Lecture 8: Descriptive statistics with R, Part II: Categorical variables and ggplot

This lecture covers descriptive statistics and corresponding graphical representation for categorical variables.

Consistent with the previous lecture, we will use the "diabetes_1" dataset as our example, which contains four quantitative variables: glu, bp, bmi and age; one categorical variable: type.

```
#read in the dataset
mydata<-read.csv("diabetes_1.csv",header=T)
dim(mydata)
```

```
## [1] 332   5
```

```
head(mydata)
```

```
##   glu bp  bmi age type
## 1 148 72 33.6  50  Yes
## 2  85 66 26.6  31   No
## 3  89 66 28.1  21   No
## 4  78 50 31.0  26  Yes
## 5 197 70 30.5  53  Yes
## 6 166 72 25.8  51  Yes
```

```
#create another variable age_cat which is a categorical variable of age#
mydata$age_cat[mydata$age<=30]<-1
mydata$age_cat[mydata$age>30 & mydata$age<=50]<-2
mydata$age_cat[mydata$age>50 ]<-3
mydata$age_cat=factor(mydata$age_cat,levels=c(1,2,3),labels=c("<=30","30-50",
">50"))
```

Now the dataset mydata contains four quantitative variables (glu, bp, bmi and age) and two categorical variables (type and age_cat).

8.1 Numerical representation

Categorical data are usually described in tabular format.

```
table(mydata$age_cat)
```

```
##
##  <=30 30-50   >50
##   205   105    22
```

Relative frequencies in a table are expressed as proportions of the row or column totals. Tables of relative frequencies can be constructed using the following two approaches:

```r
#Method 1
round(table(mydata$age_cat)/length(mydata$age_cat),2)
```

```
##
##  <=30 30-50   >50
##  0.62  0.32  0.07
```

```r
#Method 2
mytab<-table(mydata$age_cat)
round(prop.table(mytab),2)
```

```
##
##  <=30 30-50   >50
##  0.62  0.32  0.07
```

If we want to investigate the bivariate frequency distribution of two variables:

```r
df<-table(mydata$age_cat,mydata$type)
print(df)
```

```
##
##          No Yes
##   <=30  160  45
##   30-50  53  52
##   >50    10  12
```

```r
prop.table(df)
```

```
##
##                 No        Yes
##   <=30  0.48192771 0.13554217
##   30-50 0.15963855 0.15662651
##   >50   0.03012048 0.03614458
```

```r
prop.table(df,1)
```

```
##
##                No       Yes
##   <=30  0.7804878 0.2195122
##   30-50 0.5047619 0.4952381
##   >50   0.4545455 0.5454545
```

```r
#Note that the rows (1st index) sum to 1
prop.table(df,2)
```

```
##
##                 No        Yes
##   <=30  0.71748879 0.41284404
##   30-50 0.23766816 0.47706422
##   >50   0.04484305 0.11009174
```

```r
##Note that the columns (2nd index) sum to 1
```

Marginal tables:

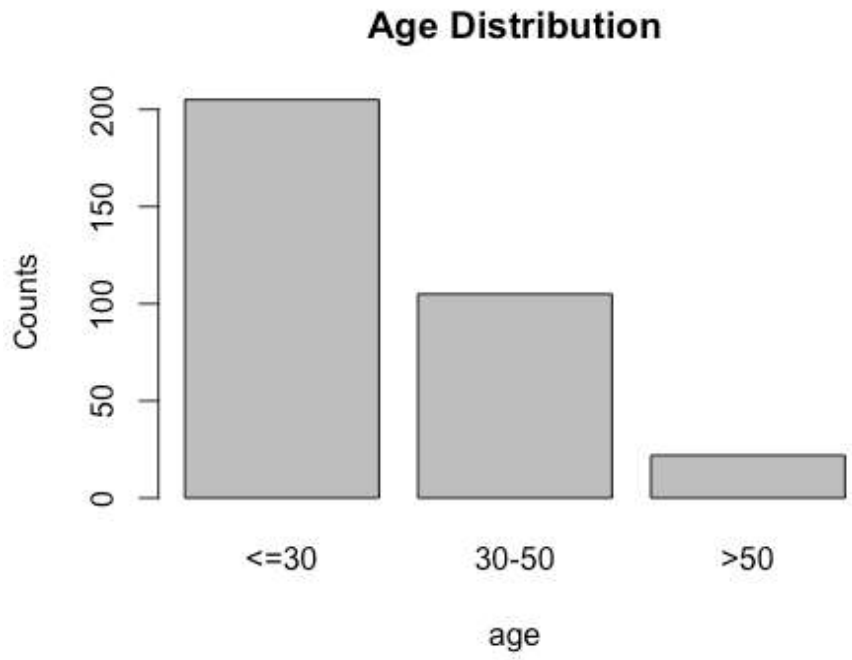
```
margin.table(df,1) #The second argument is the number of the marginal index:
1=display row totals

##
##  <=30 30-50   >50
##   205   105    22

margin.table(df,2)#The second argument is the number of the marginal index:
2=display column totals

##
##  No Yes
## 223 109
```

8.2 Graphical representation

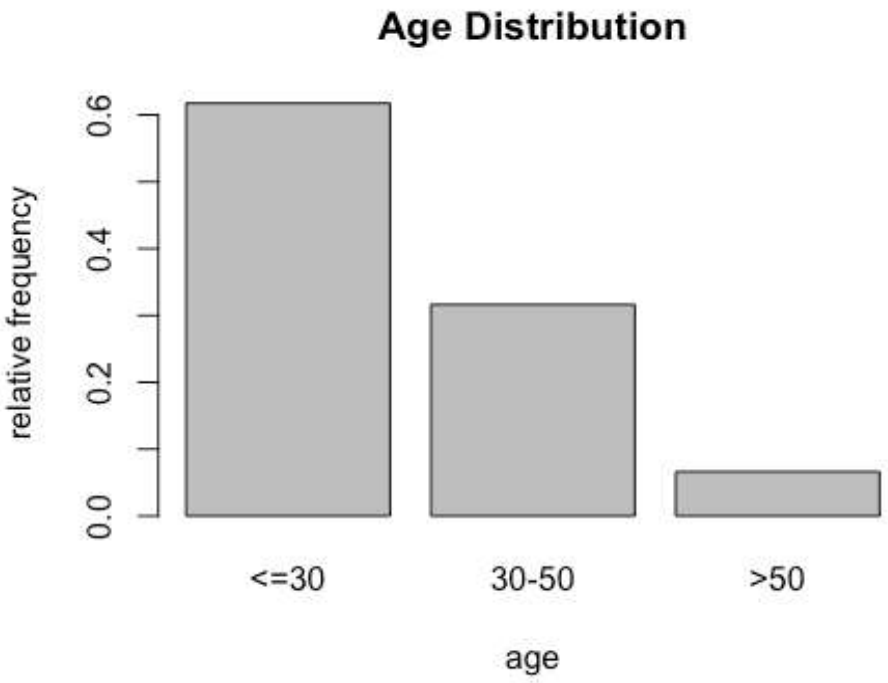
For presentation purposes, we may want to display a graph rather than a table of counts or percentages.

(1) Barplots

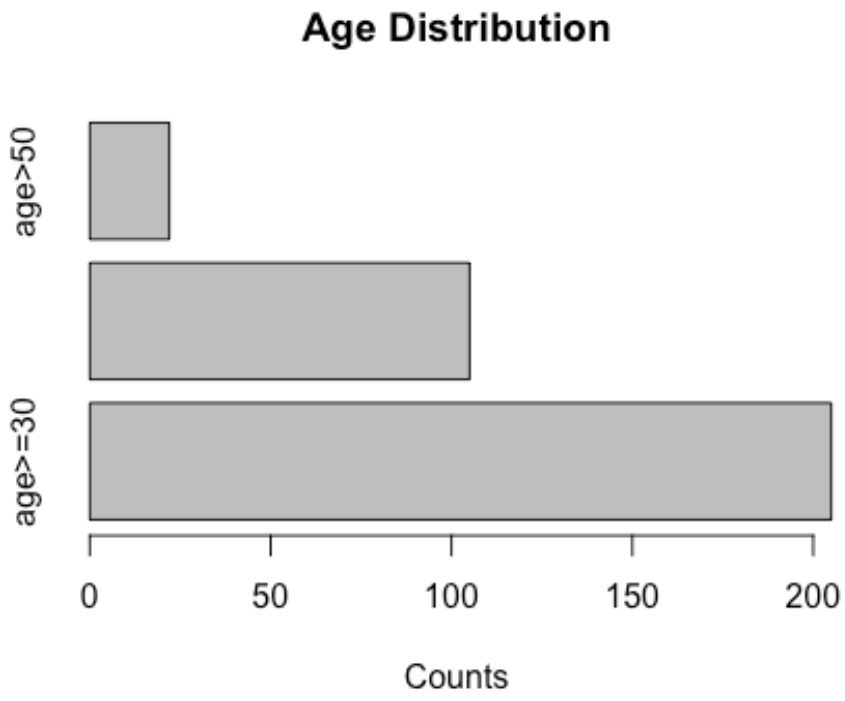
```
#Simple barplots for counts
counts <- table(mydata$age_cat)
barplot(counts, main="Age Distribution",
    xlab="age",ylab="Counts")
```



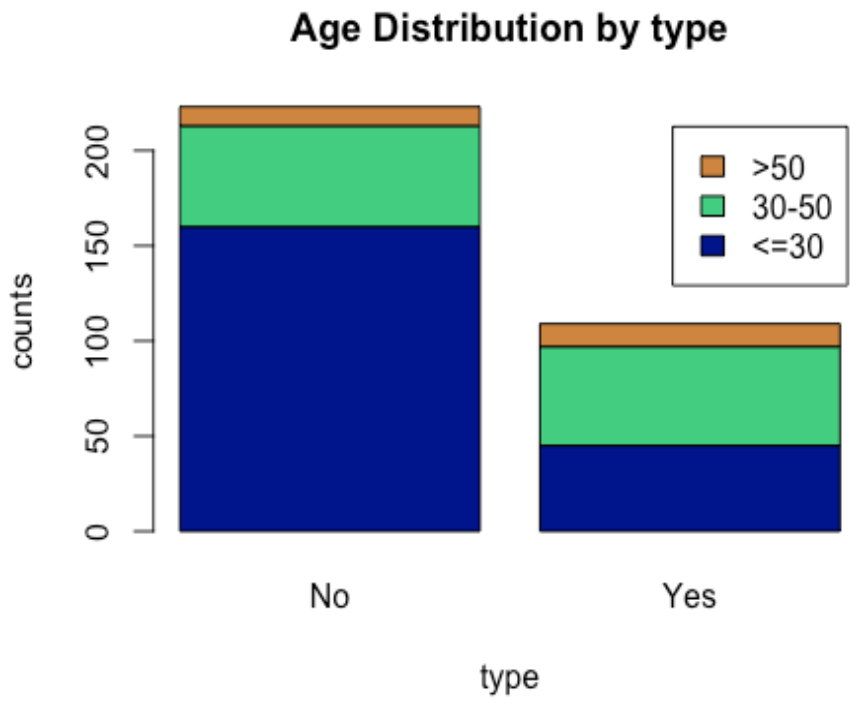
```r
#Simple barplots for frequency
freq <- prop.table(counts)
bp<-barplot(freq, main="Age Distribution",
    xlab="age",ylab="relative frequency")
```



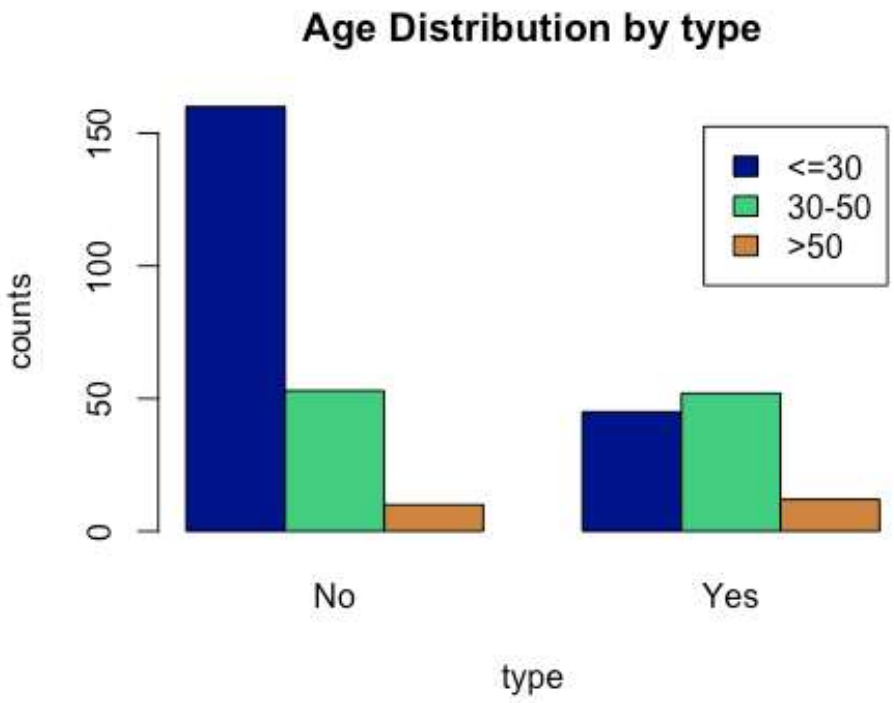

```r
# Simple horizontal barplot with added labels
counts <- table(mydata$age_cat)
barplot(counts, main="Age Distribution", horiz=TRUE,
  names.arg=c("age>=30", "30<age<=50", "age>50"),xlab="Counts")
```




```
# Stacked barplot with colors and legend
counts <- table(mydata$age_cat, mydata$type)
barplot(counts, main="Age Distribution by type",
 col=c("darkblue","seagreen3","tan3"),
    legend = rownames(counts),xlab="type",ylab="counts")
```



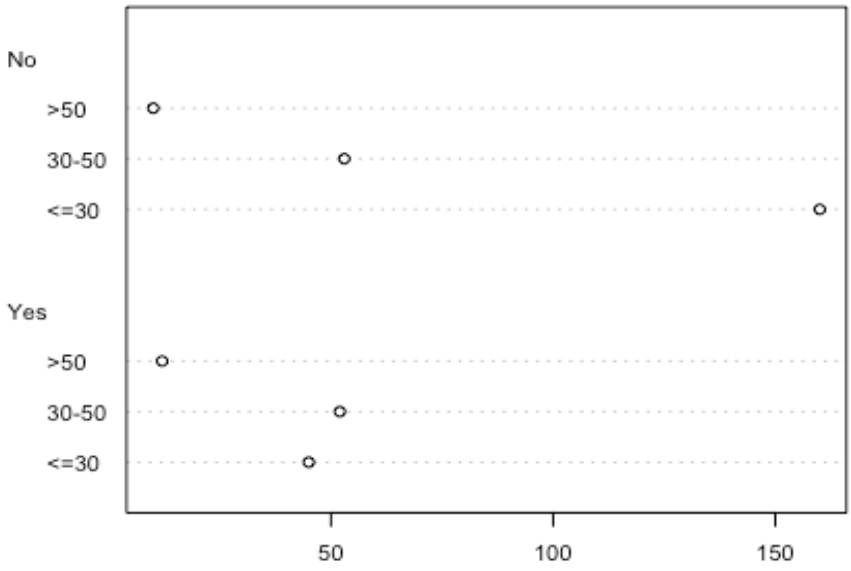

```
# Grouped barplot
counts <- table(mydata$age_cat, mydata$type)
barplot(counts, main="Age Distribution by type",
  col=c("darkblue","seagreen3","tan3"),
    legend = rownames(counts), beside=TRUE,xlab="type",ylab="counts")
```



(2) Dotcharts

Dotcharts contain the same information as barplots with beside=T, but they look quite different:

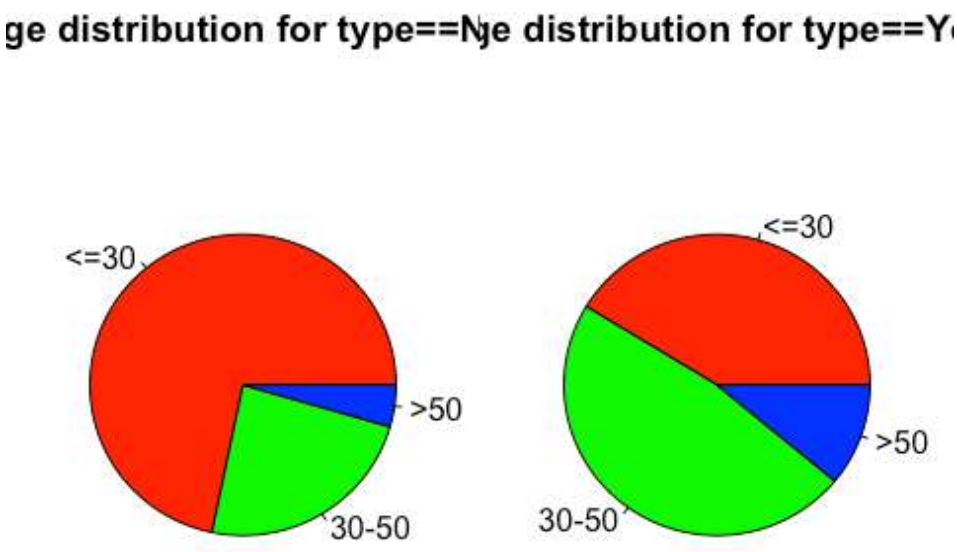
```
counts <- table(mydata$age_cat, mydata$type)
dotchart(counts,labels=row.names(counts),cex=.7,lcol="gray")
```



(3) Piecharts

Pie charts are created with the function **pie(x, labels=),** where "x" is a non-negative numeric vector indicating the area of each slice, and "labels=" specifies a character vector of names for the slices.

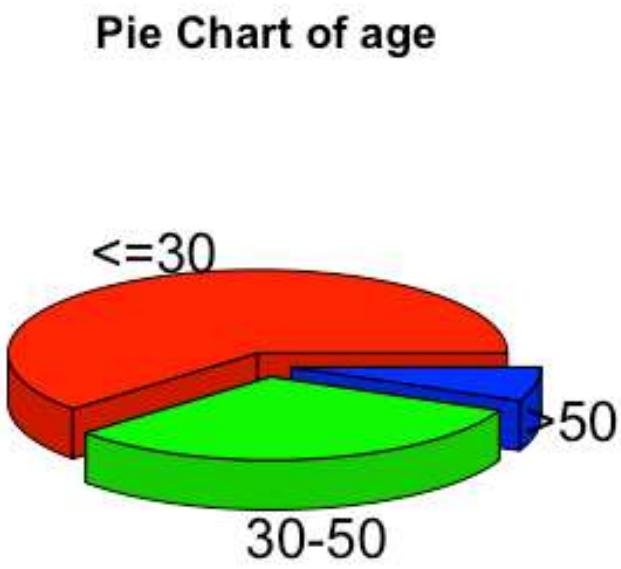
```
opar <- par(mfrow=c(1,2),mex=0.8, mar=c(1,1,2,1))
pie(table(mydata$age_cat[mydata$type=="No"]), main="Age distribution for type
==No", col=rainbow(3))
pie(table(mydata$age_cat[mydata$type=="Yes"]),main="Age distribution for type
==Yes", col=rainbow(3))
```




```
par(opar)
```

3D pie plots:

```
install.packages("plotrix")

# 3D Exploded Pie Chart
library(plotrix)

## Warning: package 'plotrix' was built under R version 3.2.5

counts <- table(mydata$age_cat)
lbls <- c("age>=30", "30<age<=50", "age>50")
pie3D(counts,labels=names(counts),explode=0.1,
    main="Pie Chart of age")
```



8.3 ggplot2 function

ggplot2() is a commonly-used package for graphing purposes in R. Compared to base graphics, ggplot2 offers more flexibility. The package author (Hadley Wickham) describes ggplot2 as: "a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics."

Please note that ggplot2() uses a different system for adding plot elements, and the data should always be in a dataframe.

```
install.packages("ggplot2")

#load the package
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.5
```

Graphics in ggplot2 begin with a call to ggplot(), which allows us to supply default data and aesthetic mappings (specified by **aes()**). We can then add layers, scales, coords, and facets with **+**.

```
ggplot      #Create a new plot
aes         #Construct aesthetic mappings
+.gg        #Add components to a plot
```

There are two major functions that we will use in ggplot2(): **qplot()** and **ggplot()**. Note that qplot() is for quick plots.

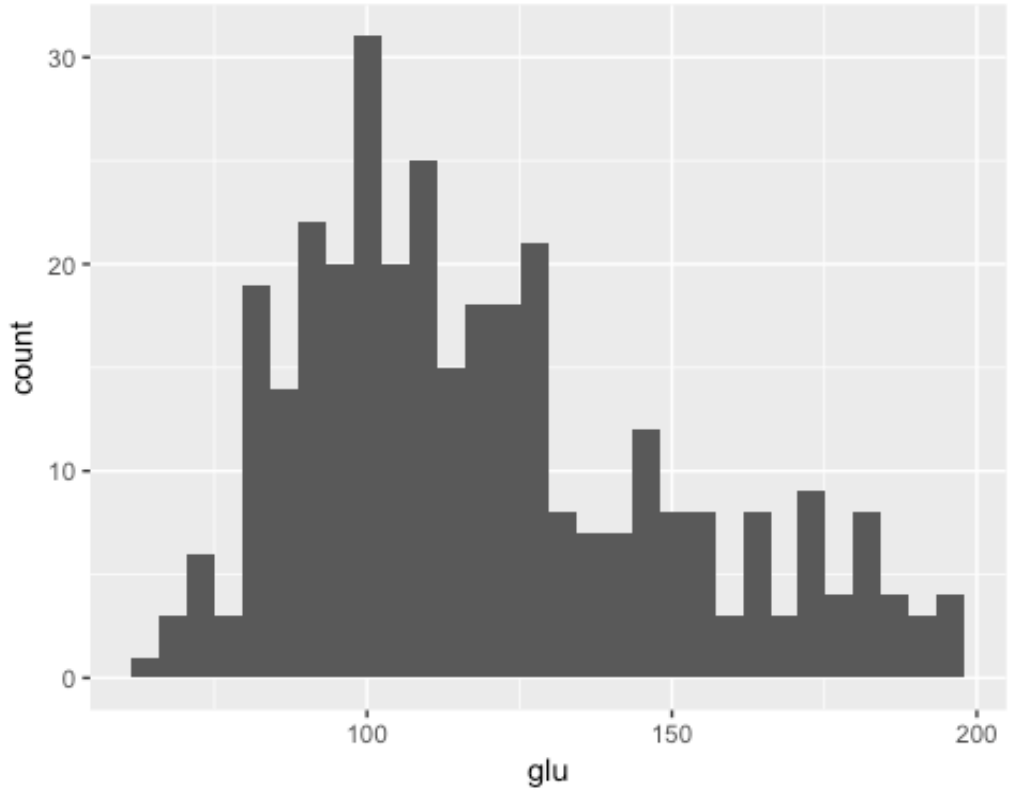
```
geom_point              #scatter plots, dot plots,etc
geom_line               #time series, trend line,etc
geom_histogram          #histogram
geom_violin             #violion plot
```

For more information, please refer to (http://ggplot2.tidyverse.org/reference/) and (http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html).

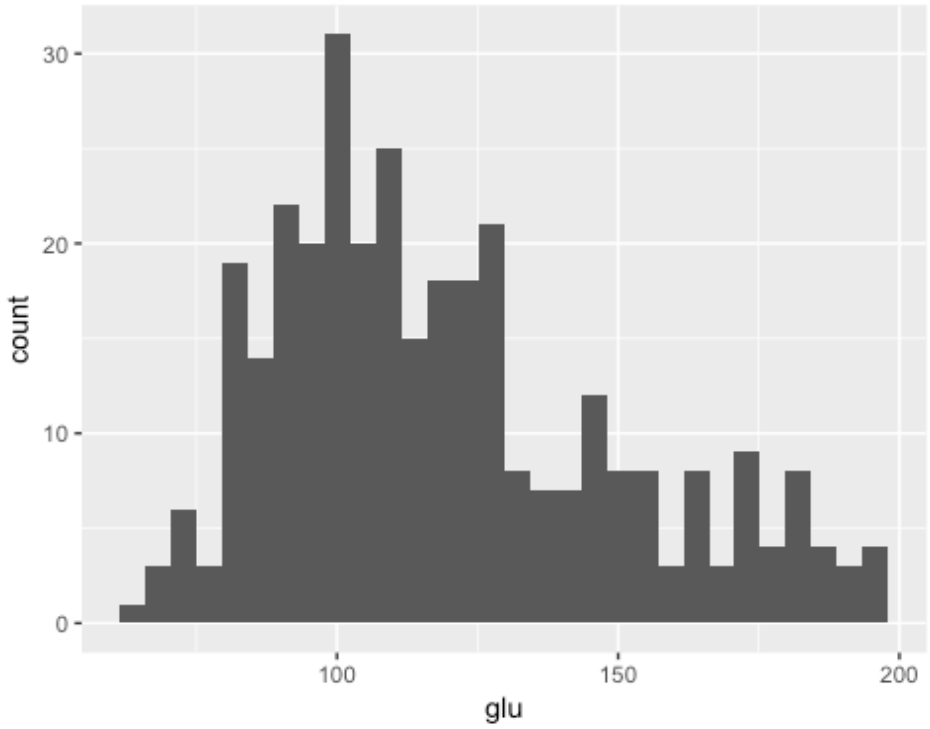
Some examples:

(1) A simple histogram for a single quantative variable

```
qplot(data=mydata,x=glu)#For a histogram, all we need to tell qplot() is whic
h dataframe to look in and which variable is on the x axis.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

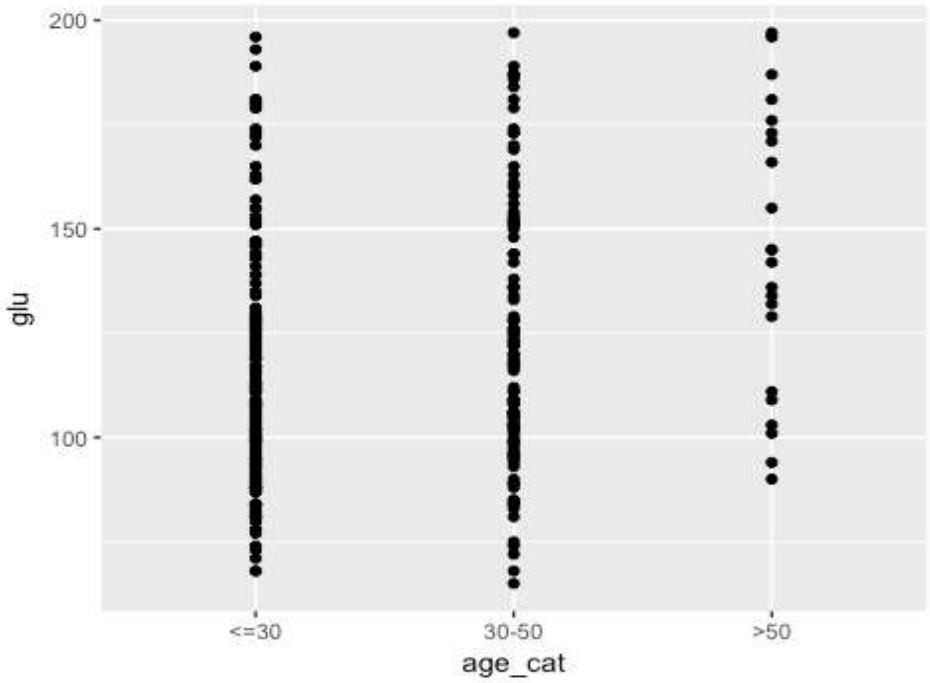



```
ggplot(mydata, aes(x = glu)) +geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better values with `binwidth`.
```

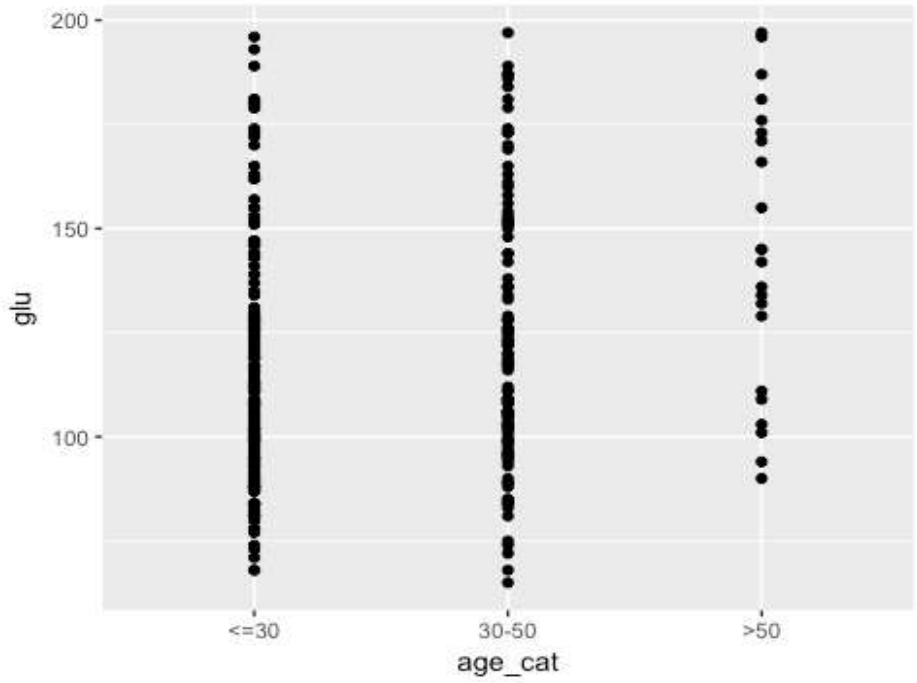


(2) For a quantitative variable by groups

```
# If we want to see how the raw values of glucose are distributed over differ
ent age groups:
qplot(data=mydata,x=age_cat,y=glu)
```

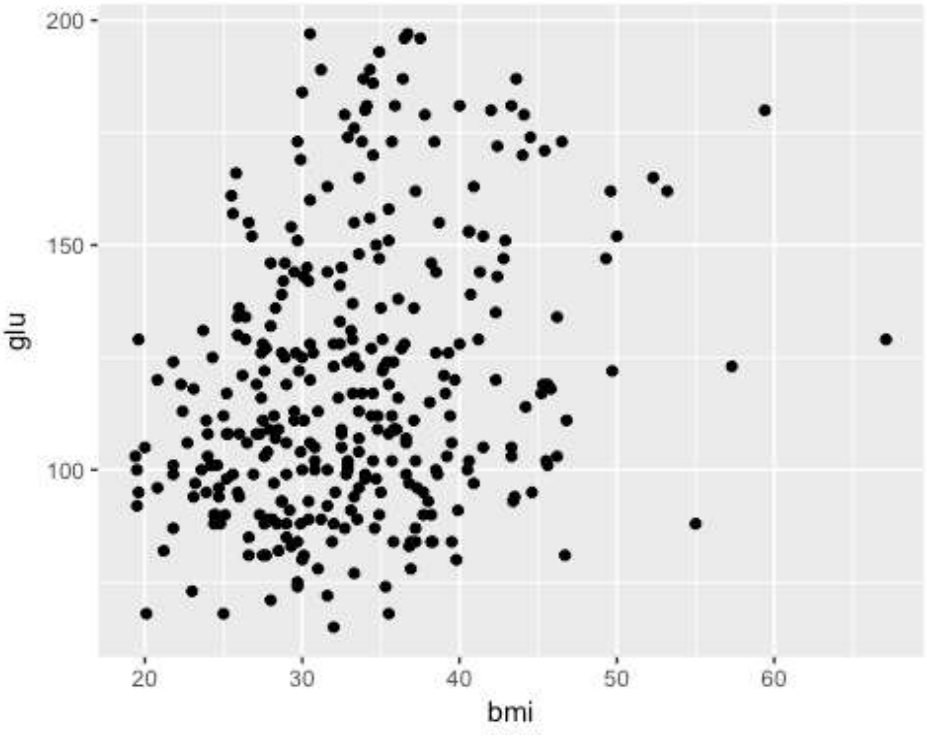



```
ggplot(mydata,aes(x=age_cat,y=glu))+geom_point()
```

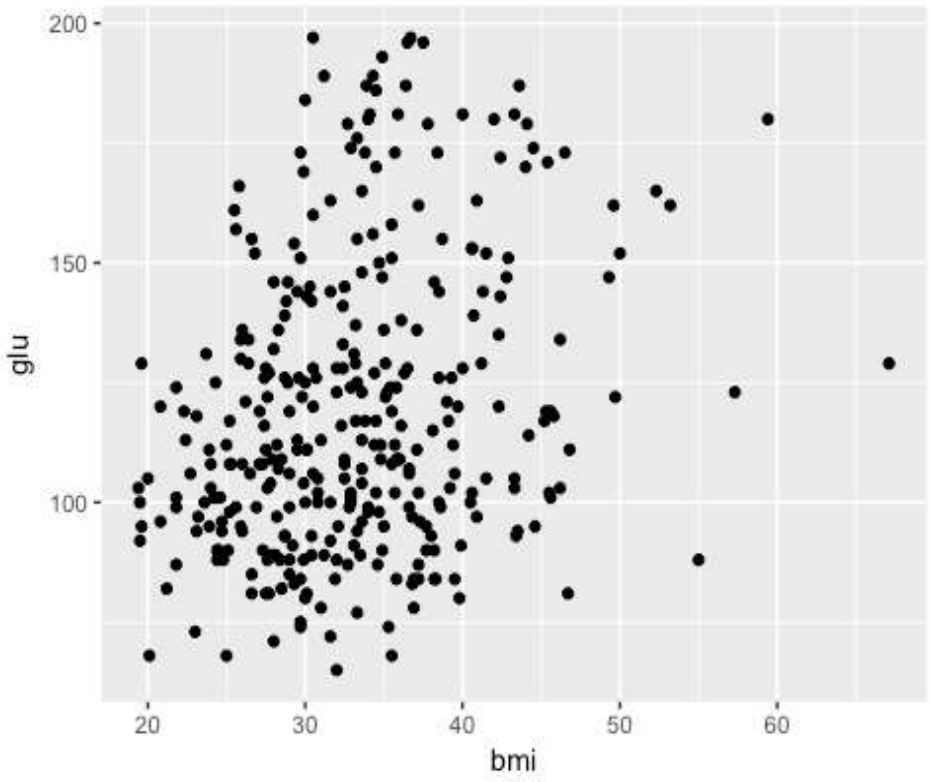


(3) For two quantitative variables

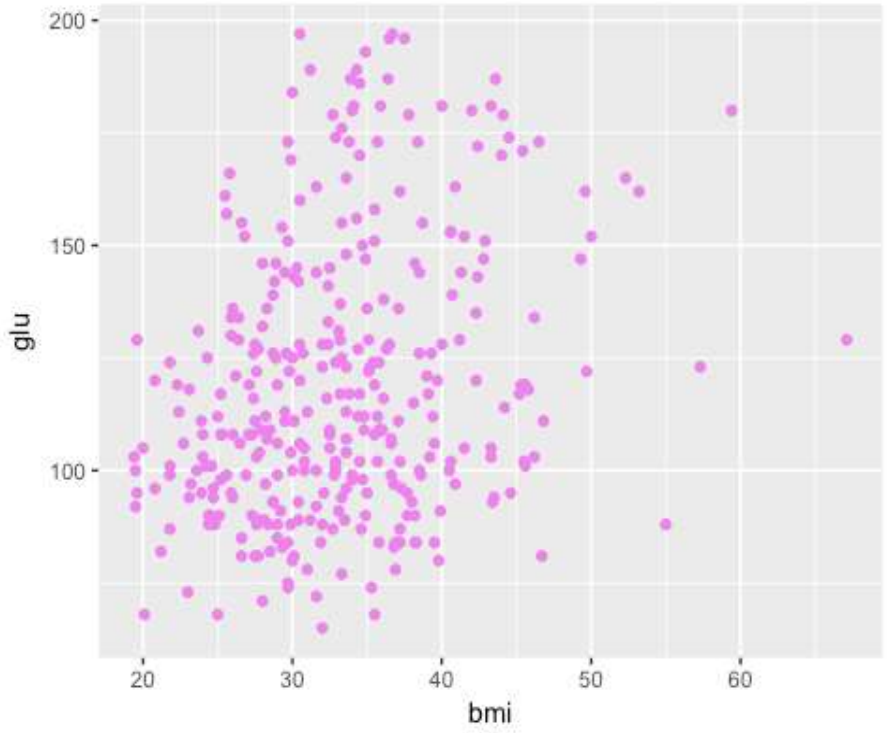
```
#a simple scatterplot
qplot(data=mydata,x=bmi,y=glu)
```




```
p1<-ggplot(mydata,aes(x=bmi,y=glu))
p1+geom_point()
```

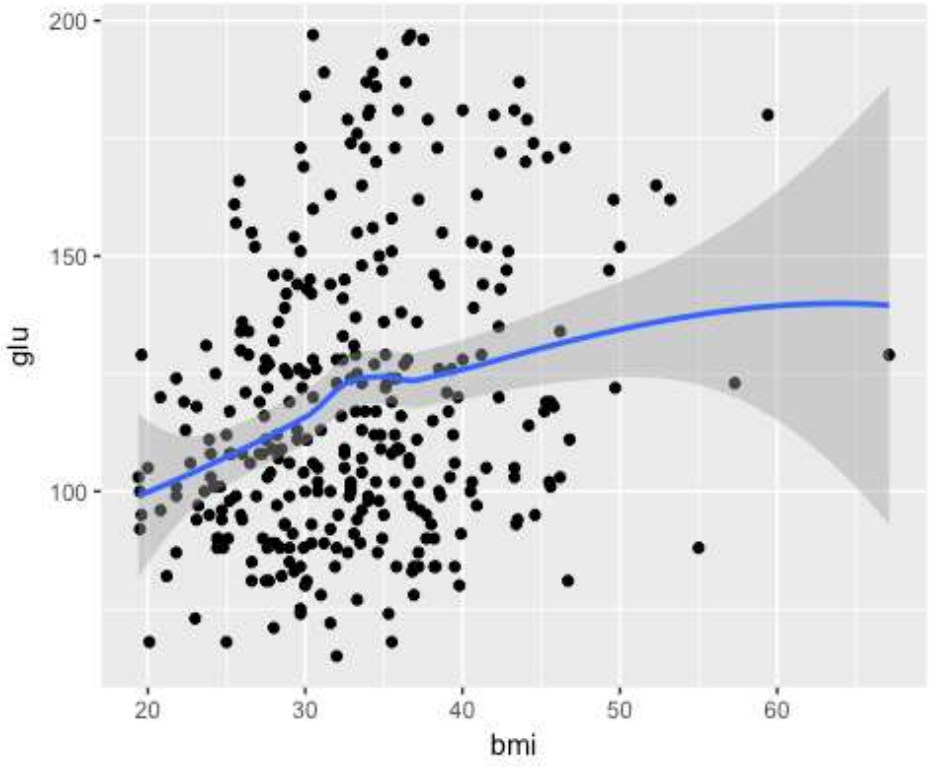



```
p1+geom_point(colour="violet")#colour: "outside" color
```

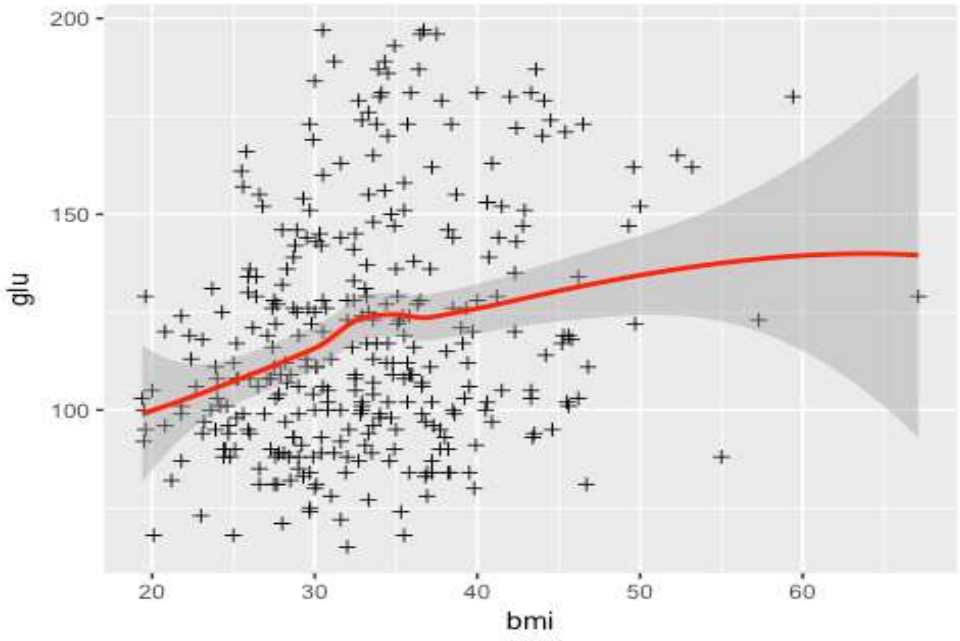



```
#add a nonparametric-regression smooth line
p1+geom_point()+geom_smooth()
```

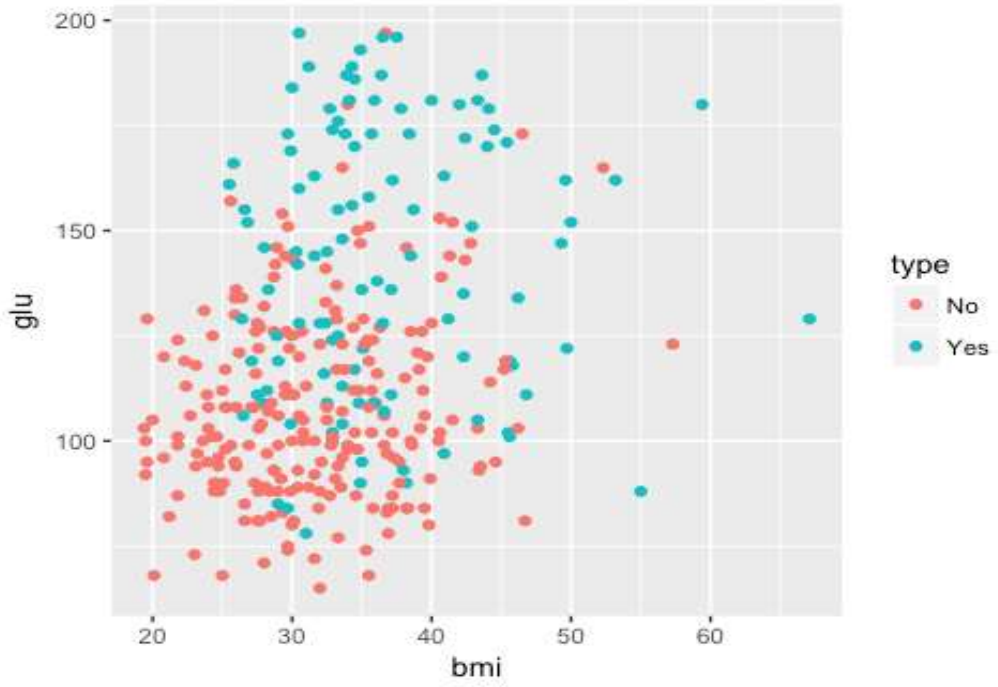
```
## `geom_smooth()` using method = 'loess'
```



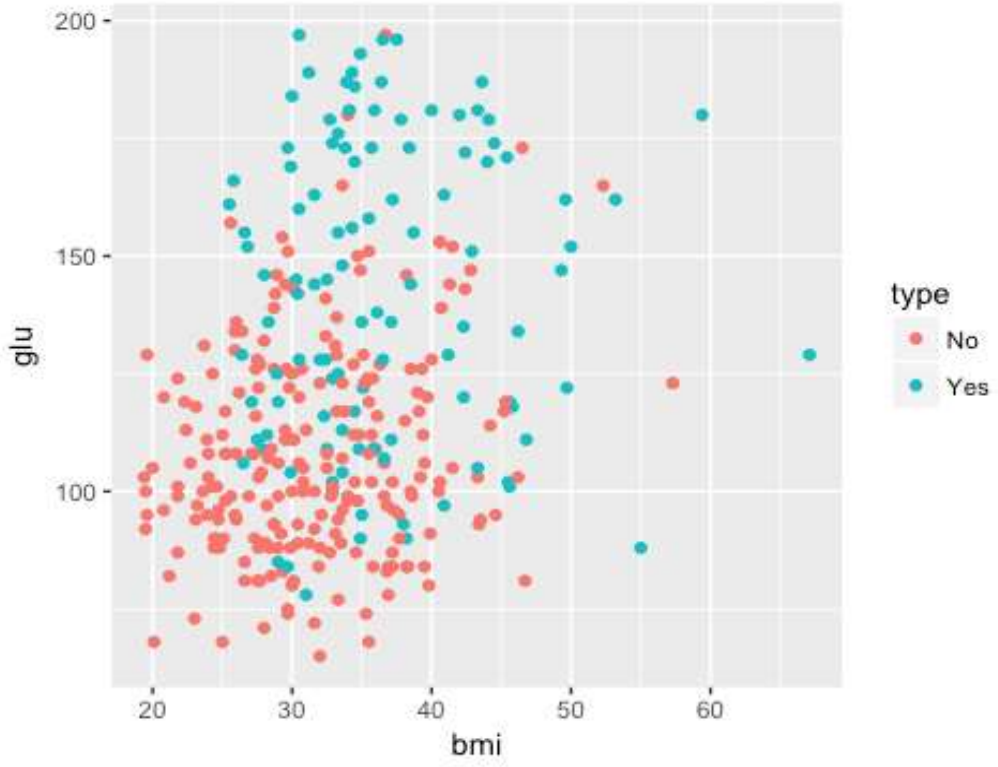

```
p1+geom_point(shape=3)+geom_smooth(colour="red")#shape: shape of the points

## `geom_smooth()` using method = 'loess'
```



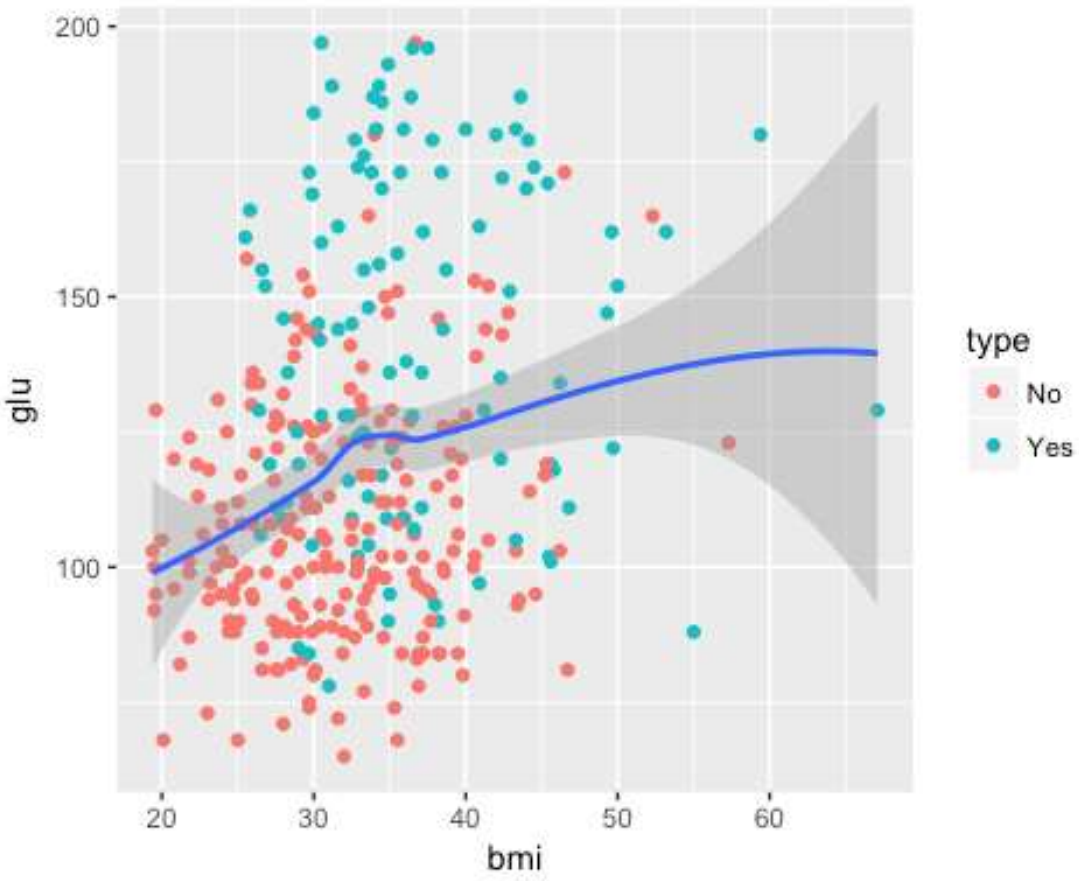

```
#a scatterplot by "type" category
qplot(data=mydata,x=bmi,y=glu,col=type)
```



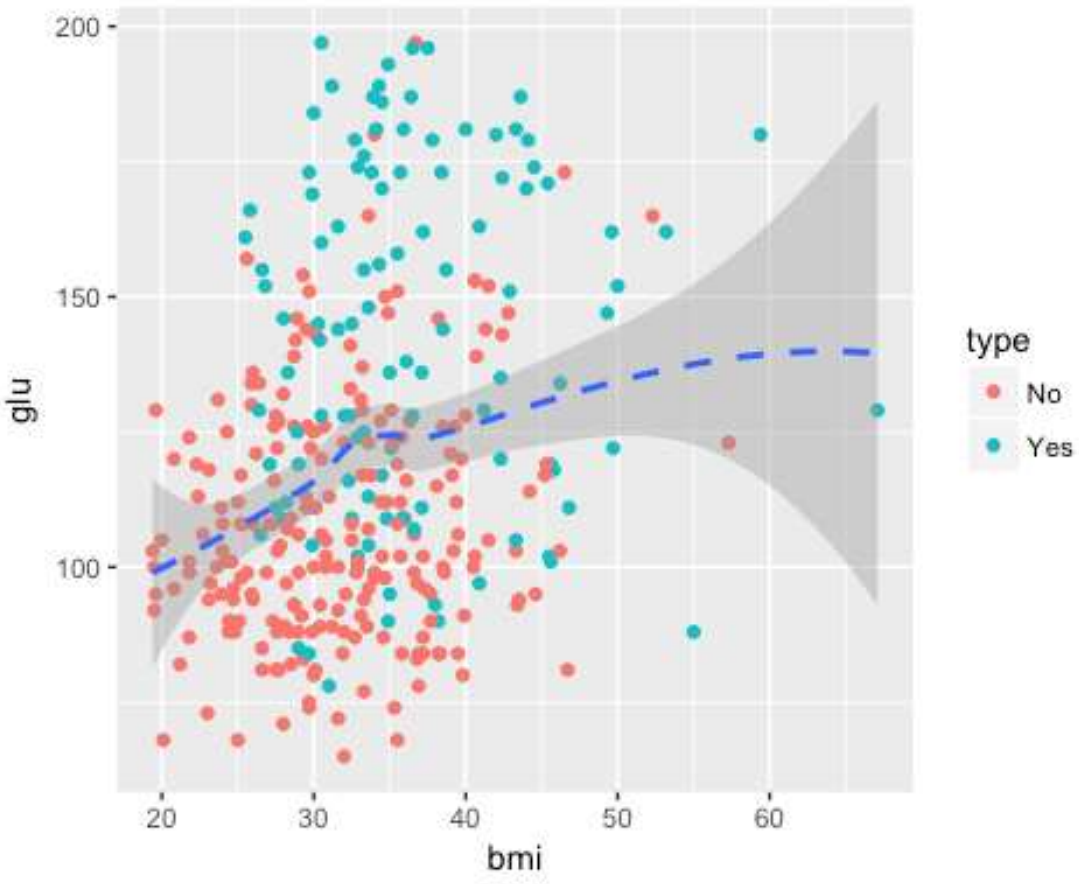

```
p1+geom_point(aes(color=type))
```




```
#add a smooth line
p1 +geom_point(aes(color = type)) +geom_smooth()

## `geom_smooth()` using method = 'loess'
```

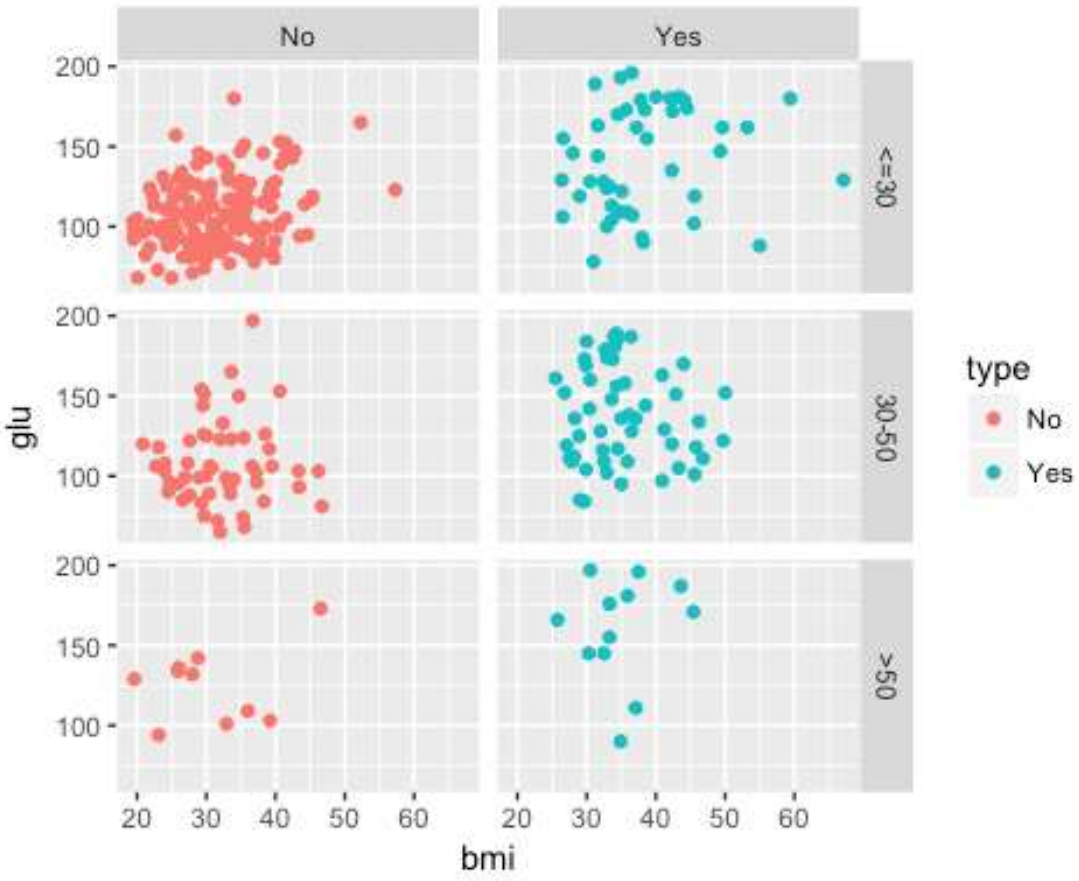



```
p1 +geom_point(aes(color = type)) +geom_smooth(linetype=2) #linetype: type of
 the line

## `geom_smooth()` using method = 'loess'
```

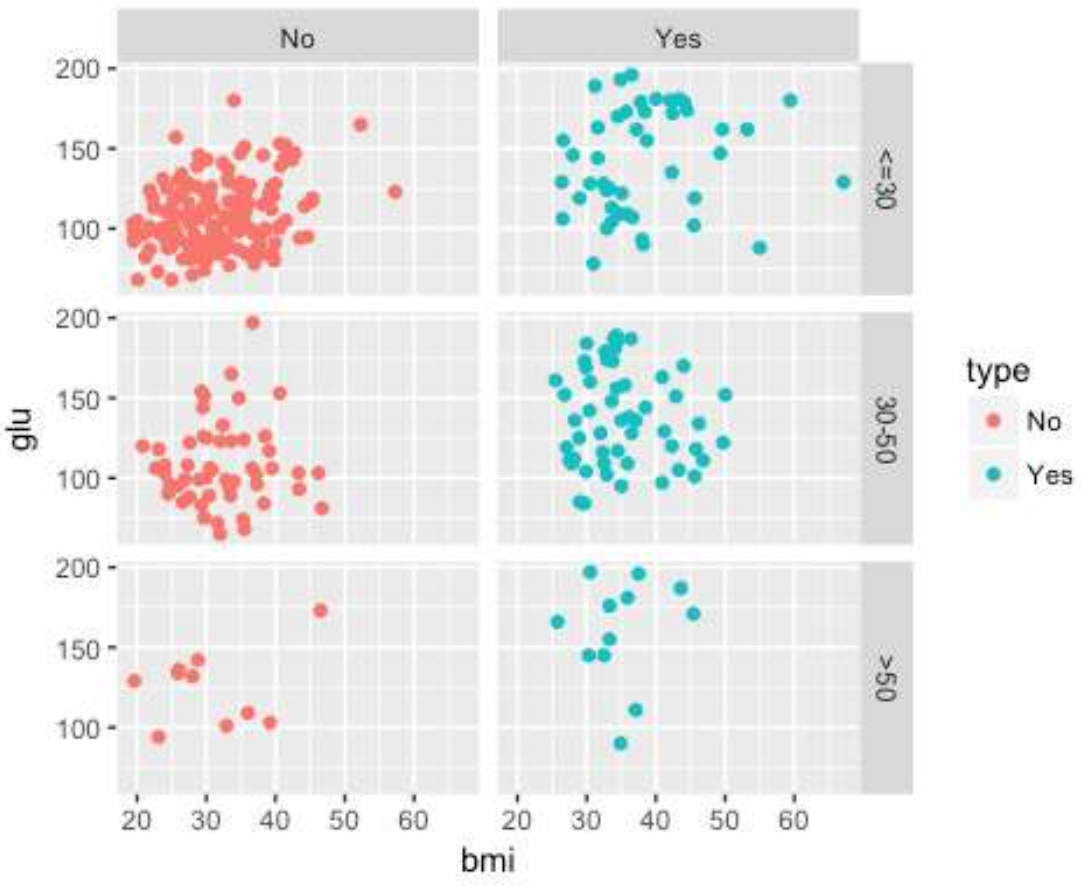


We can also plot the relationship between bmi and glucose by “type”, with each type separated into the various age categories.

```
qplot(data=mydata,x=bmi,y=glu,color=type,facets = age_cat~type)
```

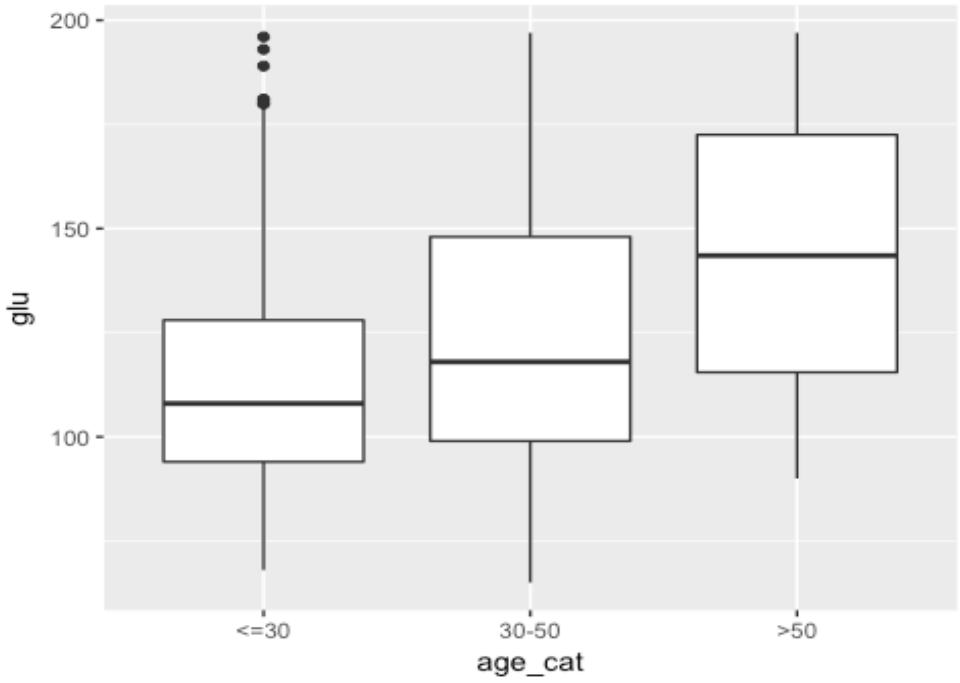



```
p1 +geom_point(aes(color = type))+facet_grid(age_cat~type)
```

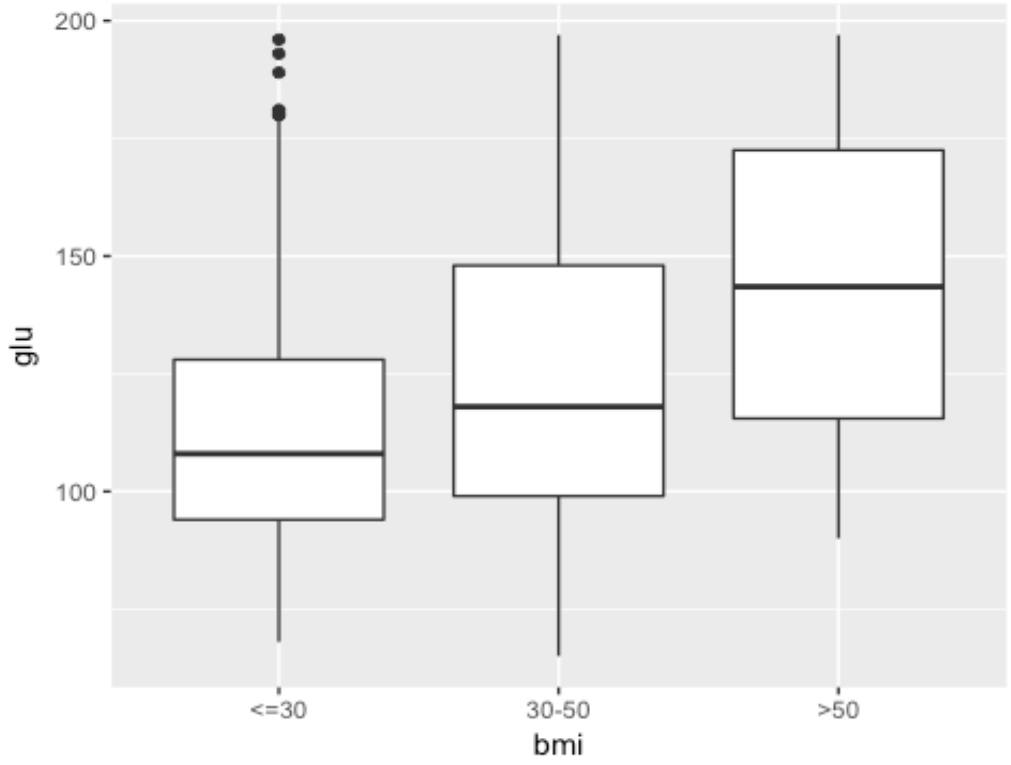


(4) For categorical variables

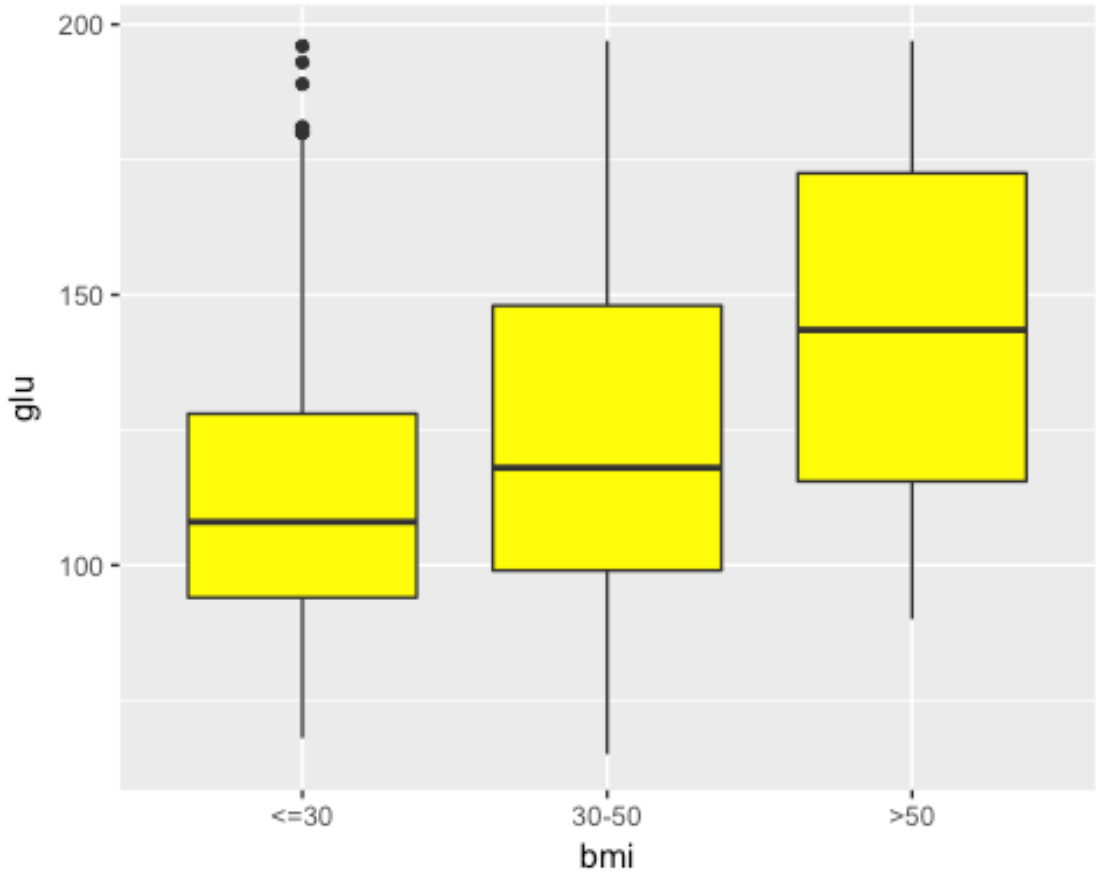
```
#a simple boxplot
qplot(data=mydata,x=age_cat,y=glu,geom="boxplot")
```



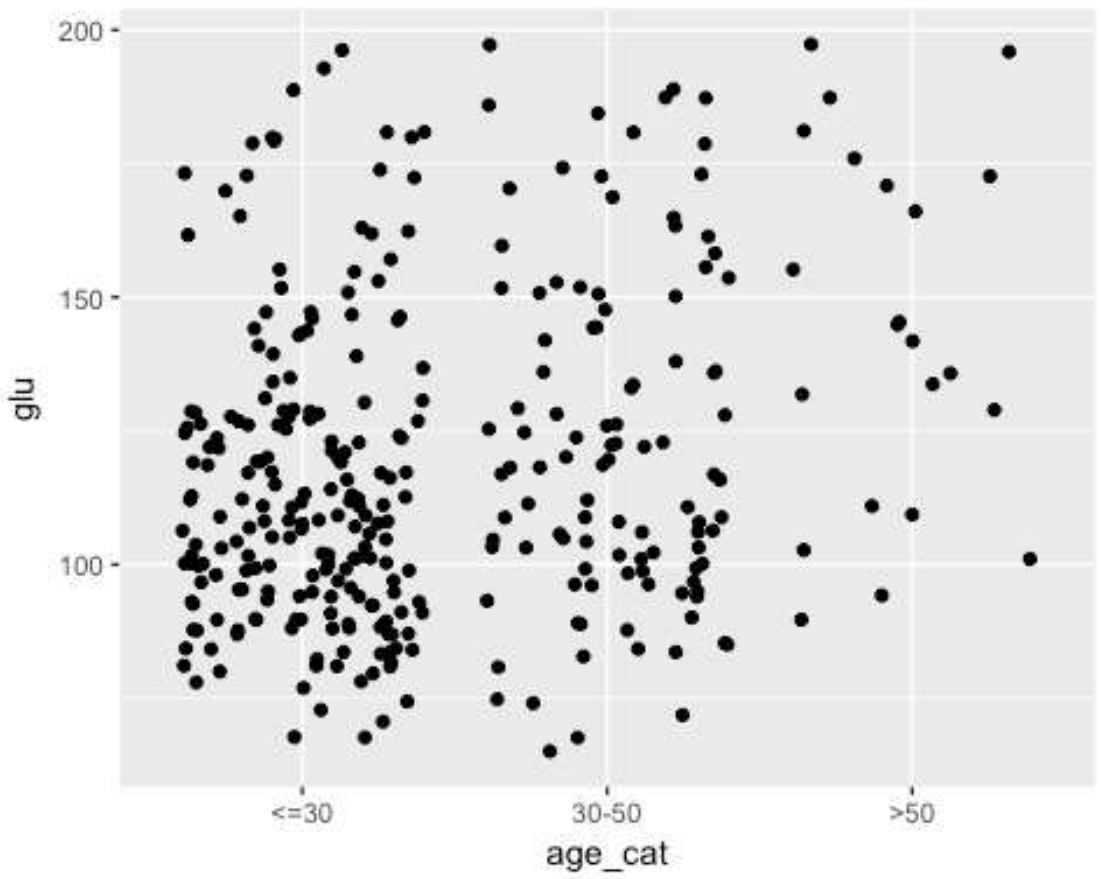

```
p1+geom_boxplot(aes(x=age_cat,y=glu))
```



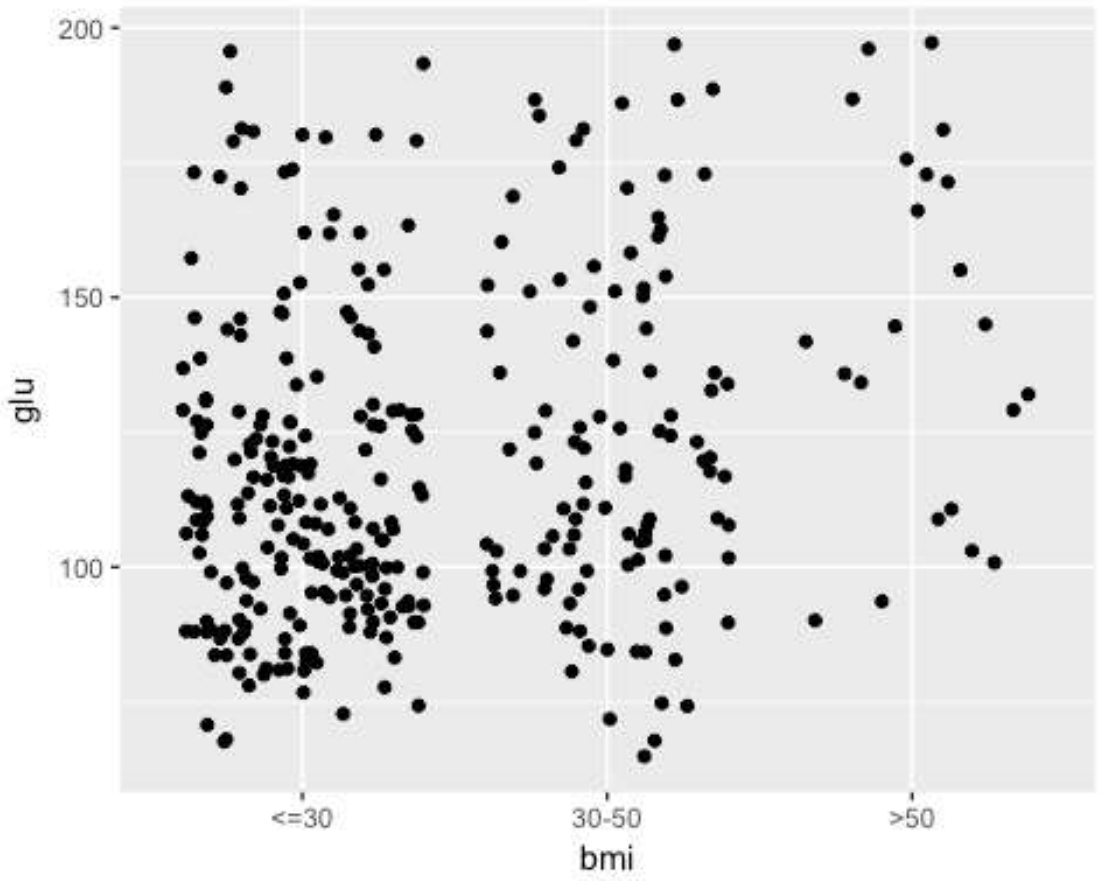

```
p1+geom_boxplot(aes(x=age_cat,y=glu),fill="yellow")#fill parameter: "inside" color
```




```
#a jitter plot
qplot(data=mydata,x=age_cat,y=glu,geom="jitter")
```




```
p1+geom_jitter(aes(x=age_cat,y=glu))
```


200
150
100
glu
<=30
30-50
>50
bmi