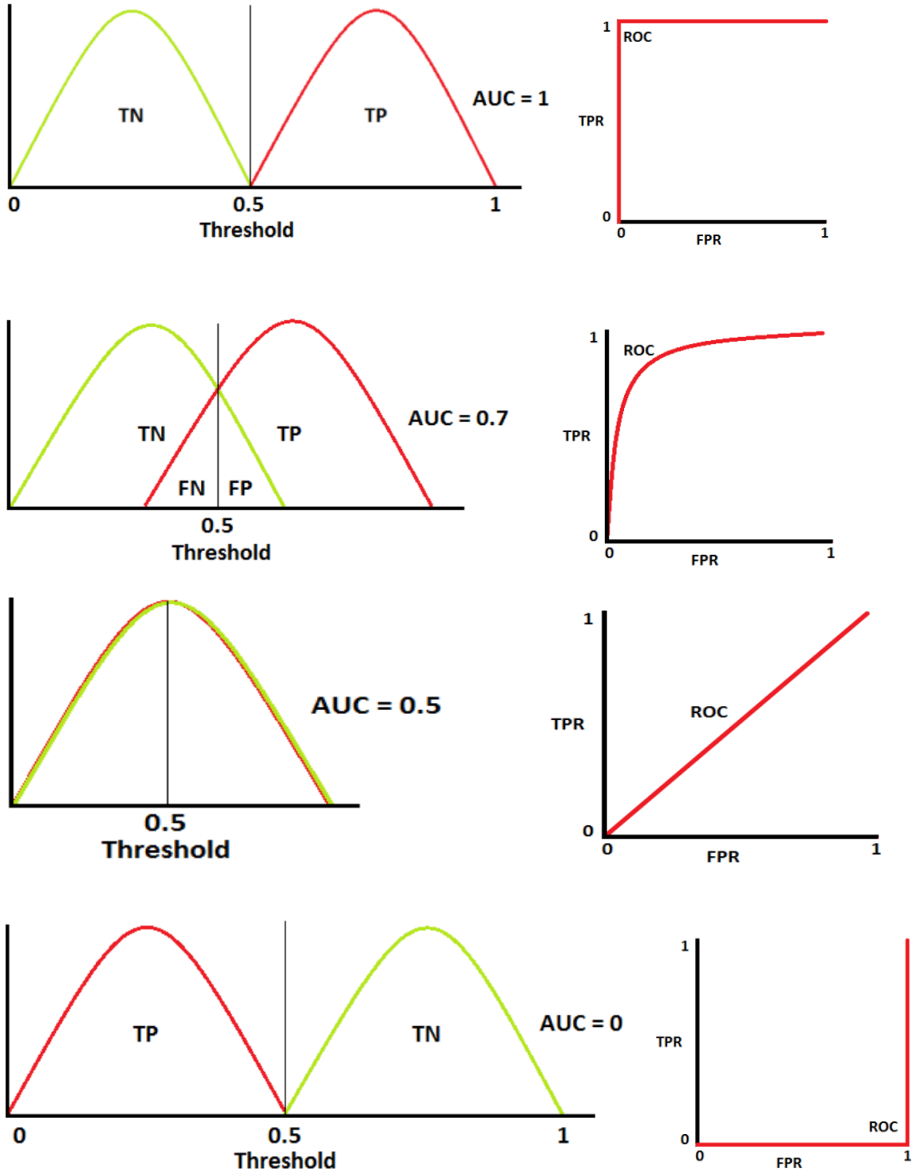
3. MaxEnt Model

※ AUC(AUROC)와 ACC

- Area Under a ROC Curve (AUROC; AUC)

AUC:





※ **Writer: Sarang Narkhede**
※ https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

3. MaxEnt Model

(5) 모델링 결과 정리 및 저장

```
#######################################################################
################################### 5. 결과 정리 및 저장하기
#정리하기

#####변수중요도
Mx.varimp <- plot(Mx.model)

# Mx.result #서식확률 예측결과
# Mx.varimp #모델 내 종합된 변수 중요도
# Mx.mvalue.df #모델 평가값
```

```
> Mx.result
  [1] 0.45298287 0.45675233 0.10862217 0.57558000 0.23007813 0.76069450
```

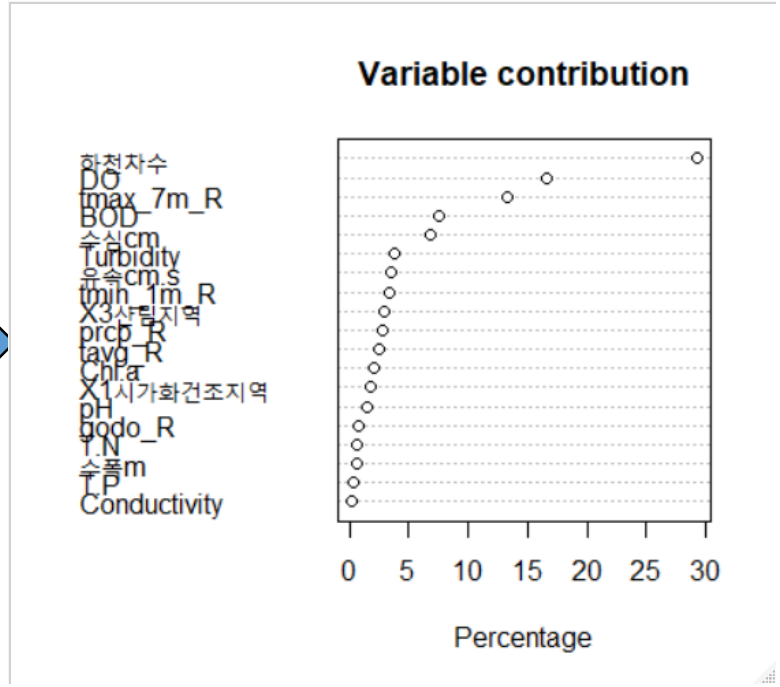
```
> Mx.varimp
    Conductivity             T.P            수폭m             T.N          godo_R
          0.2541          0.2796           0.5596          0.6538          0.7358
              pH X1시가화건조지역          Chl.a          tavg_R         prcp_R
          1.5458          1.7827           1.9839          2.4247          2.7431
      X3산림지역       tmin_1m_R        유속cm.s       Turbidity          수심cm
          2.9537          3.2798           3.4926          3.8363          6.8045
             BOD      tmax_7m_R              DO          하천차수
          7.5335         13.2485          16.6188         29.2693
```

```
> Mx.mvalue.df
    AUC   ACC
1 0.789 0.739
```

※ 상대적 중요도임

→ Sum(Mx.varimp) = 100 (%)

```
> plot(Mx.model)
```



3. MaxEnt Model

(5) 모델링 결과 정리 및 저장

```
#######################################################################
################################### 5. 결과 정리 및 저장하기
#정리하기

#####변수중요도
Mx.varimp <- plot(Mx.model)

# Mx.result #서식확률 예측결과
# Mx.varimp #모델 내 종합된 변수 중요도
# Mx.mvalue.df #모델 평가값

#평가 대상 생물
names(Total.DB)[target.sp]

#저장하기
temp.result <- as.data.frame(Mx.result)
names(temp.result) <- c("출현확률")
write.csv(temp.result, paste0("(결과)어류_모의결과(",names(Total.DB)[target.sp],").csv"))

temp.varimp <- as.data.frame(Mx.model$importance[,2])
names(temp.varimp) <- names(Total.DB)[target.sp]
write.csv(temp.varimp, paste0("(결과)어류_변수중요도(",names(Total.DB)[target.sp],").csv"))

temp.eval <- as.data.frame(t(Mx.mvalue.df))
names(temp.eval) <- names(Total.DB)[target.sp]
write.csv(temp.eval, paste0("(결과)어류_모델평가값(",names(Total.DB)[target.sp],").csv"))
```

```
> #평가 대상 생물
> names(Total.DB)[target.sp]
[1] "피라미"
```

※ 파일 저장 형태: CSV파일

(결과)어류_모델평가값(피라미)
(결과)어류_모의결과(피라미)
(결과)어류_변수중요도(피라미)

→ write.csv(): csv파일로 저장하기

3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
###### 부분의존성 그림 설정
# 부분의존성 그림 내 대상 환경 변수 설정
names(Env.DB) #사용환경명과 동일하게 선택
PDP.variables = c("godo_R","tavg_R","Conductivity", "Turbidity", "BOD", "T.N", "T.P","수폭m", "수심cm", "유속cm.s")


##### PDP 옵션
Mx.model
PDP.variables # "2. 모델 설정" 단계에서 정한 주요 환경 변수


##### 그림 크기 설정
par(mar=c(4,3,1,1))
par(mfrow=c(3, 4)) # 3x4로 그림 배열
```

```
> names(Env.DB) #사용환경명과 동일하게 선택
 [1] "Total_Code"      "하천차수"        "수온"            "DO"              "pH"              "Conductivity"    "Turbidity"       "BOD"
 [9] "NH3.N"           "NO3.N"           "T.N"             "PO4.P"           "T.P"             "Chl.a"           "X1시가화건조지역" "X2농업지역"
[17] "X3산림지역"      "X4초지지역"      "X5습지"          "X6나지"          "X7수역"          "비율"            "slope_R"         "aspect_gra"
[25] "tavg_R"         "tmax_7m_R"       "tmin_1m_R"       "prcp_R"          "godo_R"          "수폭m"           "수심cm"          "유속cm.s"
```

```
> PDP.variables
 [1] "godo_R"       "tavg_R"       "Conductivity" "Turbidity"    "BOD"          "T.N"          "T.P"          "수폭m"        "수심cm"       "유속cm.s"
```

3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### 주요 환경 변수에 대해 모델 DB(RF.DB) 내 위치(열) 찾기
for(i in 1:length(PDP.variables)){
  if(i == 1){
    PDP.var.num <- (1:length(names(RF.DB[-1])))[names(RF.DB[-1]) == PDP.variables[i]]
  } else {
    PDP.var.num <- c(PDP.var.num, (1:length(names(RF.DB[-1])))[names(RF.DB[-1]) == PDP.variables[i]])
  }
} # for i
PDP.var.num
#변수 확인
names(var.DB)[PDP.var.num] == PDP.variables # 모두 True
```

```
> PDP.var.num
 [1] 12 13  4  5  6  7  8 17 18 19
> #변수 확인
> names(var.DB)[PDP.var.num] == PDP.variables # 모두 True
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

→ DB 내 보고자 하는 변수(PDP.variables)에 해당하는 열 찾기

→ 확인

3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### 그림 내 x축 범위 정하기 (Min, Max)
length(PDP.var.num) #주요 환경 변수 개수
names(var.DB)[PDP.var.num]
x.limit <- list()
range(na.omit(var.DB[,PDP.var.num[1]]))
x.limit[[1]] <- c(0, 800)
range(na.omit(var.DB[,PDP.var.num[2]]))
x.limit[[2]] <- c(5, 15)
#
range(na.omit(var.DB[,PDP.var.num[3]]))
x.limit[[3]] <- c(0,25000)
range(na.omit(var.DB[,PDP.var.num[4]]))
x.limit[[4]] <- c(0,350)
range(na.omit(var.DB[,PDP.var.num[5]]))
x.limit[[5]] <- c(0,12)
range(na.omit(var.DB[,PDP.var.num[6]]))
x.limit[[6]] <- c(0,12)
range(na.omit(var.DB[,PDP.var.num[7]]))
x.limit[[7]] <- c(0,1.2)
range(na.omit(var.DB[,PDP.var.num[8]]))
x.limit[[8]] <- c(0,1200)
range(na.omit(var.DB[,PDP.var.num[9]]))
x.limit[[9]] <- c(0,120)
range(na.omit(var.DB[,PDP.var.num[10]]))
x.limit[[10]] <- c(0,120)
```

```
> names(var.DB)[PDP.var.num]
 [1] "godo_R"       "tavg_R"       "Conductivity" "Turbidity"
> x.limit <- list()
> range(na.omit(var.DB[,PDP.var.num[1]]))
[1]   0 720
> x.limit[[1]] <- c(0, 800)
> range(na.omit(var.DB[,PDP.var.num[2]]))
[1]  6.54779 13.43050
> x.limit[[2]] <- c(5, 15)
```

※ 1번 변수: 고도(godo)

고도 범위: 0 ~ 720 (m)

X축 범위 설정: 0 ~ 800 (m)

※ 2번 변수: 평년 평균기온(tavg)

고도 범위: 6.5 ~ 13.4 (℃)

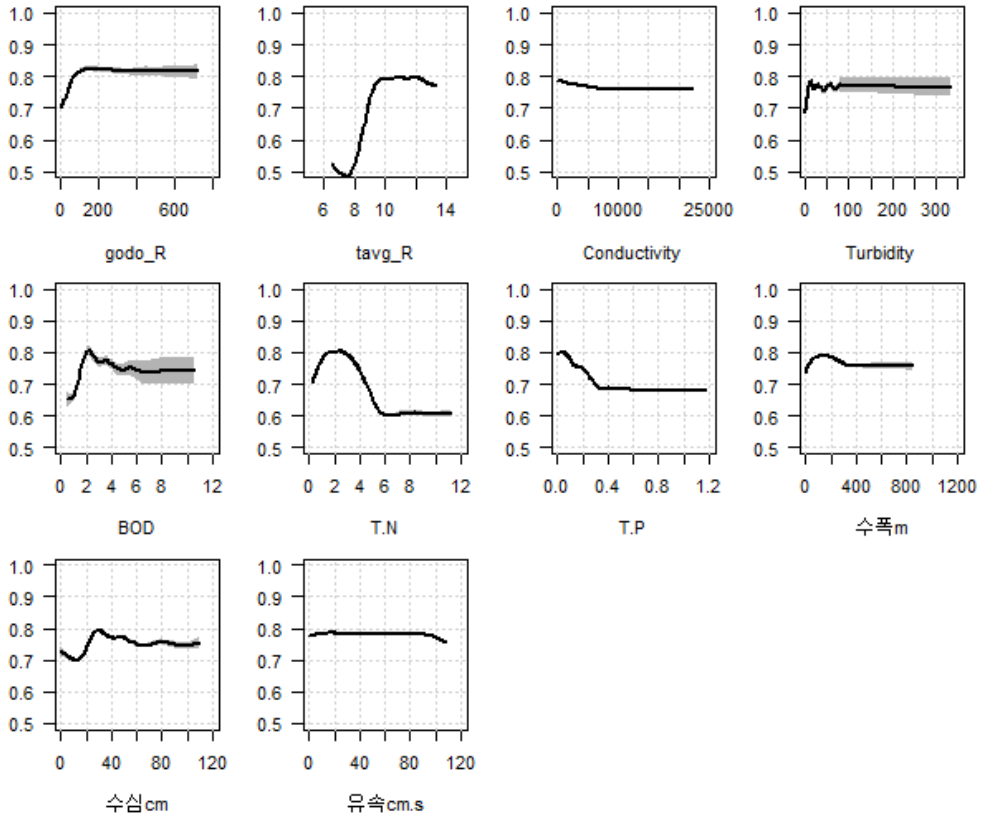
X축 범위 설정: 5 ~ 15 (℃)

3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### PDP 그리기
for(j in 1:length(PDP.var.num)){ # 1) PD값 구하기
  temp <- model.DB[-1]
  xx <- unique(temp[,PDP.var.num[j]])
  yy <- numeric(length(xx))

  for(num in 1:length(xx)){
    temp[,j] <- xx[num]
    preds <- predict(Mx.model, temp)
    yy[num] <- mean(preds)
  }

  # 2) GAM모델 용 DB 제작
  gam.iv.DB <- data.frame("y"=yy, "x"=(xx*var.info[PDP.var.num[j],3]+var.info[PDP.var.num[j],2] ))

  if(length(unique(gam.iv.DB$x))>10){
    gam.iv <- predict(gam(y ~ s(x), data=gam.iv.DB), se=T)
  } else {
    gam.iv <- predict(gam(y ~ s(x, k=length(unique(gam.iv.DB$x))), data=gam.iv.DB), se=T)
  }

  fit <- gam.iv$fit
  se <- gam.iv$se.fit # 신뢰구간
  lcl <- fit - 1.96*se
  ucl <- fit + 1.96*se

  lin.bind <- data.frame("x"=gam.iv.DB$x, fit)
  pol.bind <- data.frame("x"=gam.iv.DB$x, ucl, lcl)

  x.pol <- c(gam.iv.DB[order(gam.iv.DB$x),2],gam.iv.DB[order(gam.iv.DB$x, decreasing=T),2])
  y.pol <- c(pol.bind[order(pol.bind$x),2], pol.bind[order(pol.bind$x, decreasing=T),3])

  ### y축 범위 재설정
  ylimit <- c(0.2, 0.8)

  # 3) PDP 그리기
  plot(c(0.5,0.5), xlim=c(x.limit[[j]][1],
                          x.limit[[j]][2]),
       ylim=ylimit, type="n", yaxt="n", xlab=PDP.variables[j], ylab="")
  grid()
  polygon(x.pol, y.pol, col="gray70",border=NA) #"black")
  lines(lin.bind[order(lin.bind$x),], col="black", lwd=2)
  axis(2, las=2)
}# for j
```

→ for 함수를 통해 대상 변수 plot을 한 번에 그려줌



3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
##### PDP 그리기
for(j in 1:length(PDP.var.num)){ # 1) PD값 구하기
  temp <- model.DB[-1]
  xx <- unique(temp[,PDP.var.num[j]])
  yy <- numeric(length(xx))

  for(num in 1:length(xx)){
    temp[,j] <- xx[num]
    preds <- predict(Mx.model, temp)
    yy[num] <- mean(preds)
  }

  # 2) GAM모델 용 DB 제작
  gam.iv.DB <- data.frame("y"=yy, "x"=(xx*var.info[PDP.var.num[j],3]+var.info[PDP.var.num[j],2] ))

  if(length(unique(gam.iv.DB$x))>10){
    gam.iv <- predict(gam(y ~ s(x), data=gam.iv.DB), se=T)
  } else {
    gam.iv <- predict(gam(y ~ s(x, k=length(unique(gam.iv.DB$x))), data=gam.iv.DB), se=T)
  }
```

※ Algorithm :
모델 DB에서 대상 변수(PDP.variables)의 값을
대상 변수의 고유값(unique)으로 반복적으로 치환한 후
(for num), 모델 예측 결과값의 변화를 확인함

→ 'GAM 모델'을 통해 부분의존성 효과에 대한 모델 결과값을 fitting해줌(추세선)
※ 주의사항: 'model.DB'객체는 환경 변수가 표준화(scale)가 된 값이기에,
표준화 이전 값으로 반환해주고 fitting을 진행함 (var.info 이용)

※ 표준화 z = (X – mean)/sd → X = z(≒ Model.DB값) * sd + mean

```
> var.info
                name          mean           sd
1           하천차수    5.51034483 1.983052e+00
2                DO    9.41471264 1.862191e+00
3                pH    7.89172414 5.201697e-01
4      Conductivity  298.82321839 1.131712e+03
5         Turbidity   10.09839080 2.004259e+01
6               BOD    1.93471264 1.110015e+00
7               T.N    2.38296552 1.217586e+00
8               T.P    0.06201379 9.026808e-02
9             Chl.a    3.72850575 6.416491e+00
10 X1시가화건조지역   11.03603908 1.325349e+01
11       X3산림지역   26.55498621 2.354068e+01
12           godo_R  104.70679977 1.168564e+02
13           tavg_R   10.95712149 1.350842e+00
14         tmax_7m_R   27.84787563 9.843470e-01
15         tmin_1m_R   -8.42283251 2.617238e+00
16           prcp_R 1232.35022299 1.018252e+02
17            수폭m   85.40620690 1.182695e+02
18            수심cm   30.60091954 1.783805e+01
19          유속cm.s   25.47241379 2.329885e+01
```

3. MaxEnt Model

(6) 부분 의존성 그림(Partial Dependence Plot; PDP) 그리기

```
fit <- gam.iv$fit
se <- gam.iv$se.fit # 신뢰구간
lcl <- fit - 1.96*se
ucl <- fit + 1.96*se

lin.bind <- data.frame("x"=gam.iv.DB$x, fit)
pol.bind <- data.frame("x"=gam.iv.DB$x, ucl, lcl)

x.pol <- c(gam.iv.DB[order(gam.iv.DB$x),2],gam.iv.DB[order(gam.iv.DB$x, decreasing=T),2])
y.pol <- c(pol.bind[order(pol.bind$x),2], pol.bind[order(pol.bind$x, decreasing=T),3])

### y축 범위 재설정
ylimit <- c(0.2, 0.8)

# 3) PDP 그리기
plot(c(0.5,0.5), xlim=c(x.limit[[j]][1],
                        x.limit[[j]][2]),
     ylim=ylimit, type="n", yaxt="n", xlab=PDP.variables[j], ylab="")
grid()
polygon(x.pol, y.pol, col="gray70",border=NA) #"black")
lines(lin.bind[order(lin.bind$x),], col="black", lwd=2)
axis(2, las=2)
}# for j
```

→ 신뢰구간(95%, 1.96)은 다각형(polygon으로 표현함)

→ y축 범위: 설정 가능함 (e.g. 0.5~1.0)

※ 실질적으로 그림을 그리는 함수부분

순서: 1. Plot(): 배경, 범위 지정

(type="n"으로 내용을 표시하지 않음, yaxt="n"로 y축 표시하지 않음)

2. grid(): 배경 격자

3. polygon(): 신뢰구간 표시

4. lines(): GAM모델을 이용한 추세선 표시

5. axis(2): y축 표시

3. MaxEnt Model

(7) 신규 자료를 이용한 확률 예측

```
#######################################################################
################################### 7. 모델을 이용한 서식 확률 예측 (신규자료 바탕)

##### 신규자료 불러오기
# new.env <- read.csv("수생태_어류_환경.csv", stringsAsFactors=F)
new.env <- na.omit(var.DB)

##### 표준화
## 기존 모델 내 변수 순서와 신규 자료 내 변수 순서 맞춤
for(k in 1:length(names(new.env))){
  new.num <-(1:length(names(new.env)))[var.info[,1]== names(new.env)[k]]
  if(k == 1){
    new.num2 <- new.num
  } else{
    new.num2 <- c(new.num2, new.num)
  }
}
names(new.env)[new.num2] # 모델 내 변수 순서로 정렬
order.env <- new.env[new.num2]

## scale(표준화) : z = (Mean - X)/sd
scaled.env <- order.env
for(j in 1:length(new.num2)){
  scaled.env[,j] <- (order.env[,j] - var.info[j,2])/(var.info[j,3])
}

##### 모델 예측
Mx.pred <- predict(Mx.model, scaled.env)
pred.result <- data.frame("서식확률"=Mx.pred)

# 저장하기
write.csv(pred.result, "(R결과)어류_MaxEnt_신규모의결과.csv")
```

※ **신규자료**: 새로운 지점의 환경자료를 의미

※ **모델링과 동일 과정을 거침**
→ NA자료 제거, 환경변수 표준화

※ **모의**: predict()함수 이용

국립환경과학원

경희대학교 KYUNG HEE UNIVERSITY

THE UNIVERSITY OF SEOUL 서울시립대학교

EnFRA

수생태계모델연구 컨소시엄

감사합니다

2018-2019년에 수행된 국립환경과학원 연구 과제 :

차세대 수질. 수생태계 예측모델 개발 및 적용성 평가(II)

- 국내 수질. 수생태계 모니터링 자료를 활용한 수생태 모델 활용성 평가”로 수행된 연구 결과의 일부임