데이터 분석 기초 - 데이터 파악 및 수정

환경생태데이터사이언스 실습 September 24, 2019

지난 주 과제 풀이

간단한 과제

큐 모듈을 적절히 변형하여 다음과 같은 생태계 지수를 계산할 수 있는 모형을 작성하세요.

1. 현재 생물 종과 각각의 종의 수를 입력하여 Shannon index 구하기.
2. 새로운 종이 들어왔을 때 각기 다른 생물 종의 숫자 (풍부도), Shannon index (다양성 지수)를 계산 및 출력.
3. 하나의 종이 사라졌을 때 생물 종의 숫자 (풍부도), Shannon index (다양성 지수)를 계산 및 출력.

- Shannon index (다양성): $S = -\sum plog(p)$
- Richness (풍부도): R

간단한 해결 방법

```
CalDiv <- function (){
    q <- c (); qsize <- 0
    Species_in <- function (data){
        q <<- c (q, data)
        qsize <<- qsize + 1
        prob <- q/sum(q)
        Shannon_index <- - sum(prob*log(prob))
        return (c(qsize, Shannon_index))
    }
```

간단한 해결 방법

```
    Species_out <- function (){
        first <- q[1]
        q <<- q[ - 1]
        qsize <<- qsize - 1
        prob <- q/sum(q)
        Shannon_index <- - sum(prob*log(prob))
        return (c(qsize, Shannon_index))
    }
    size <- function (){ return (qsize)}
    return ( list (Species_in = Species_in,
                   Species_out = Species_out,  size = size))
}
```

다른 해결 방법

```
Diversity <- function(){
    q <- c(); qsize <- 0
    species <- c()
    Species_in <- function(Name, Number){
        q <<- c(q, Number)
        species <<- c(species, Name)
        qsize <<- qsize + 1
        prob <- q/sum(q)
        ShannonIndex <- -sum(prob*log(prob))
        return(list("Richness" = qsize, "Diversity" = ShannonIndex))
    }
```

다른 해결 방법

```
    Species_out <- function(Name){
        Out <- q[c(species==Name)]
        q <<- q[!c(species==Name)]
        qsize <<- qsize - 1
        prob <- q/sum(q)
        ShannonIndex <- -sum(prob*log(prob))
        return(list("Out" = Out, "Richness" = qsize,
                    "Diversity" = ShannonIndex))
    }
    size <- function(){return(qsize)}
    return(list(Species_in = Species_in,
                Species_out = Species_out, size = size))
}
```

오늘의 학습 목표

데이터 분석 기초

1. 데이터 호출 및 저장
2. 데이터 구조 및 요약 통계치 확인
3. 데이터에 자료 추가
4. 데이터에 함수 적용
5. 데이터 분리 및 병합
6. 기초 도표 그리기

데이터 호출 및 저장

기본 데이터 호출 방법

R에서 주로 이용하는 데이터 형식: 문서 기반 자료

1. csv: Comma separated values
2. tsv: tab separated values
3. dat: csv 형태이거나 tsv 형태일 가능성이 있음
4. 공백으로 구분된 자료, 특정 기호로 구분된 자료

기본 데이터 호출 방법

R 기본 내장 함수를 이용한 자료 호출

```
# ReadData <- read.table("/path/to/the/file", sep = " ", header = T)

Read_csv_1 <- read.table("./Data/KMA_20190916_0922.csv", sep=",",
                         header=T)
Read_tsv_1 <- read.table("./Data/MeteorologicalData.tsv", sep="\t",
                         header=T)
Read_dat_1 <- read.table("./Data/MeteorologicalData.dat", sep="\t",
                         header=T)
```

csv 파일은 read.csv() 명령어를 이용하여 호출

```
Read_csv <- read.csv("./Data/KMA_20190916_0922.csv")
```

파일 호출 옵션

- `header`: 데이터에 열이름이 있는지 확인 (TRUE, FALSE)
- `na.strings`: NA로 지정된 값이 있으면 명시 (TRUE, FALSE)
- `stringAsFactors`: 문자열을 팩터로 읽을 것인지 판단 (TRUE, FALSE)

기본 데이터 저장 방법

- 데이터 호출 방법과 유사

```
write.table(Data, "/path/to/the/file/", sep = " ",
            col.names = T, row.names = T)
write.csv(Read_csv, "./Data/KMA_20190916_0922_edited.csv",
          row.names=F)
# Check whether file is in the folder or not
```

- col.names: 열이름을 데이터와 함께 저장
- row.names: 행이름을 데이터와 함께 저장 (행 이름이 없으면 행의 순서를 행이름으로 저장)

작업 공간에 있는 데이터 바이너리로 저장 및 호출

- 작업 공간에 있는 객체를 저장할 때

```
save(Data, "/path/to/the/file/")
```

- 작업 공간에 있는 모든 데이터를 저장

```
save.image(Data, "/path/to/the/file/")
```

- 저장한 데이터 호출

```
load("/path/to/the/file")
```

데이터 구조 및 요약 통계치 확인

데이터의 구조를 직접 확인하는 방법

```r
head(Read_csv) # Check first 6 rows of the data
```

```
##   ID            Time Temperature Precipitation WindVelocity
## 1 90 2019-09-16 1:00        16.5            NA          1.8
## 2 90 2019-09-16 2:00        16.2            NA          0.9
## 3 90 2019-09-16 3:00        16.2            NA          1.3
## 4 90 2019-09-16 4:00        16.8            NA          1.6
## 5 90 2019-09-16 5:00        16.9            NA          1.1
## 6 90 2019-09-16 6:00        17.3            NA          1.2
##   WindDirection
## 1           290
## 2           270
## 3           230
## 4           270
## 5           270
## 6           270
```

데이터의 구조를 직접 확인하는 방법

```r
tail(Read_csv) # Check last 6 rows of the data
```

```
##        ID             Time Temperature Precipitation
## 13531 295 2019-09-21 19:00        17.2           6.0
## 13532 295 2019-09-21 20:00        17.4          10.5
## 13533 295 2019-09-21 21:00        17.5           8.5
## 13534 295 2019-09-21 22:00        17.5           5.0
## 13535 295 2019-09-21 23:00        17.5           6.0
## 13536 295  2019-09-22 0:00        17.7           1.5
##       WindVelocity WindDirection
## 13531          1.7            50
## 13532          2.2            20
## 13533          1.8            20
## 13534          2.0            20
## 13535          1.9            20
## 13536          1.8            50
```

명령어를 이용하여 데이터 구조 확인 방법

```r
class(Read_csv) # Check the data class
```

```
## [1] "data.frame"
```

```r
str(Read_csv) # Check the structure of data
```

```
## 'data.frame':    13536 obs. of  6 variables:
##  $ ID           : int  90 90 90 90 90 90 90 90 90 90 ...
##  $ Time         : Factor w/ 144 levels "2019-09-16 1:00",..: 1 12
##  $ Temperature  : num  16.5 16.2 16.2 16.8 16.9 17.3 17.6 18.6 20.
##  $ Precipitation: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ WindVelocity : num  1.8 0.9 1.3 1.6 1.1 1.2 1.2 1.3 1.1 1.9 ...
##  $ WindDirection: int  290 270 230 270 270 270 270 290 340 50 ...
```

데이터의 기본 속성 확인 후 변수형 수정

위 데이터에서 ID는 기상대의 고유 번호를 뜻하므로 factor여야 하고, Time 또한 시간이므로 시간 자료로 형 변환해야한다.

```
# integer to factor
Read_csv$ID <- as.factor(Read_csv$ID)
Read_csv$Time <- strptime(Read_csv$Time, format=c("%Y-%m-%d %H:%M"))
str(Read_csv$ID)
```

```
##  Factor w/ 94 levels "90","93","95",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
str(Read_csv$Time)
```

```
##  POSIXlt[1:13536], format: "2019-09-16 01:00:00" "2019-09-16 02:00
```

summary 함수를 이용한 전체 데이터 요약 통계 확인

```
summary(Read_csv)
```

```
##         ID                Time
##  90     :  144   Min.   :2019-09-16 01:00:00
##  93     :  144   1st Qu.:2019-09-17 12:45:00
##  95     :  144   Median :2019-09-19 00:30:00
##  98     :  144   Mean   :2019-09-19 00:30:00
##  99     :  144   3rd Qu.:2019-09-20 12:15:00
##  100    :  144   Max.   :2019-09-22 00:00:00
##  (Other):12672
##   Temperature    Precipitation     WindVelocity
##  Min.   : 6.50   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:17.00   1st Qu.: 0.400   1st Qu.: 0.700
##  Median :19.70   Median : 1.700   Median : 1.400
##  Mean   :19.99   Mean   : 2.464   Mean   : 1.815
##  3rd Qu.:22.90   3rd Qu.: 3.900   3rd Qu.: 2.500
##  Max.   :31.50   Max.   :23.500   Max.   :13.400
```

데이터에 자료 추가

데이터에 새로운 열 추가

새로운 행을 만들고 강우량이 기록된 행은 “Rain”, 그리고 그렇지 않은 곳은 “No Rain” 이라고 기록하기

```
Read_csv$RainOrNot <-
    ifelse( is.na( Read_csv$Precipitation ), "No Rain", "Rain" )
head(Read_csv$RainOrNot)
```

```
## [1] "No Rain" "No Rain" "No Rain" "No Rain" "No Rain"
## [6] "No Rain"
```

```
tail(Read_csv$RainOrNot)
```

```
## [1] "Rain" "Rain" "Rain" "Rain" "Rain" "Rain"
```

주어진 벡터를 이용한 열 추가

```
Column_Data <- data.frame("Country" = rep("Korea", nrow(Read_csv)),
                          "Season" = rep("Autumn", nrow(Read_csv)))
head(Column_Data, 3)
```

```
##   Country Season
## 1   Korea Autumn
## 2   Korea Autumn
## 3   Korea Autumn
```

```
ColumnBind <- cbind(Read_csv, Column_Data); head(ColumnBind, 2)
```

```
##   ID                Time Temperature Precipitation
## 1 90 2019-09-16 01:00:00        16.5            NA
## 2 90 2019-09-16 02:00:00        16.2            NA
##   WindVelocity WindDirection RainOrNot Country Season
## 1          1.8           290   No Rain   Korea Autumn
## 2          0.9           270   No Rain   Korea Autumn
```

주어진 벡터를 이용한 행 추가

```
Row_Data <- Read_csv[1:3, ] # Crop first 6 rows of Read_csv data
RowBind <- rbind(Read_csv, Row_Data); tail(RowBind, 6)
```

```
##        ID                Time Temperature Precipitation
## 13534 295 2019-09-21 22:00:00        17.5           5.0
## 13535 295 2019-09-21 23:00:00        17.5           6.0
## 13536 295 2019-09-22 00:00:00        17.7           1.5
## 13537  90 2019-09-16 01:00:00        16.5            NA
## 13538  90 2019-09-16 02:00:00        16.2            NA
## 13539  90 2019-09-16 03:00:00        16.2            NA
##       WindVelocity WindDirection RainOrNot
## 13534          2.0            20      Rain
## 13535          1.9            20      Rain
## 13536          1.8            50      Rain
## 13537          1.8           290   No Rain
## 13538          0.9           270   No Rain
## 13539          1.3           230   No Rain
```

데이터에 함수 적용

apply 계열 함수

함수	설명	다른 함수와 비교했을 때의 특징
apply( )	배열 또는 행렬에 주어진 함수를 적용한 뒤 그 결과를 벡터, 배열 또는 리스트로 반환	배열 또는 행렬에 적용
lapply( )	벡터, 리스트 또는 표현식에 함수를 적용하여 그 결과를 리스트로 반환	결과가 리스트
sapply( )	lapply와 유사하지만 결과를 벡터, 행렬 또는 배열로 반환	결과가 벡터, 행렬 또는 배열
tapply( )	벡터에 있는 데이터를 특정 기준에 따라 그룹으로 묶은 뒤 각 그룹마다 주어진 함수를 적용하고 그 결과를 반환	데이터를 그룹으로 묶은 뒤 함수를 적용
mapply( )	sapply의 확장된 버전으로, 여러 개의 벡터 또는 리스트를 인자로 받아 함수에 각 데이터의 첫째 요소들을 적용한 결과, 둘째 요소들을 적용한 결과, 셋째 요소들을 적용한 결과 등을 반환	여러 데이터를 함수의 인자로 적용

출처: 서명구 (2014). R을 이용한 데이터 처리 & 분석 실무. 길벗

apply

```
ApplyTest <- RowBind
# Apply apply function to rows
#apply(ApplyTest[,c(3:6)], 1, FUN=sum) # 1: by row, 2: by column
# Apply apply function to each column from 3 to 6.
apply(ApplyTest[,c(3:6)], 2, FUN=sum) # 1: by row, 2: by column
```

```
##   Temperature Precipitation  WindVelocity WindDirection
##            NA            NA            NA            NA
```

```
apply(ApplyTest[,c(3:6)], 2, FUN=sum, na.rm=T)
```

```
##   Temperature Precipitation  WindVelocity WindDirection
##      270089.5        3978.9       24526.3     1931070.0
```

```
apply(ApplyTest[,c(3:6)], 2, FUN=function(x){sum(x, na.rm=T)})
```

```
##   Temperature Precipitation  WindVelocity WindDirection
##      270089.5        3978.9       24526.3     1931070.0
```

lapply와 sapply

```r
lapply(ApplyTest[,c(3:5)], FUN=sum)
```

```
## $Temperature
## [1] NA
##
## $Precipitation
## [1] NA
##
## $WindVelocity
## [1] NA
```

```r
sapply(ApplyTest[,c(3:5)], FUN=sum)
```

```
##   Temperature Precipitation  WindVelocity
##            NA            NA            NA
```

lapply와 sapply

```r
lapply(ApplyTest[,c(3:5)], FUN=sum, na.rm=T)
```

```
## $Temperature
## [1] 270089.5
##
## $Precipitation
## [1] 3978.9
##
## $WindVelocity
## [1] 24526.3
```

```r
sapply(ApplyTest[,c(3:5)], FUN=sum, na.rm=T)
```

```
##   Temperature Precipitation  WindVelocity
##      270089.5        3978.9       24526.3
```

lapply

```
lapply(1:3, FUN=function(x){x*2})
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 6
```

```
unlist(lapply(1:3, FUN=function(x){x*2}))
```

```
## [1] 2 4 6
```

tapply

```
tapply(ApplyTest$WindVelocity, ApplyTest$RainOrNot,
       FUN = mean, na.rm = T)
```

```
##  No Rain     Rain
## 1.710062 2.590341
```

```
tapply(ApplyTest$Temperature, ApplyTest$RainOrNot,
       FUN = mean, na.rm = T)
```

```
##  No Rain     Rain
## 20.30308 17.68675
```

데이터 분리 및 병합

데이터 분리 및 병합

함수	특징
split( )	주어진 조건에 따라 데이터를 분리한다.
subset( )	주어진 조건을 만족하는 데이터를 선택한다.
merge( )	데이터를 공통된 값에 기준해 병합한다.

출처: 서명구 (2014). R을 이용한 데이터 처리 & 분석 실무. 길벗

split

split 함수는 특정 팩터로 정해진 열의 요소 별로 자료를 나눈 후 리스트 형식으로 저장한다.

```
SplitResult <- split(Read_csv, Read_csv$ID)
head(SplitResult)
```

```
## $`90`
##     ID                Time Temperature Precipitation
## 1   90 2019-09-16 01:00:00        16.5             NA
## 2   90 2019-09-16 02:00:00        16.2             NA
## 3   90 2019-09-16 03:00:00        16.2             NA
## 4   90 2019-09-16 04:00:00        16.8             NA
## 5   90 2019-09-16 05:00:00        16.9             NA
## 6   90 2019-09-16 06:00:00        17.3             NA
## 7   90 2019-09-16 07:00:00        17.6             NA
## 8   90 2019-09-16 08:00:00        18.6             NA
## 9   90 2019-09-16 09:00:00        20.5             NA
## 10  90 2019-09-16 10:00:00        22.7             NA
```

subset

subset 함수는 특정 조건에 맞는 자료들을 선택하여 추출한다.

```
SubsetResult <- subset(Read_csv, ID==90,
                       select=c("ID", "Time", "WindDirection"))
head(SubsetResult)
```

```
##   ID                 Time WindDirection
## 1 90 2019-09-16 01:00:00           290
## 2 90 2019-09-16 02:00:00           270
## 3 90 2019-09-16 03:00:00           230
## 4 90 2019-09-16 04:00:00           270
## 5 90 2019-09-16 05:00:00           270
## 6 90 2019-09-16 06:00:00           270
```

merge

두 개의 데이터 프레임을 공통된 요소를 기준으로 병합한다.

```
# Read station name
stnName <- read.csv("./Data/stnInfo_20190923231722.csv",
                    fileEncoding="euc-kr")
MergedData <- merge(Read_csv, stnName, by.x = c("ID"), by.y=c("□□"))
head(MergedData)
```

```
##    ID                Time Temperature Precipitation
## 1 100 2019-09-16 01:00:00        12.5            NA
## 2 100 2019-09-16 01:00:00        12.5            NA
## 3 100 2019-09-16 02:00:00        11.4            NA
## 4 100 2019-09-16 02:00:00        11.4            NA
## 5 100 2019-09-16 03:00:00        11.0            NA
## 6 100 2019-09-16 03:00:00        11.0            NA
##   WindVelocity WindDirection RainOrNot     □□□
## 1            0             0   No Rain 2006-11-07
## 2            0             0   No Rain 1971-07-15
```

기타 다양한 데이터 처리 함수

데이터 정렬 (sort와 order)

sort 함수는 벡터를 정렬한 값을 반환하고 order 함수는 벡터를 정렬했을 때 순서를 반환한다. 즉 하나의 벡터를 정렬하고자 하면 sort 함수를 쓰고 데이터 프레임 전체를 특정 열에 대하여 정렬하고자 할 때에는 order 함수를 쓴다.

```
x <- rnorm(5, 10, 10)
x
```

```
## [1]  4.731310 14.712878 -1.454767 22.065373 14.635022
```

```
sort(x)
```

```
## [1] -1.454767  4.731310 14.635022 14.712878 22.065373
```

```
order(x)
```

```
## [1] 3 1 5 2 4
```

데이터 정렬 (sort와 order)

```r
sort(x, decreasing=T)
```

```
## [1] 22.065373 14.712878 14.635022  4.731310 -1.454767
```

```r
order(x, decreasing=T)
```

```
## [1] 4 2 5 1 3
```

```r
Read_csv_sorted <-
    Read_csv[order(as.numeric(as.character(Read_csv$ID)),
                   decreasing = T),]
head(Read_csv_sorted)
```

```
##        ID                Time Temperature Precipitation
## 13393 295 2019-09-16 01:00:00        23.3            NA
## 13394 295 2019-09-16 02:00:00        23.3            NA
## 13395 295 2019-09-16 03:00:00        23.2            NA
## 13396 295 2019-09-16 04:00:00        23.1            NA
```

원하는 조건을 가진 원소의 위치

which, which.max, which.min 함수는 각각 조건에 맞는 원소의 위치와 최대, 혹은 최소 값의 위치를 알려준다.

```
# Location of elements whose temperature are larger than 30
which(Read_csv$Temperature > 30)
```

```
##  [1]  1599  1600  6543  9566  9567  9568  9998  9999 10000
## [10] 11151 12160 12590 12591
```

```
which.max(Read_csv$Temperature) # Location of the maximum value
```

```
## [1] 9998
```

```
which.min(Read_csv$Temperature) # Location of the minimum value
```

```
## [1] 750
```

원하는 조건을 가진 원소의 위치

```
# Extract rows whose temperature is larger than 30
# Read_csv[which(Read_csv$Temperature > 30 ), ]
 # Extract a row with the maximum temperature
Read_csv[which.max(Read_csv$Temperature),]
```

```
##       ID                Time Temperature Precipitation
## 9998 257 2019-09-18 14:00:00        31.5            NA
##      WindVelocity WindDirection RainOrNot
## 9998          2.4           110   No Rain
```

```
# Extract a row with the minimum temperature
Read_csv[which.min(Read_csv$Temperature),]
```

```
##      ID                Time Temperature Precipitation
## 750 100 2019-09-17 06:00:00         6.5            NA
##     WindVelocity WindDirection RainOrNot
## 750            0             0   No Rain
```

기초 도표 그리기

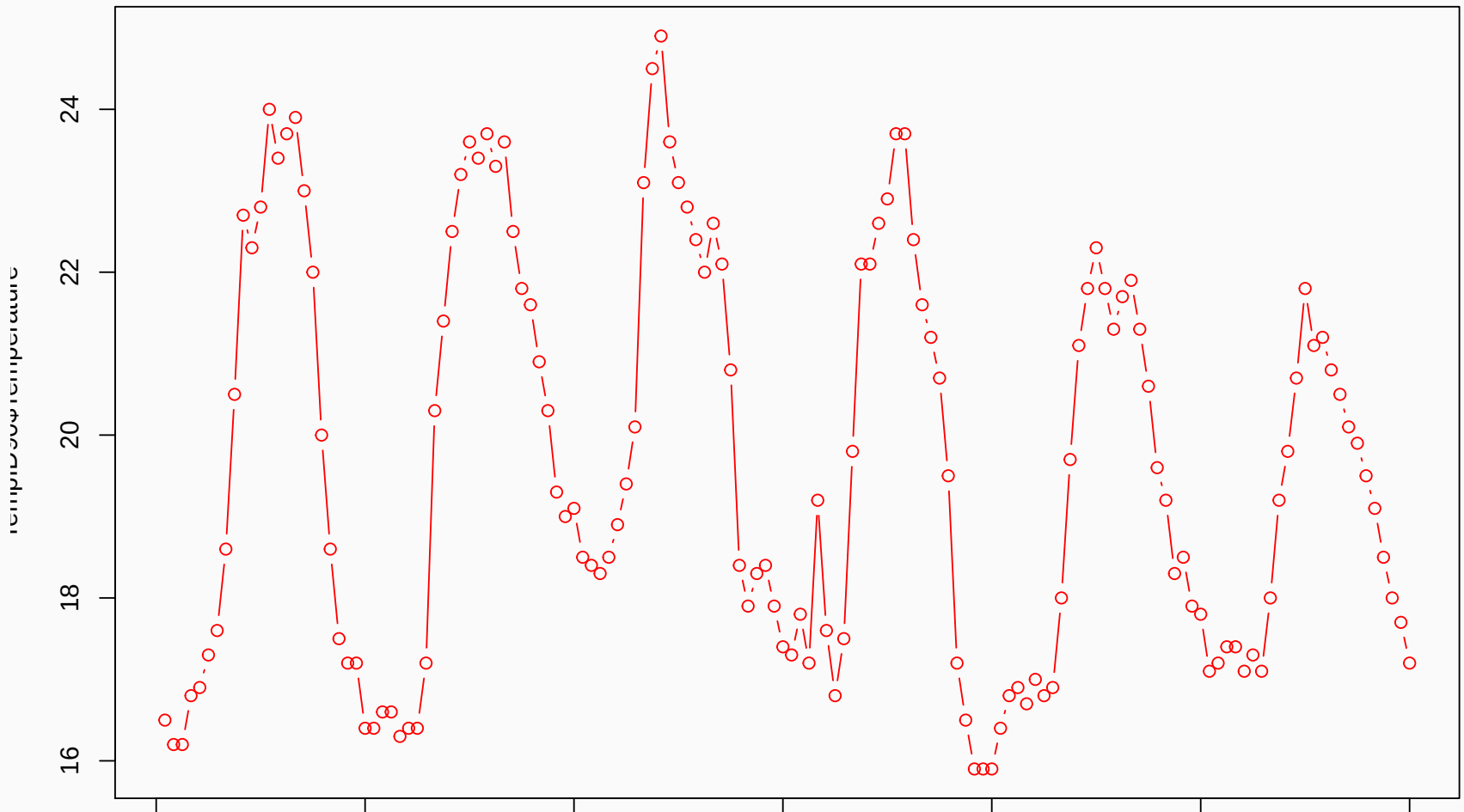
기초 도표 그리기

plot 함수를 사용하면 기본적인 도표를 작성할 수 있다.

```
# Basic usage
# plot(x = X_value, y = Y_value, type = c("p", "l", "b"),
#      xlab = "X label", ylab = "Y label", main = "Title",
#      col = "red")
# Using relationship
# plot( y_value ~ x_value, data = "Data")
```

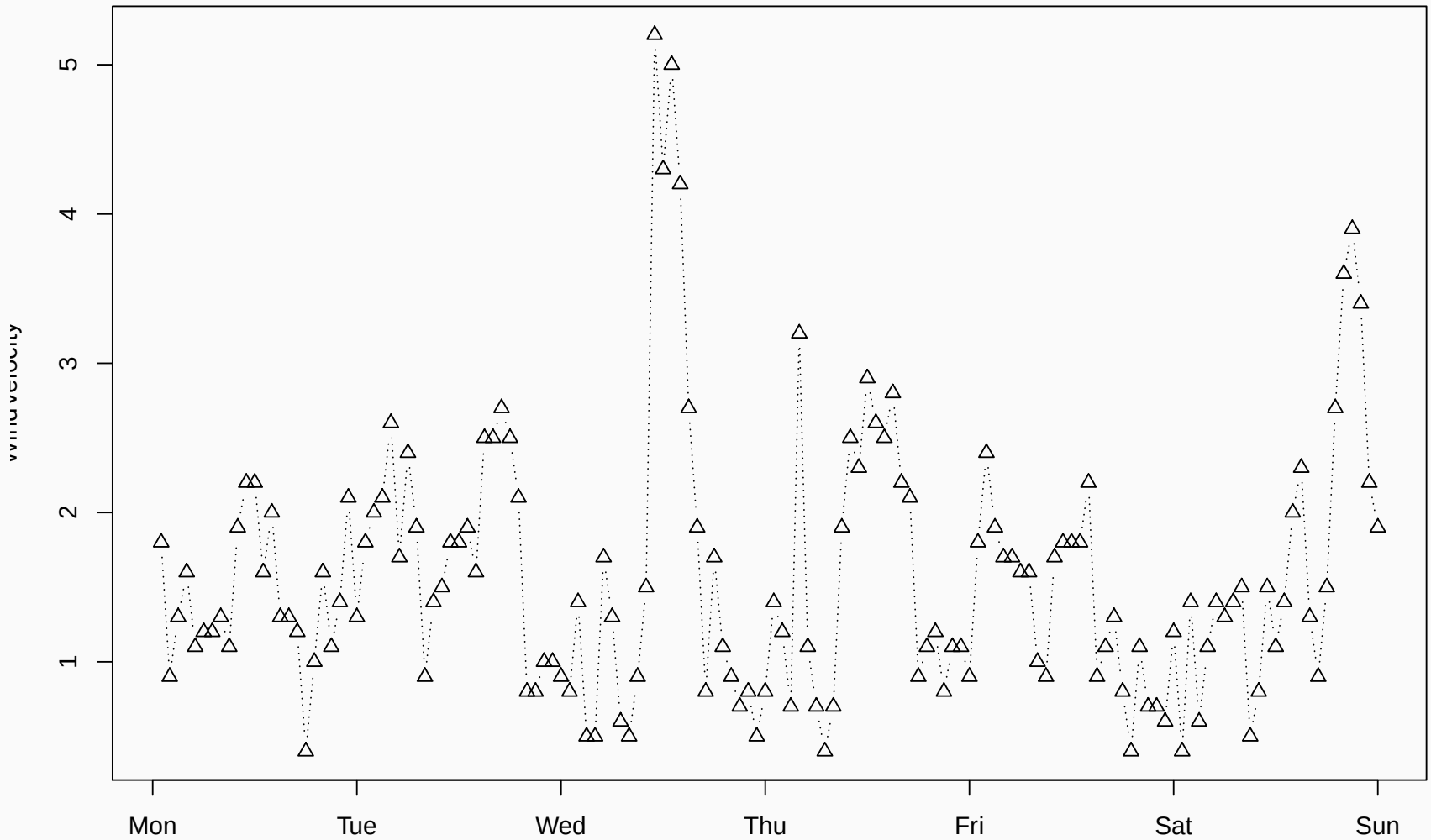
기초 도표 그리기

```
TempID90 <- subset(Read_csv, ID==90,
                   select=c(Time, Temperature, WindVelocity))
plot(x = TempID90$Time, y = TempID90$Temperature,
     type="b", col="red")
```



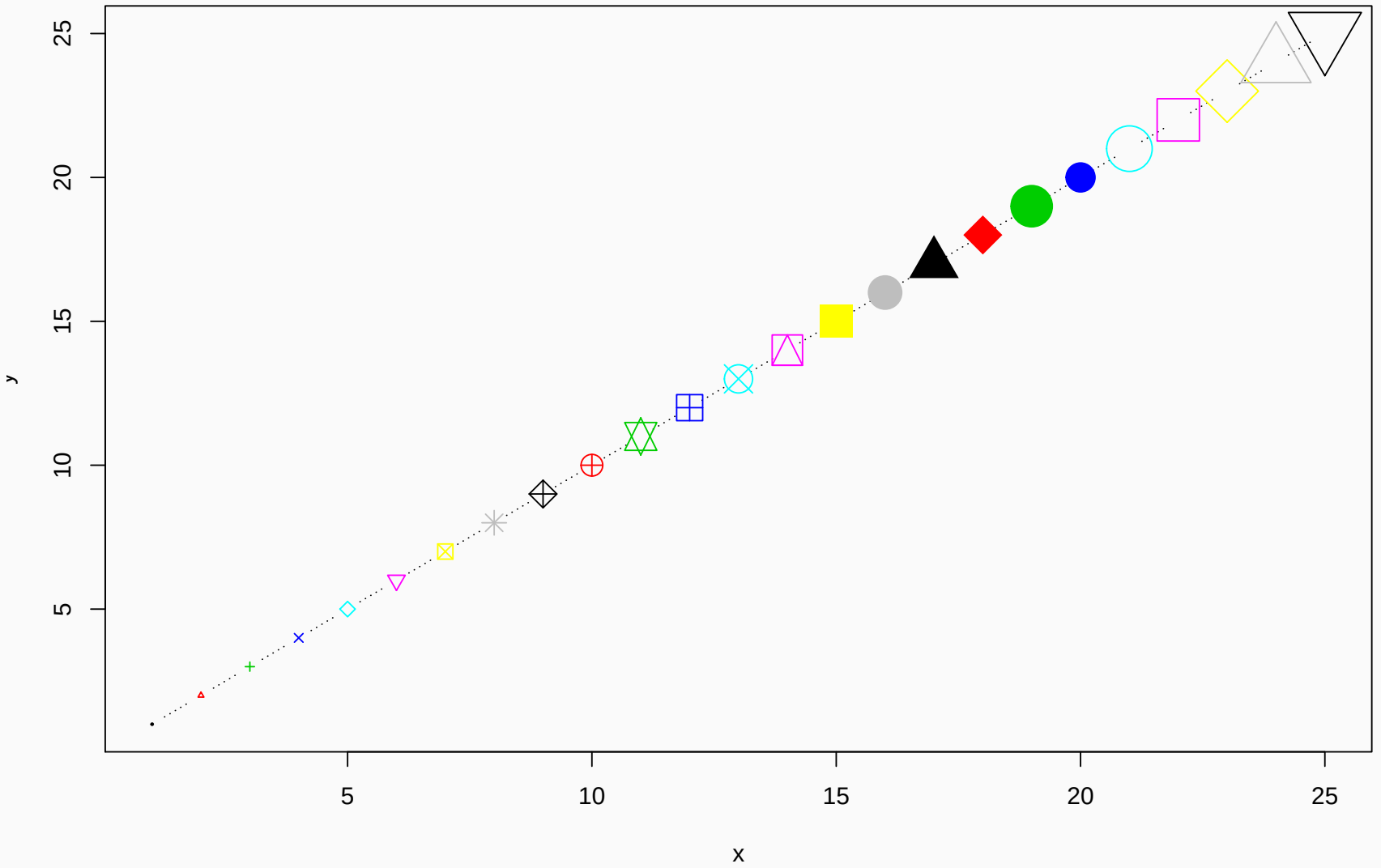
기초 도표 그리기

```r
TempID90$Time <- as.POSIXct(TempID90$Time)
plot(WindVelocity ~ Time, data = TempID90,
     type = "b", pch=24, lty = 3)
```



기초 도표 그리기

```
y <- x <- c(1:25)
plot(y ~ x, type = "b", pch=x, col=x, lty=3, cex=x/5)
```



기초 도표 그리기

```
plot(y ~ x, type = "b", pch=x, col=x, lty=3, cex=x/5)
points(2*y ~ x, type = "b", pch=x, col=x, lty=4, cex=x/5)
points(3*y ~ x, type = "b", pch=x, col=x, lty=1, cex=x/5)
```