9강 고급 그래픽 기법(1)

이화여대 통계학과 이은경교수

목 차

1 빅데이터 시각화란?

빅데이터 시각화

- 빅데이터 : 대량의 정형, 비정형의 자료
- 빅데이터 분석은 이러한 자료로 부터 정보를 추출하고 결과를 분석하는 기술뿐 아니라 결과의 전달을 위해 필요한 기술들을 모두 포함
- 빅데이터로부터 유용한 정보를 추출하고 이를 시각화하여 알기 쉽게 정보를 전달하는 기술이 중요해짐
- Infovis, Infographics, data visualization, statistical graphics, visual analytics 등의 다양한 용어와 분야들이 생겨남
- 자료로부터 추출된 정보를 그림으로 표현하여 효과적으로 전달해야 함
- 복잡한 자료를 가능한 간결하고 명확하게 그림으로 표현해야 함
- 기존의 방법으로는 한계가 있음

2 R을 이용한 빅데이터 시각화

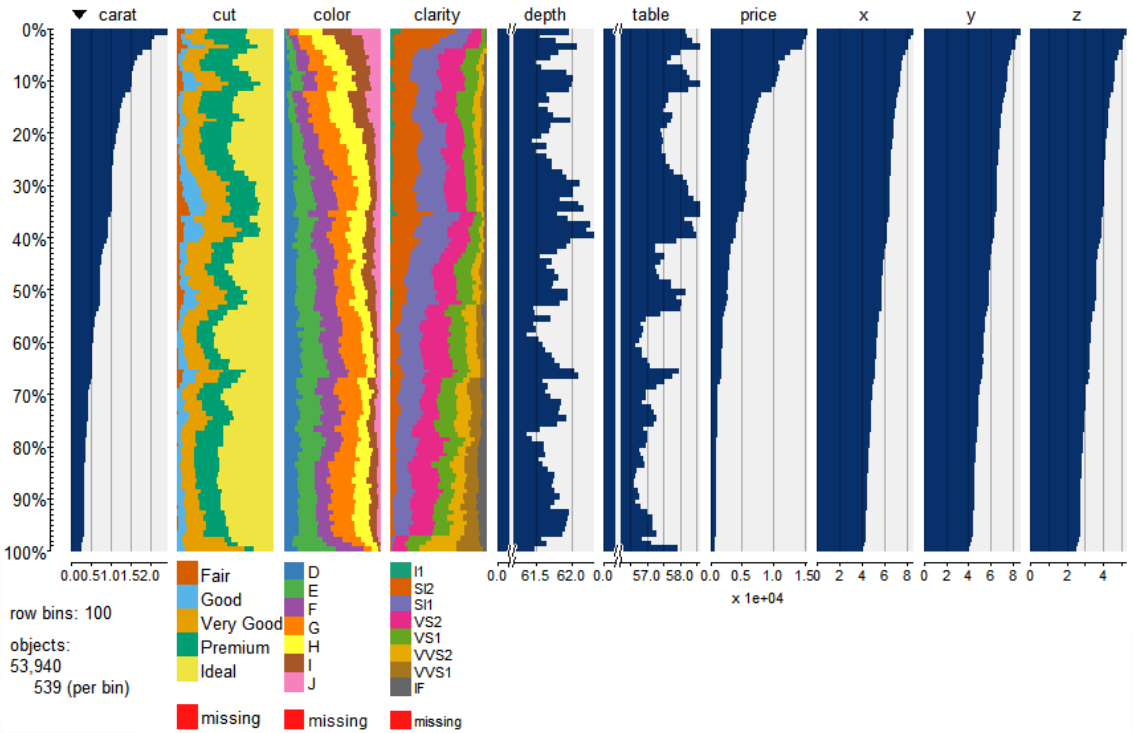
tabplot

- 대용량 자료를 특정 변수에 따라 범주화 한 후 각 범주에 따라 정리한 자료를 시각화하는 방법
- 하나의 열은 하나의 변수를 나타내며 각 행은 범주를 나타냄
- 연속변수 : 각 범주의 평균을 막대그래프 형태로 나타냄
- 범주형 변수 : 누적막대그래프의 형태로 나타냄
- R의 기본 라이브러리가 아니므로 사용 전에 설치해야 함
- 범주형 변수와 연속변수를 함께 표현할 수 있음

diamonds 자료

- 53940개의 다이아몬드에 대한 자료
- ggplot2라이브러리에 내장
  - carat : 무게
  - cut : 커팅 기술 (Fair/Good/Very Good/Premium/Ideal)
  - color : 원석의 색 (D/E/F/G/I/J)
  - clarity : 원석의 투명도(IF, VVS2,VVS1,VS2,VS1,SI2, SI1,I1)
  - price : 가격
  - x,y,z : 다이아몬드의 길이, 폭, 깊이
  - depth : 깊이의 비율 = 2*z/(x+y)
  - table : 가장 넓은 면의 폭

tabplot



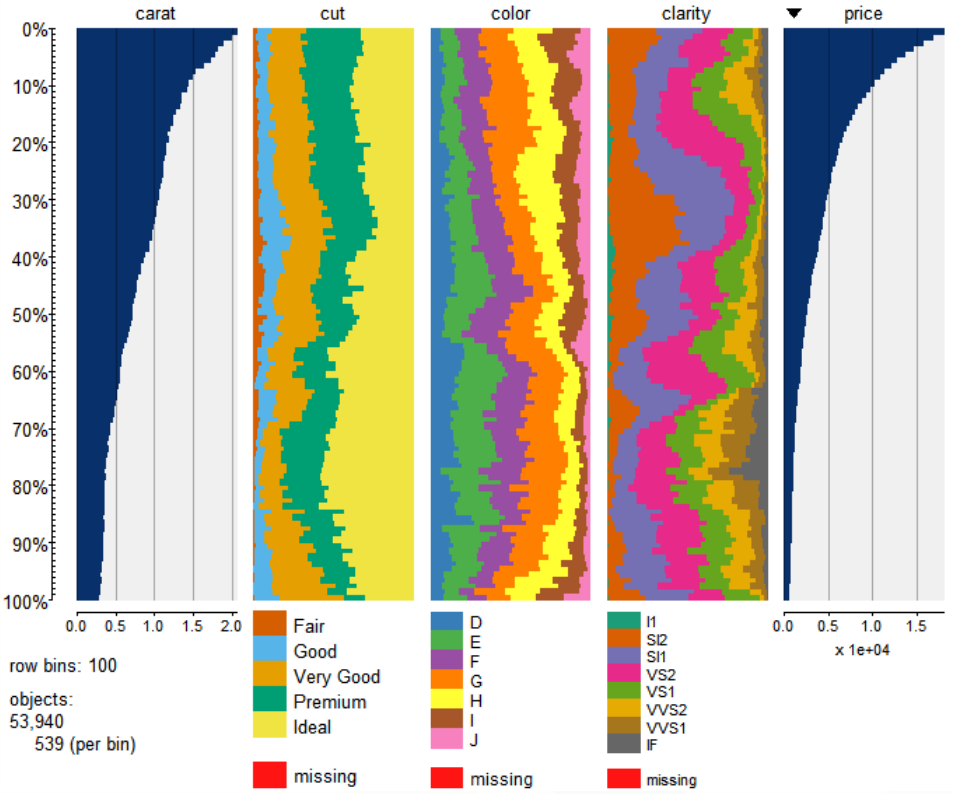
- 첫번째 변수(carat)을 기준으로 자료전체를 순서화하고 이 값을 기준으로 범주로 나누어 그린 그림

```
> library(tabplot)
> tableplot(diamonds)
```

tabplot

- Select 옵션을 이용하여 변수를 선택
- sortCol 옵션을 이용하여 정렬기준변수를 선택

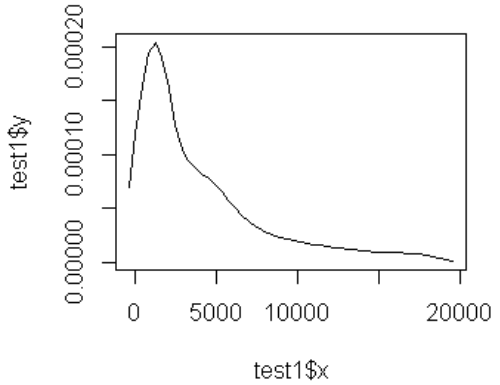
```
> tableplot(diamonds,
+    select=c(carat,cut,color,clarity,price),
+    sortCol=price)
```



ash

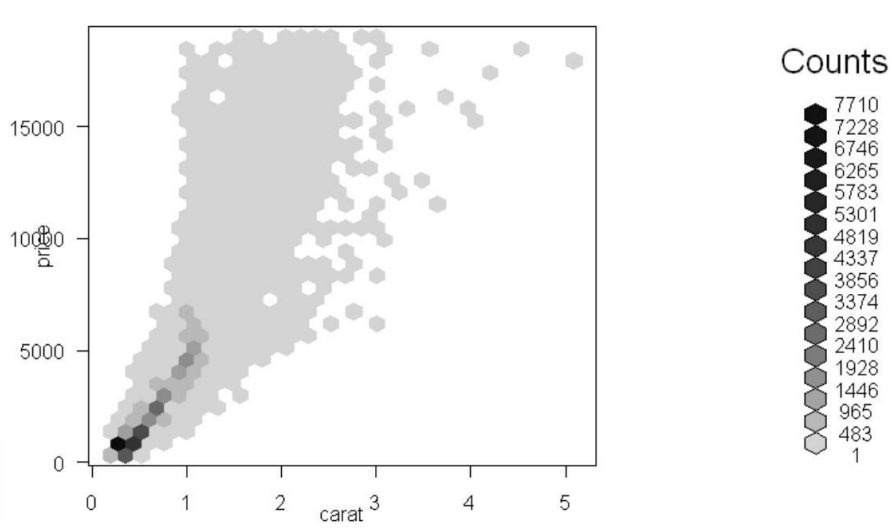
- ash 라이브러리 : averaged shifted histogram을 위하여 자료의 값을 범주화하고 범주 내의 분포함수 값을 계산해주는 함수를 제공
- bin1 : 자료를 범주화하기 위한 함수
- ash1 : average shifted 방법을 이용하여 분포를 계산하는 함수

```
> library(ash)
> test1<-ash1(bin1(diamonds$price,nbin=50),5)
[1] "ash estimate nonzero outside interval ab"
> plot(test1,type='l')
```



hexbin

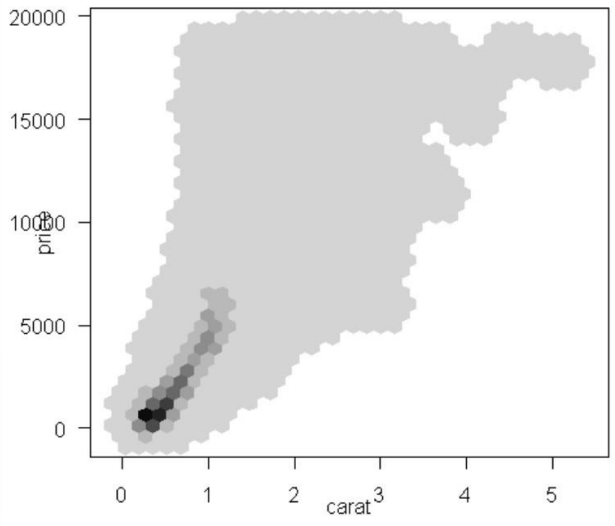
- 2차원의 자료를 범주화하여 각 범주내의 관측수를 계산하는 함수 hexbin을 제공




```
> library(hexbin)
> library("grid")
> x <- diamonds$carat
> y <- diamonds$price
> bin <- hexbin(x,y)
> plot(bin, xlab = "carat", ylab= "price")
```

hexbin

- smooth.hexbin 함수 : hexbin을 이용하여 범주화한 자료를 smoothing 기술을 이용하여 좀 더 부드러운 형태의 분포모양으로 바꿔주는 함수




```
> smbin <- smooth.hexbin(bin)
> plot(smbin, xlab = "carat", ylab = "price")
```

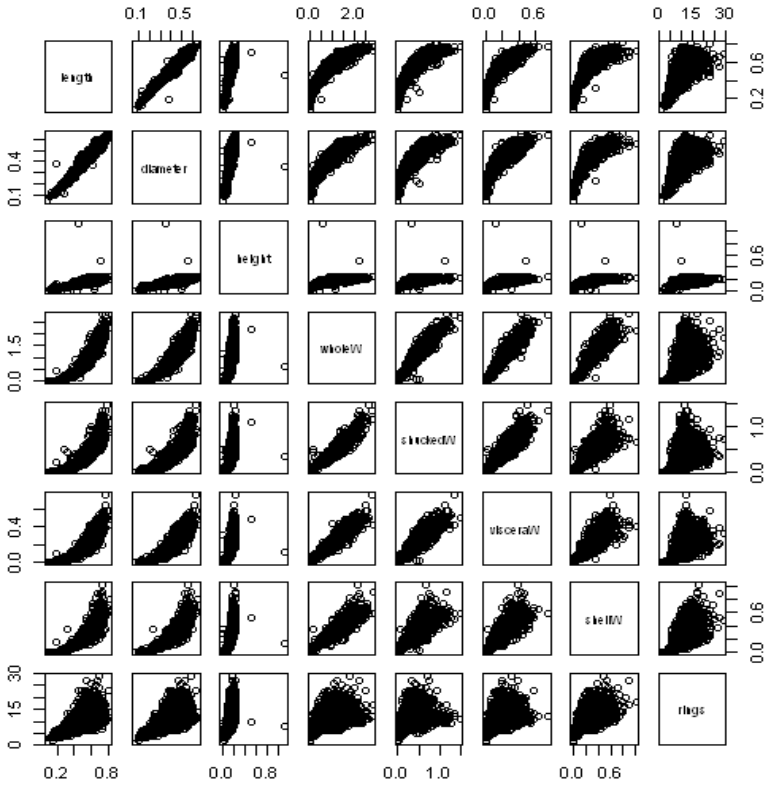
scagnostics

- 변수의 수가 늘어남에 따라 산점도 행렬의 크기는 점점 커지고 각 변수들 간의 관계를 살펴보기 힘들어짐
- 산점도 행렬로부터 각 산점도의 특징을 알아내기 위해 개발된 라이브러리
- 9가지 특성에 대한 값을 계산
  - outlying : 특이점
  - skewed : 치우침
  - clumpy : 자료의 뭉친 정도
  - sparse : 흩어짐
  - striated : 줄무늬
  - convex : 볼록한 정도
  - skinny : 자료가 좁게 모여있는 정도
  - stringy : 선을 이루고 있는 정도
  - monotonic : 단조 증가/감소 여부

scagnostics

- abalone 자료의 산점도 행렬
- 28개의 산점도

```
> pairs(abalone[, -1])
```



scagnostics

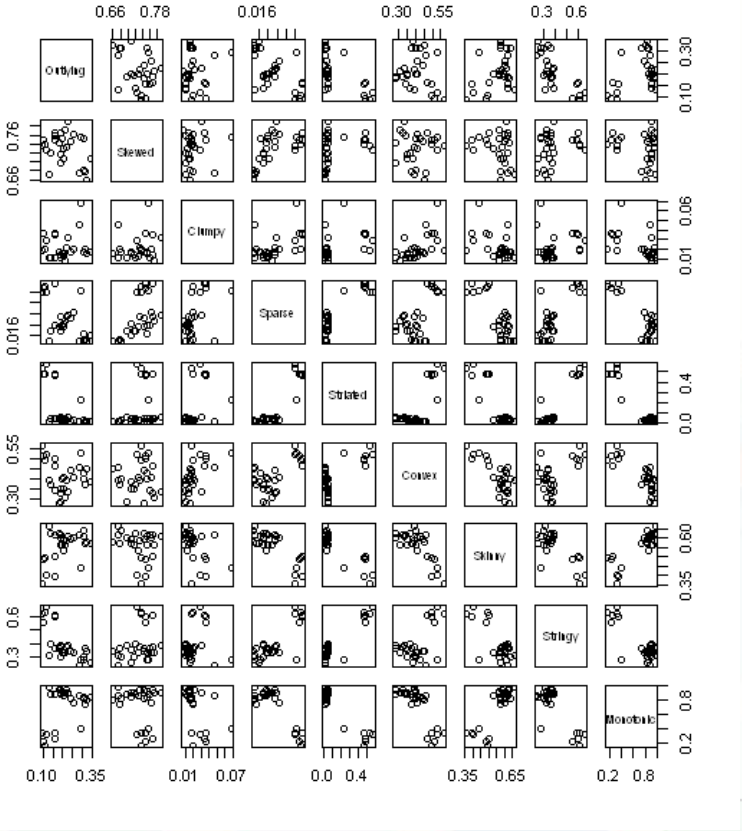
- Scagnostics 함수를 이용
- 9개 특성 *(산점도 수)의 행렬형태로 표시됨

```
> library(scagnostics)
> scag.abalone<-scagnostics(abalone[,-1])
> round(t(scag.abalone),2)
                    Outlying Skewed Clumpy Sparse Striated Convex Skinny Stringy Monotonic
length * diameter       0.13   0.67   0.01   0.02     0.02   0.40   0.66   0.31      0.97
length * height         0.31   0.75   0.02   0.02     0.01   0.43   0.61   0.28      0.85
diameter * height       0.28   0.67   0.05   0.02     0.01   0.46   0.62   0.24      0.87
length * wholeW         0.18   0.74   0.02   0.02     0.05   0.28   0.63   0.37      0.94
diameter * wholeW       0.19   0.77   0.01   0.02     0.04   0.33   0.62   0.37      0.94
.
.
.
shellW * rings          0.16   0.75   0.04   0.03     0.46   0.50   0.39   0.60      0.35
attr(,"class")
[1] "scagnostics"
```

scagnostics

- Scagnostics의 계산결과를 쉽게 파악하기 위하여 9가지 특성을 변수로, 각 변수 쌍의 산점도를 관측으로 하여 산점도 행렬을 그려볼 수 있음

```
> pairs(t(scag.abalone))
```

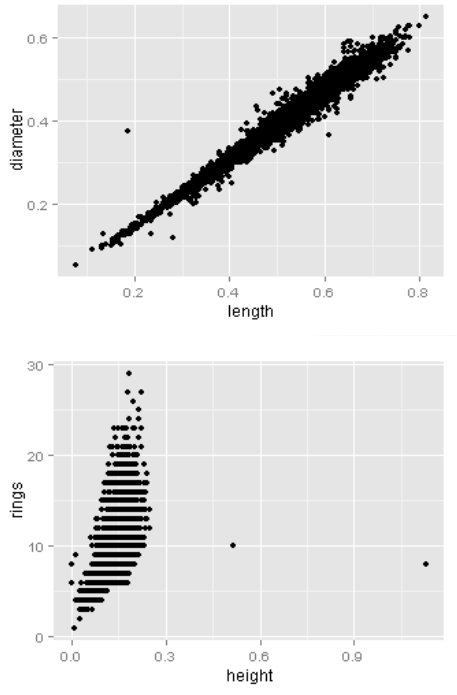


scagnostics

- scagnosticsOutliers 함수 : scagnostics 결과의 특이점을 찾아줌

```
> Notnormal.plot <- scagnosticsOutliers(scag.abalone)
> Notnormal.plot[Notnormal.plot]
  length * diameter     height * rings
               TRUE               TRUE
> round(scag.abalone[, Notnormal.plot], 4)
```

	length * diameter	height * rings
Outlying	0.1339	**0.2912**
Skewed	0.6724	0.7568
Clumpy	0.0110	**0.0685**
Sparse	0.0151	0.0241
Striated	0.0186	0.2238
Convex	0.3959	0.5294
Skinny	0.6644	0.4386
Stringy	**0.3112**	0.2838
Monotonic	**0.9748**	**0.3999**



3 R GUI 소개

R 그래픽 사용자 인터페이스 (R GUI)

- R은 커맨드 라인 사용자 인터페이스를 기반으로 하여 R의 시행을 위해서는 명령어들을 입력해야 함
- R GUI를 위한 패키지들이 개발되어 있어 R에서도 GUI를 구현할 수 있음
- tcltk
  - 스크립트 언어인 Tcl과 Tk GUI tool kit을 R에서 사용할 수 있도록 개발된 라이브러리
  - 비교적 간단한 프로그래밍으로 GUI 구현이 가능
- RGtk2
  - GUI를 만들기 위한 툴킷 라이브러리인 Gimp tool kit(GTK)을 R에서 이용할 수 있도록 개발된 라이브러리

4 gWidget을 이용한 R GUI 구현

gWidgets

- gWidgets
  - tcltk, RGkt2 등의 GUI를 위한 R 라이브러리에 대하여 독립적인 GUI를 만들 수 있도록 하는 API를 제공하는 라이브러리.
  - R에서의 GUI 프로그래밍을 훨씬 간편하게 만들어줌

```
> library(gWidgets)
> options("guiToolkit"="RGtk2")
```

gcheckbox

- 체크상자를 만들기 위한 함수

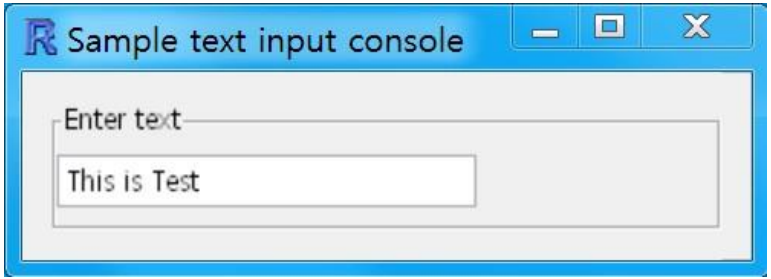
```
> win.1 <- gwindow("Sample checkbox")
> tmp<-gframe("Favorate color", cont=win.1)
> checkbox.1<-gcheckbox("White", cont=tmp)
> checkbox.2<-gcheckbox("Black", cont=tmp)
> checkbox.3<-gcheckbox("Red", cont=tmp)
> checkbox.4<-gcheckbox("Green", cont=tmp)
> checkbox.5<-gcheckbox("Blue", cont=tmp)
> checkbox.6<-gcheckbox("Pink", cont=tmp)
> checkbox.7<-gcheckbox("Yellow", cont=tmp)
```


```
> svalue(checkbox.1);
[1] TRUE
> svalue(checkbox.2);
[1] FALSE
> svalue(checkbox.3);
[1] FALSE
> svalue(checkbox.4);
[1] FALSE
> svalue(checkbox.5);
[1] TRUE
> svalue(checkbox.6);
[1] FALSE
> svalue(checkbox.7);
[1] FALSE
```

gedit

- 문자 입력을 위한 GUI를 만들기 위한 함수

```
> win.2 <- gwindow("Sample text input console")
> tmp<-gframe("Enter text",cont=win.2)
> edit.sample<-gedit("",cont=tmp)
```




```
> svalue(edit.sample)
[1] "This is Test"
```

gmenu

- GUI의 메뉴를 만들기 위한 함수

```
> win.3 <- gwindow("Sample menu")
> menu.list<-list()
> menu.list$AA$handler = function(h,...) print("A")
> menu.list$BB$handler = function(h,...) print("B")
> menu.list$CC$C1$handler = function(h,...) print("C1")
> menu.sample<-gmenu(menu.list,cont=win.3)
```

gtext / gstatusbar

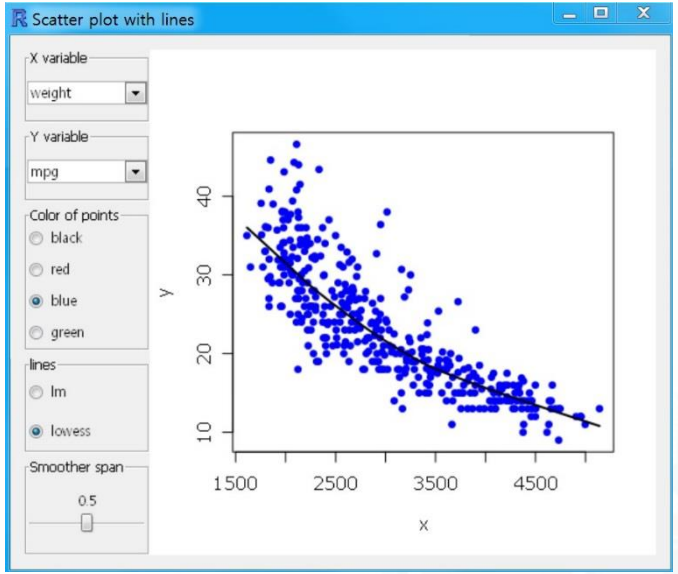
- gtext : 문자 입력 에디터를 만들기 위한 함수
- gstatusbar : GUI의 상태표시바를 만들기 위한 함수

```
> win.4 <- gwindow("Sample gtext and gstatusbar")
> #tmp<-gframe("Text Editor",cont=win.4)
> text.sample<-gtext("Type here!",cont=win.4)
> statusbar.sample<-gstatusbar("Typing here...",cont=win.4)
> svalue(statusbar.sample)<-"Still typing..."
```

```
> svalue(text.sample)
[1] "Type here!₩nEnter your new article here!!"
```

예제 : simpleXYplotGui

- X변수의 선택
- Y변수의 선택
- 점의 색 선택
  - black/red/blue/green
- 선의 형태 선택
  - lm/lowess
- lowess에서 smoother span 선택
  - 0 에서 1 사이의 실수
- XY 산점도



simpleXYplotGui

- 기본 GUI 정의

```
> library(gWidgets)
> options("guiToolkit"="RGtk2")
> simpleXYplotGui<- function(sam.data) {
+
+      win <- gwindow("Scatter plot with lines")
+
+      gp <- ggroup(horizontal=FALSE, cont=win)
+
+     availVars <- colnames(sam.data)
```

simpleXYplotGui

- X / Y 변수 선택

```
+
+     tmp <- gframe("X variable", container=gp, expand=TRUE)
+     Xvar <- gcombobox(availVars, cont=tmp,
+                       handler=updatePlot)
+
+     tmp <- gframe("Y variable", container=gp, expand=TRUE)
+     Yvar <- gcombobox(availVars, cont=tmp,
+                       handler=updatePlot)
```

simpleXYplotGui

- 점의 색 선택
  - black/red/blue/green
- 선의 형태 선택
  - lm/lowess

```
+     tmp <- gframe("Color of points", container=gp, expand=TRUE)
+     colchoose <- gradio(c("black","red","blue","green"), cont=tmp,
+                         handler =updatePlot)
+
+     tmp <- gframe("lines", container=gp, expand=TRUE)
+     linechoose <- gradio(c("lm","lowess"), cont=tmp,
+                         handler =updatePlot)
```

simpleXYplotGui

- lowess에서 smoother span 선택
  - 0 에서 1 사이의 실수
- 그래프를 위한 윈도우 설정

```
+     tmp <- gframe("lowess span", container=gp, expand=TRUE)
+     spanAdjust <- gslider(from=0,to=1,by=.01, value=0.5,
+                           cont=tmp, expand=TRUE, handler =updatePlot)
+
```

simpleXYplotGui

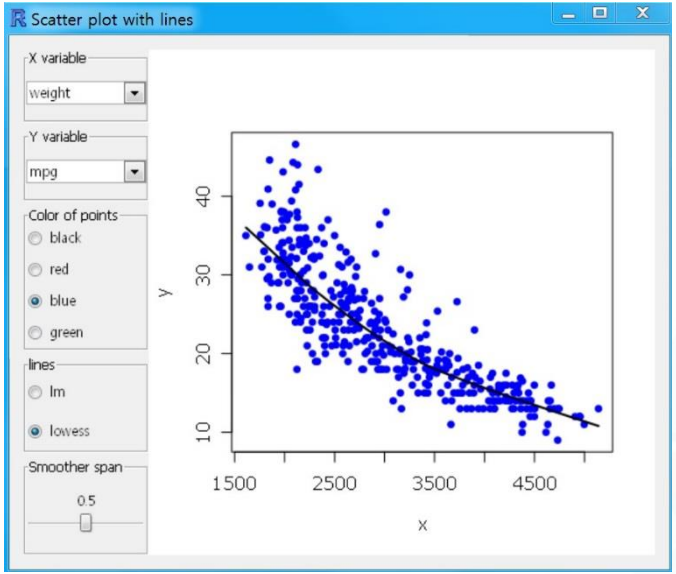
- updatePlot : GUI 입력값에 따라 XY 산점도를 새롭게 그리는 이벤트를 다루는 함수

```
+     updatePlot <- function(h,...) {
+         x<-sam.data[,svalue(Xvar)]
+         y<-sam.data[,svalue(Yvar)]
+         plot(x,y,pch=16,col=svalue(colchoose))
+         if(svalue(linechoose)=="lm")
+         {  abline(lm(y~x),lwd=2)
+         } else
+         {  lines(lowess(x,y,f=svalue(spanAdjust)),lwd=2)
+         }
+     }
+      add(win, ggraphics())
+ }
```

simpleXYplotGui

- simpleXYplotGui 실행

```
> simpleXYplotGui(autompg)
```



다음시간 안내

고급 그래픽기법(2)