Visualization

R Studio

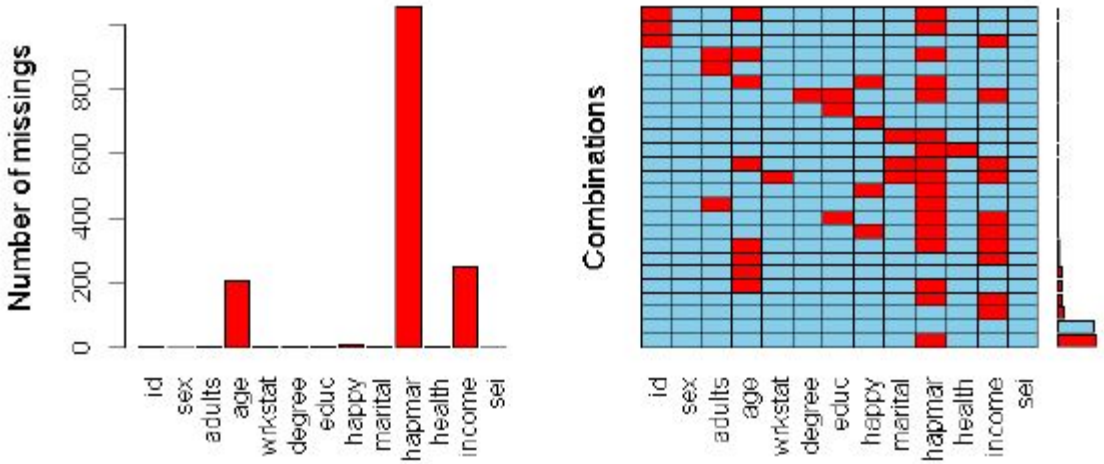
Visualize missing value patterns

#Load package

library(“VIM”)

library(“mice”)

aggr(myd, prop=FALSE, numbers=TRUE)



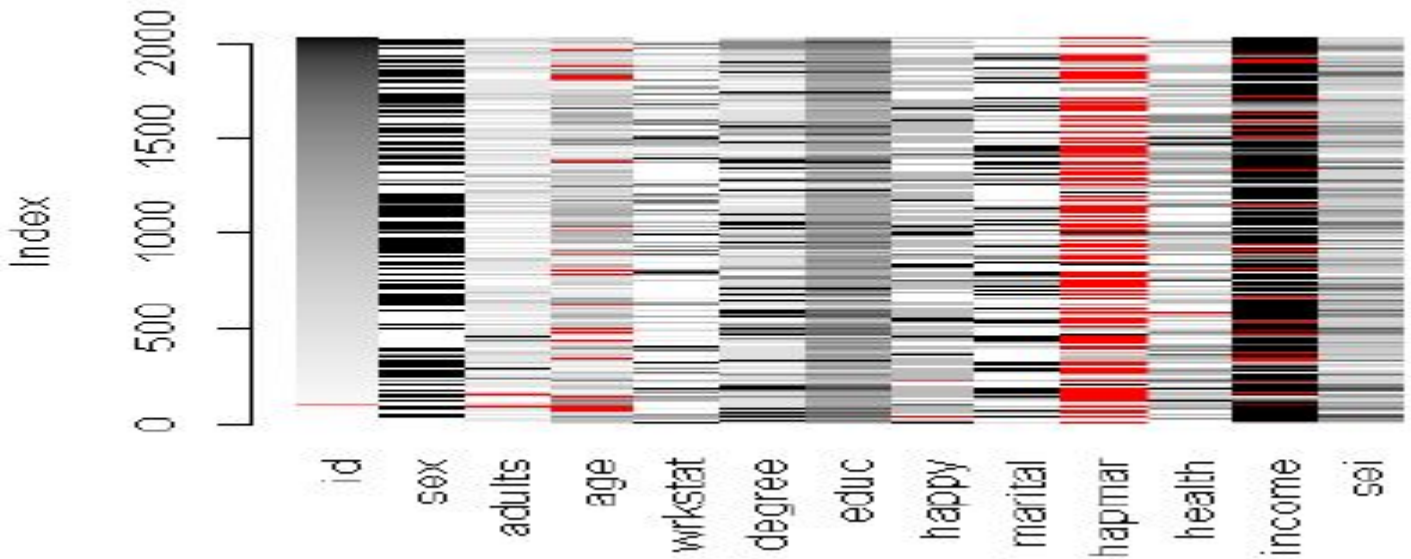
Matrixplot

#Load package

library(“VIM”)

library(“mice”)

#matrixplot(myd)



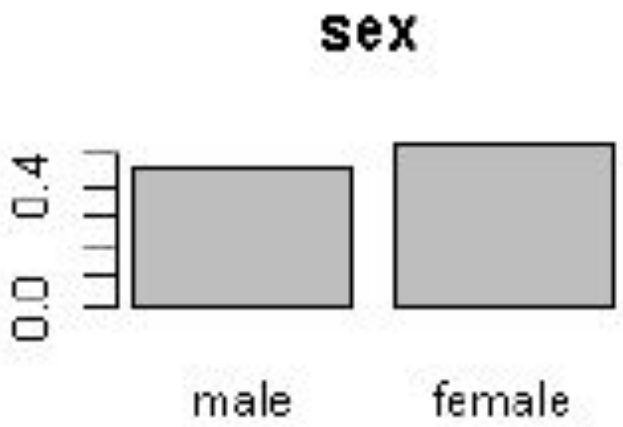
Red indicates missing values

Barcharts

par(mfrow=c(2,2))

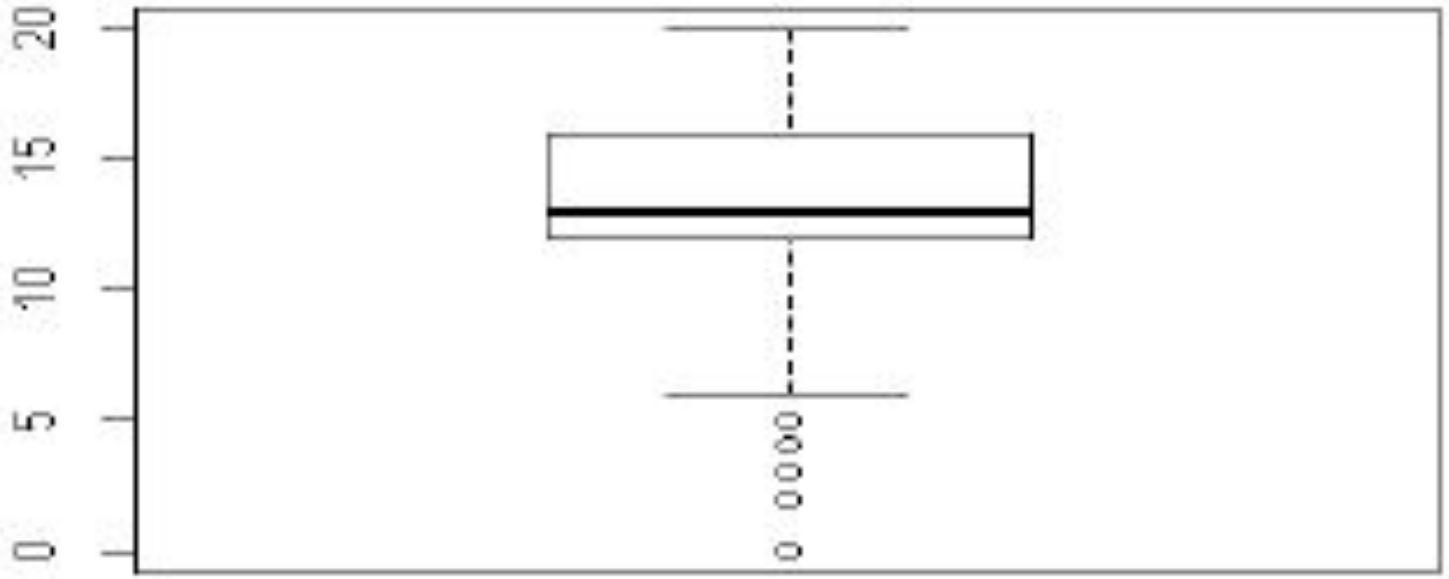
attach(myd)

barplot(prop.table(table(sex)), main="sex")



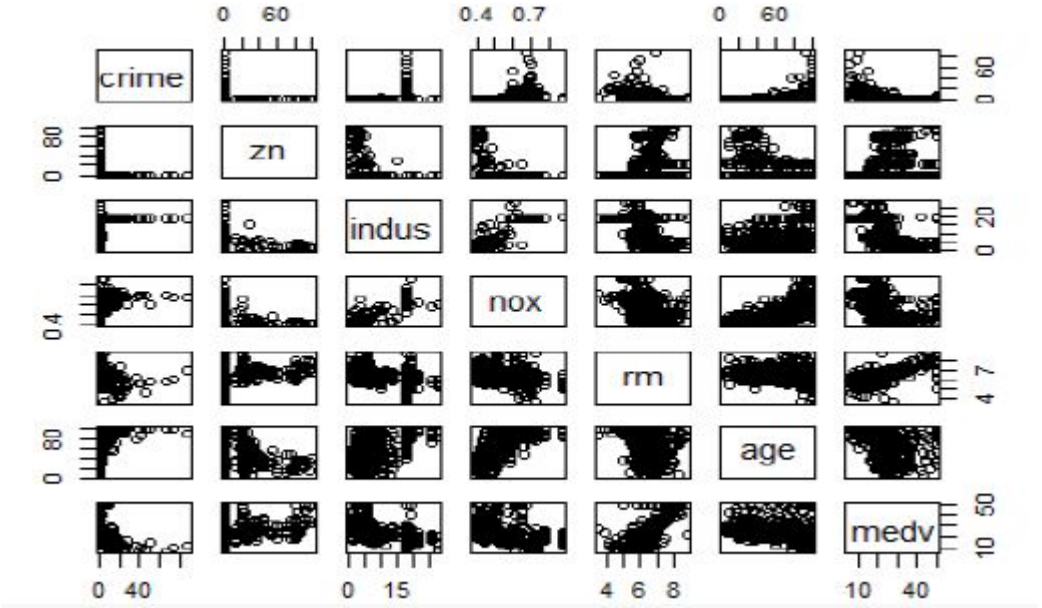
Boxplot

boxplot(myd$educ)



Scatterplot matrix

pairs(mydpc)



Correlation Matrix

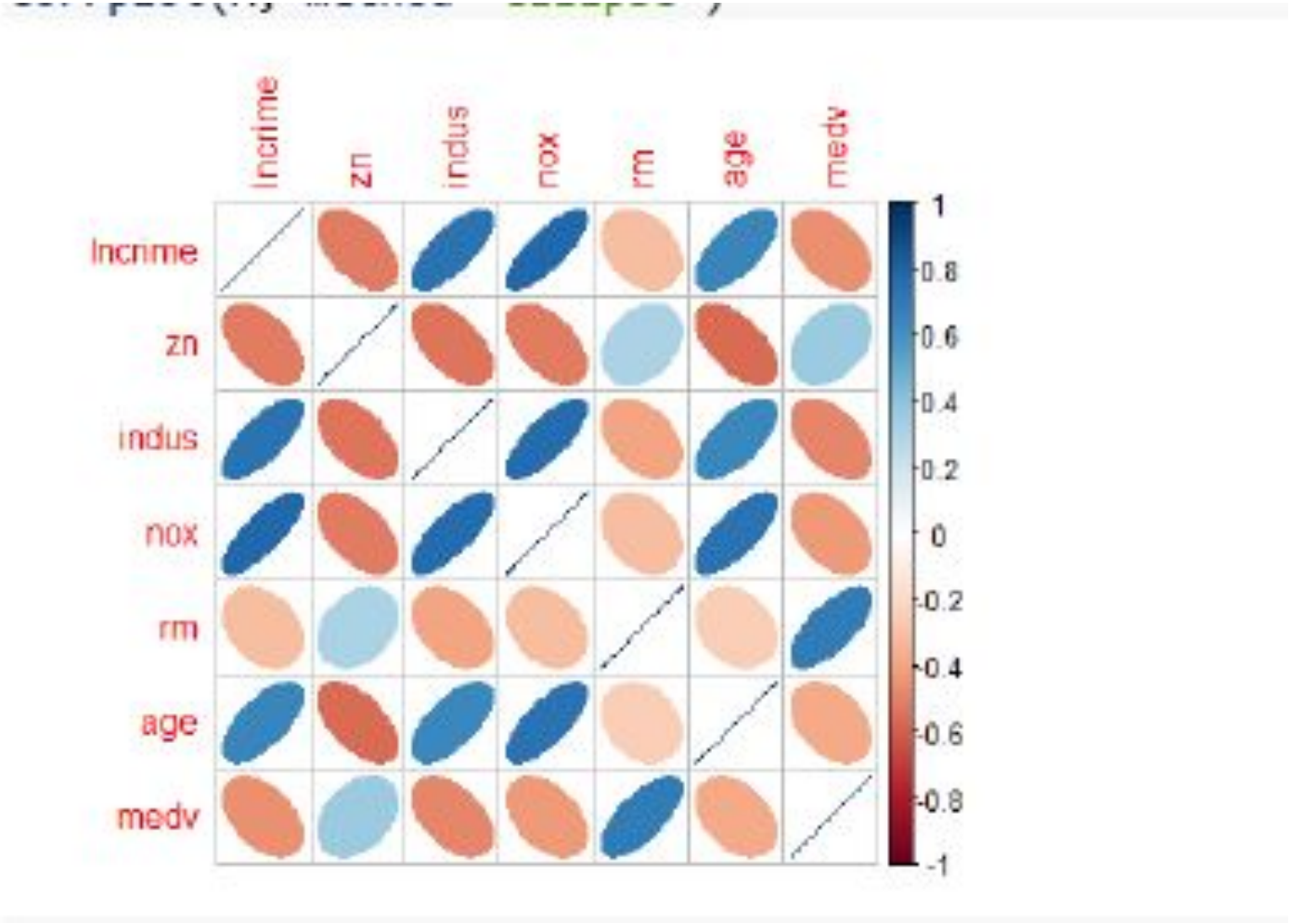
#Correlation Matrix

M=cor(mydpc)

#Plot Correlation Values

library(corrplot)

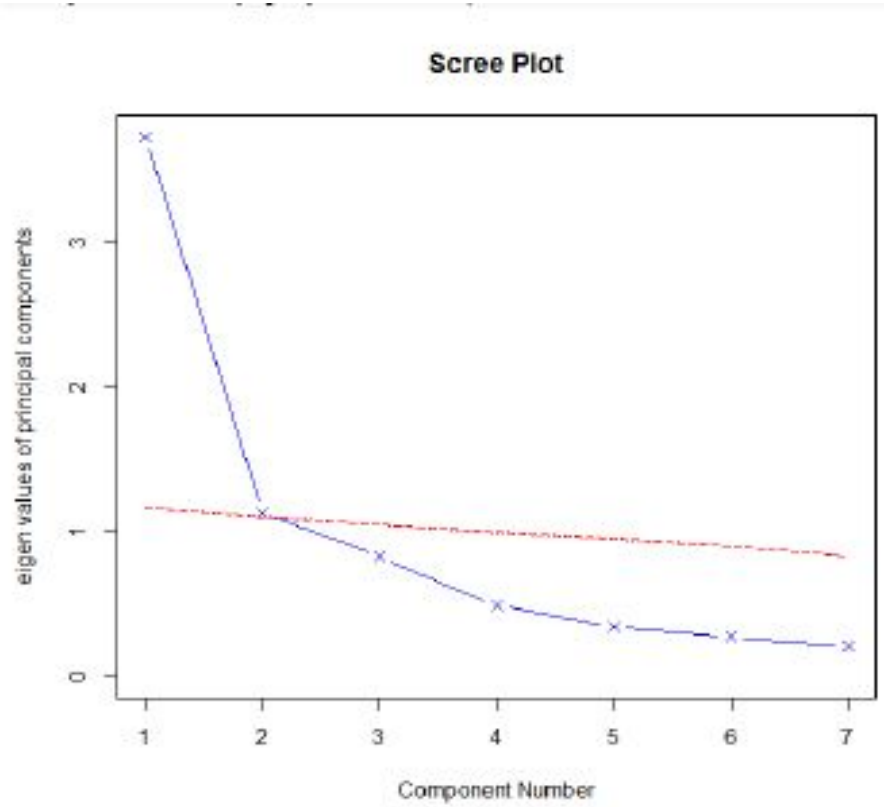
corrplot(M, method="ellipse")



Scree plot

#Scree plot

fa.parallel(mydpc, fa="pc", n.iter=100, show.legend=FALSE, main="Scree Plot")

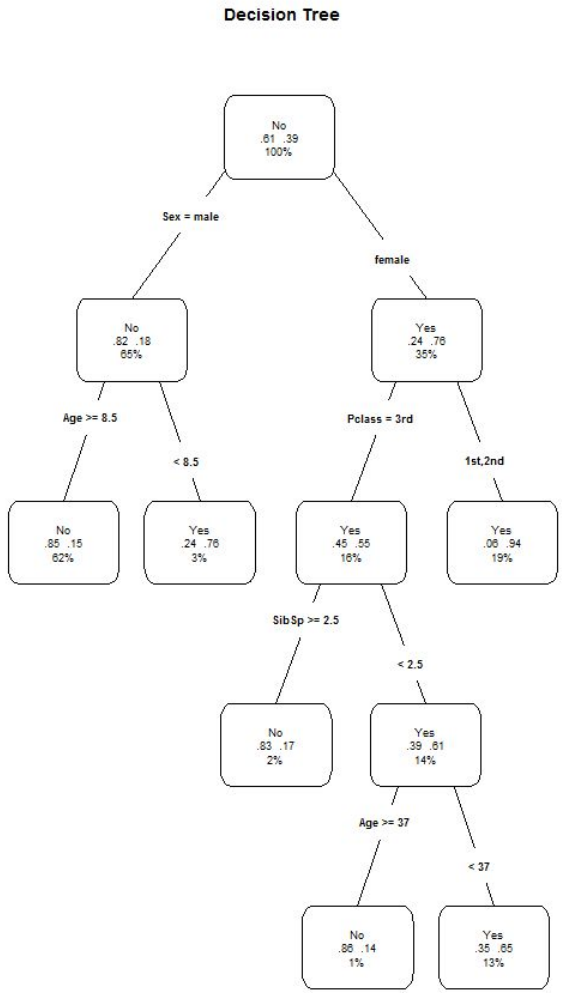


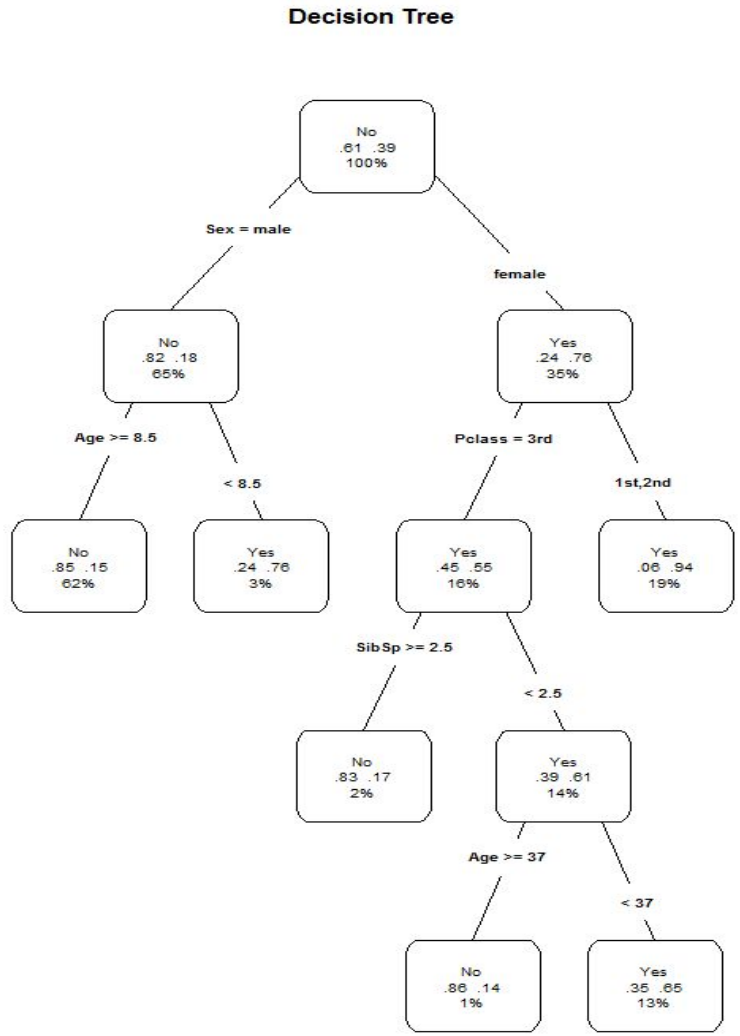

```
## Parallel analysis suggests that the number of factors =  NA  and the
number of components =  2
```

Decision Tree

dtree=rpart(Survived~., data=myddt.train, method="class", parms=list(split="gini"))

prp(dtree, type=4, extra=104, fallen.leaves=FALSE, main="Decision Tree", faclen=0)

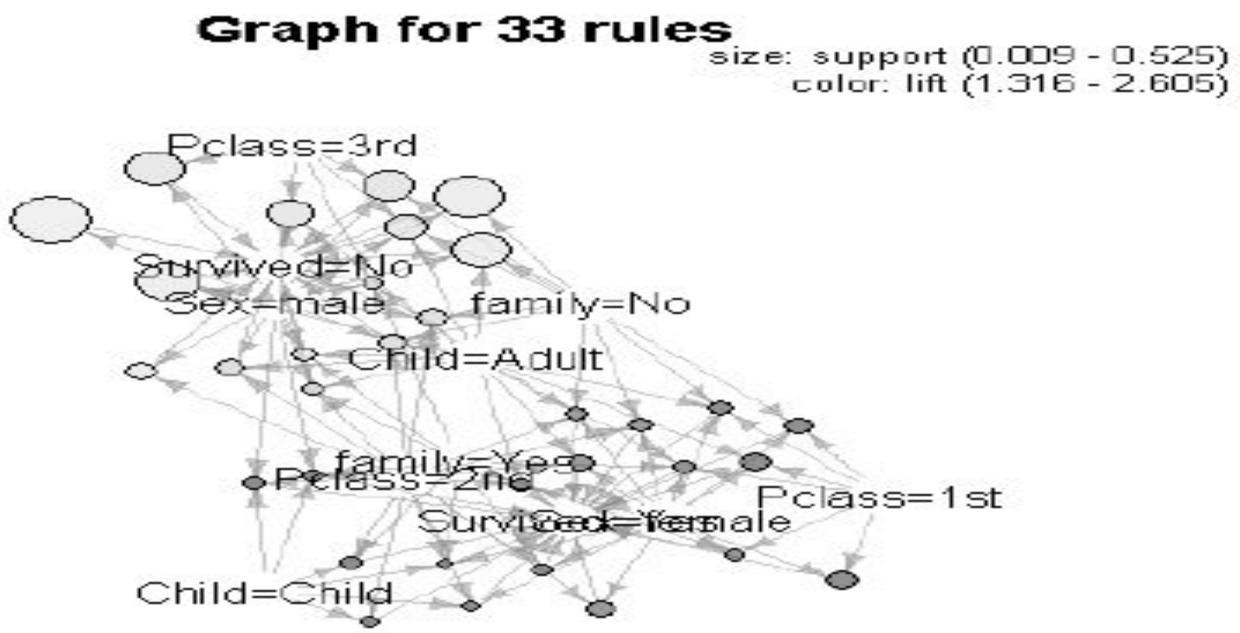




Associasion Rules

Library(arulesViz)

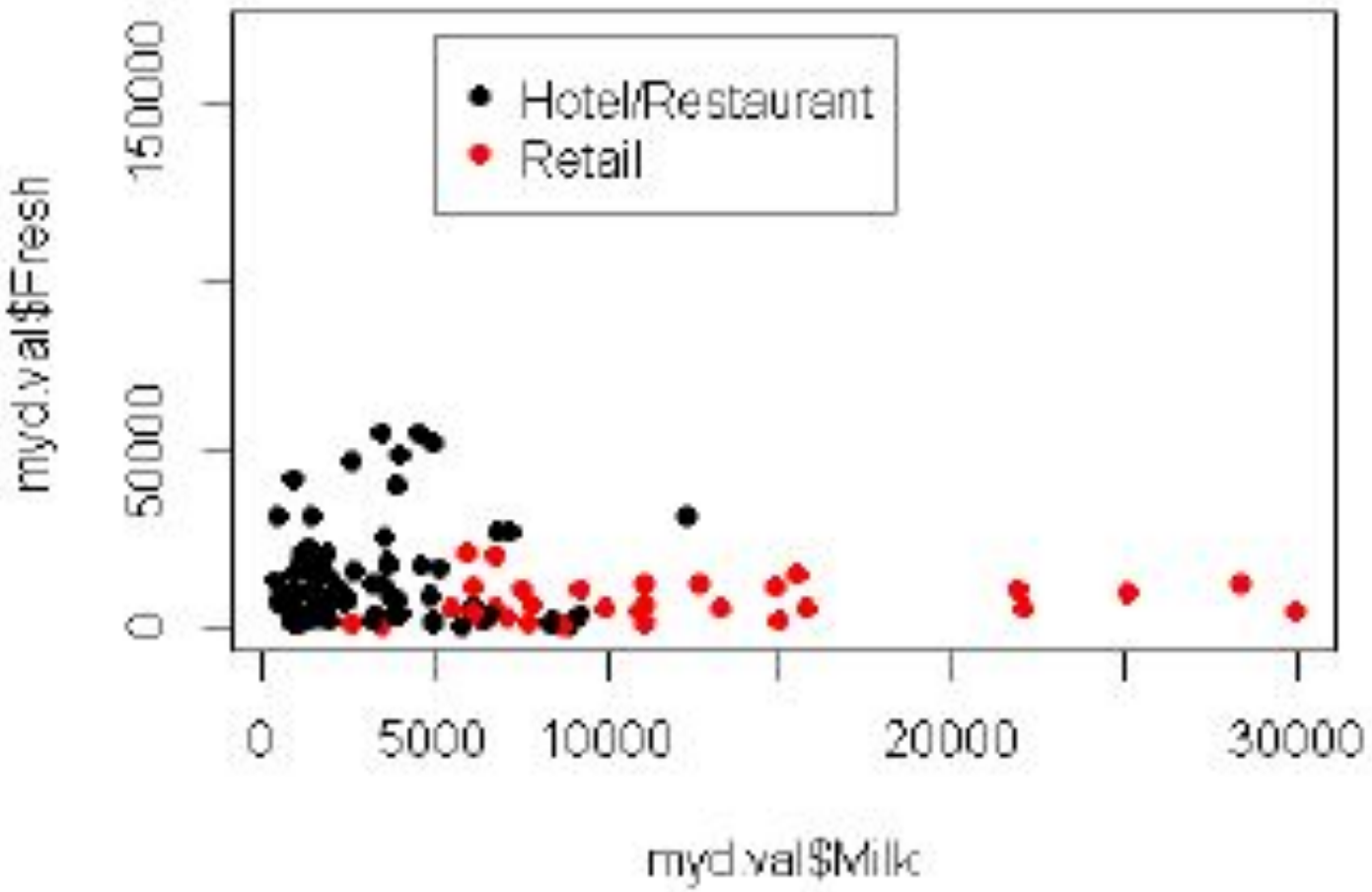
plot(rules, method="graph", control=list(type="items"))



Classification Visualization

plot(myd.val$Milk, myd.val$Fresh, col=as.numeric(pred), ylim=c(0, 170000), pch=16)

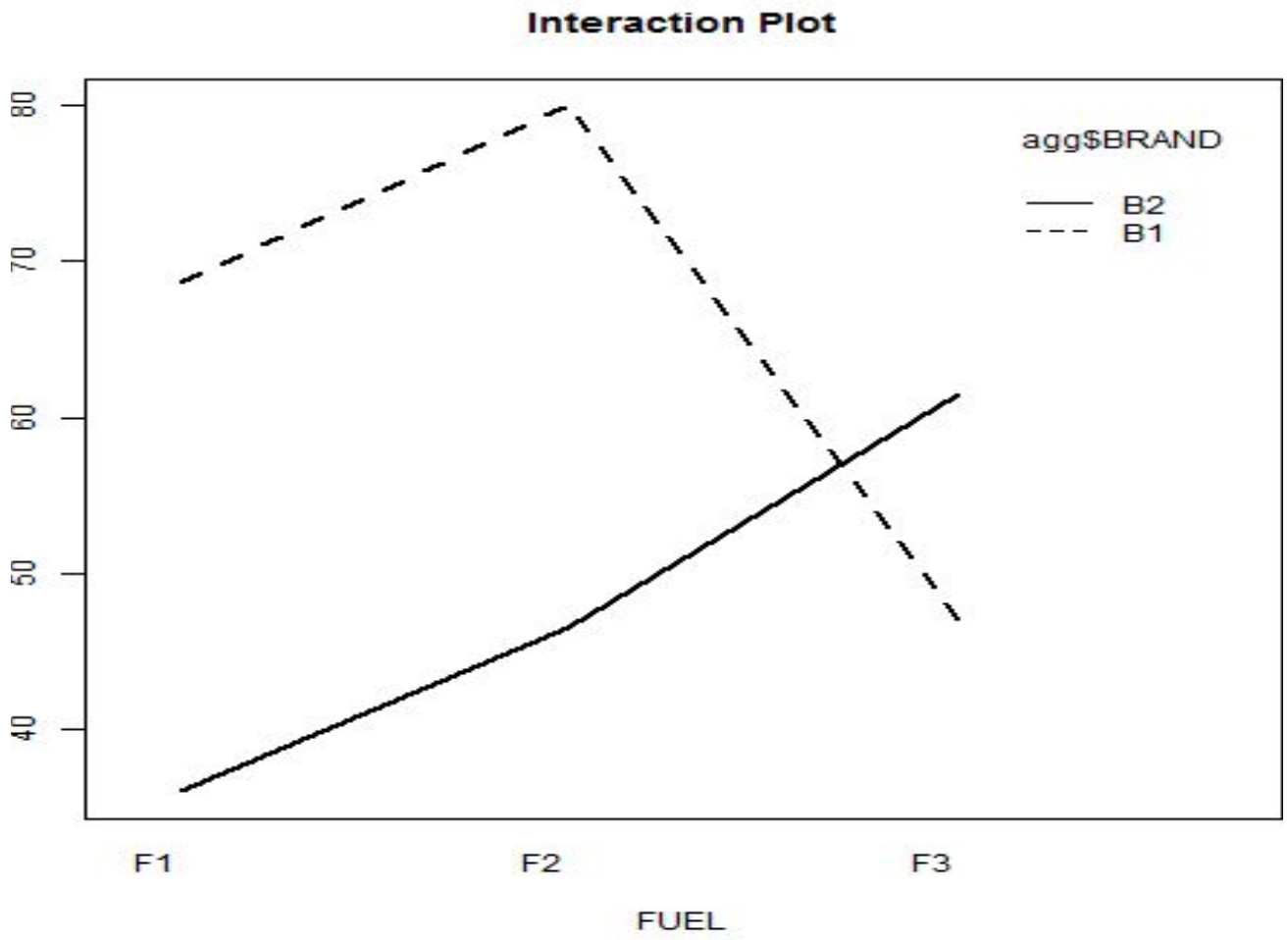
legend(x=5000, y=170000, legend=c("Hotel/Restaurant", "Retail"), col=c(1,2), pch=16)



Interaction Plot

agg=aggregate(PERFORM~FUEL+BRAND, DIESEL, mean)

interaction,plot(agg$FUEL, agg$Brand, agg$PERFORM, main=”Interaction Plot”, xlab=”FUEL”, ylab=”Mean of PERFORM”, lwd=2)

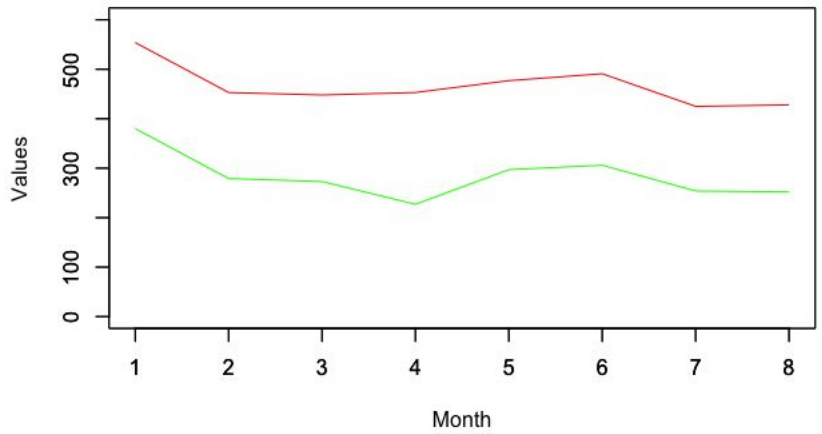


Normal Probability Plot

```
qqnorm(myd$gpa)
qqline(myd$gpa, col="red")
```

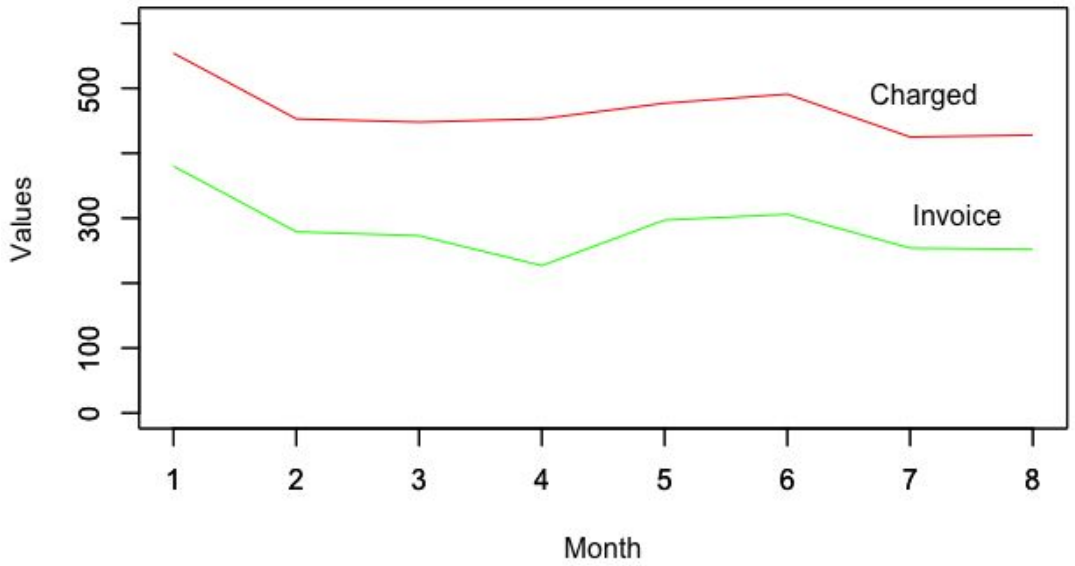
Plot Two Lines

```
plot(relativity$Month, relativity$Total.Users,
type="l", col="red", ylab =
"",ylim=c(0,600),xlab="")

par(new=TRUE)

plot(relativity$Month,
relativity$Total.Relativity, type="l",
col="green",ylim=c(0,600),xlab =
"Month",ylab="Values")
```



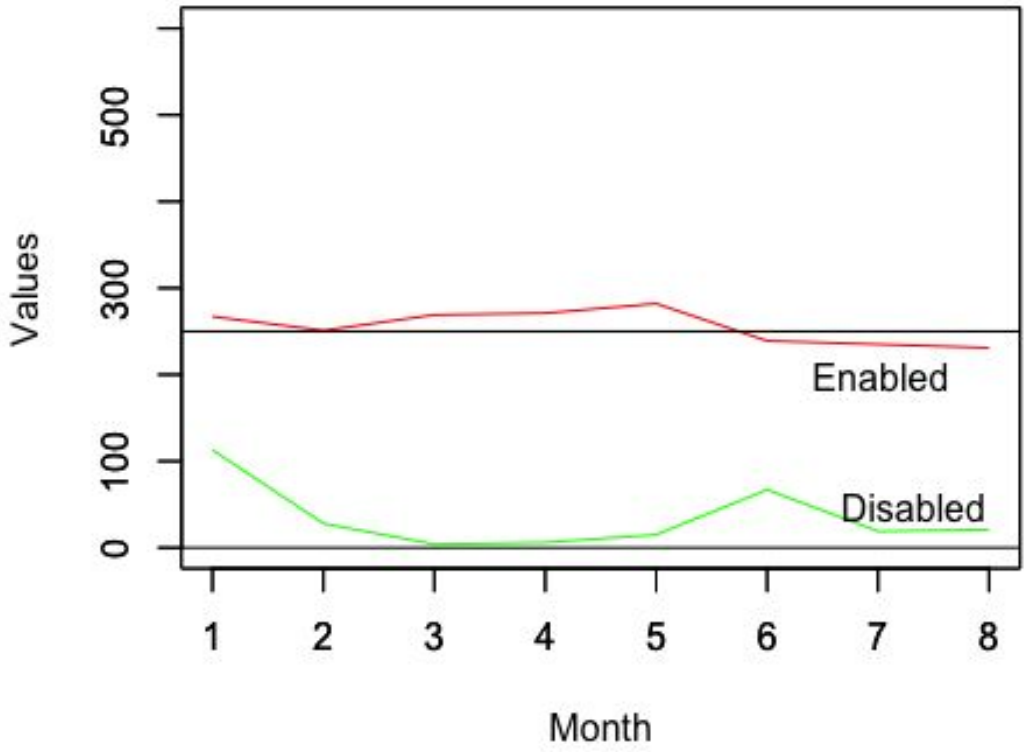
Text Locator (Point and Click)

```
plot(relativity$Month, relativity$Total.Users, type="l", col="red", ylab = "",ylim=c(0,600),xlab="")

par(new=TRUE)

plot(relativity$Month, relativity$Total.Relativity, type="l", col="green",ylim=c(0,600),xlab = "Month",ylab="Values")

text(locator(), labels=c("Charged","Invoice"))
```



Add Trendline and Locate

```
plot(relativity$Month,
relativity$Relativity.Enabled, type="l",
col="red", ylab = "",ylim=c(0,600),xlab="")

par(new=TRUE)

plot(relativity$Month,
relativity$Relativity.Disabled., type="l",
col="green",ylim=c(0,600),xlab =
"Month",ylab="Values")

text(locator(),
labels=c("Enabled","Disabled"))

abline(v=0, h=0)

abline(v=250, h=250)
```



QPLOT

```
#qplot

library(dplyr)

library(mosaic)

library(lubridate)

library(ggplot2)


#First six rows of dataset

head(Births78)


#Format some of data

##ymd=year-month-date

Births=mutate(Births78, date=ymd(date), wd=wday(date),
wday=wday(date, label=TRUE, abbr=TRUE))

head(Births, 2)


#First six rows of HELPrct

head(HELPrct)
```

Qplot Diagram

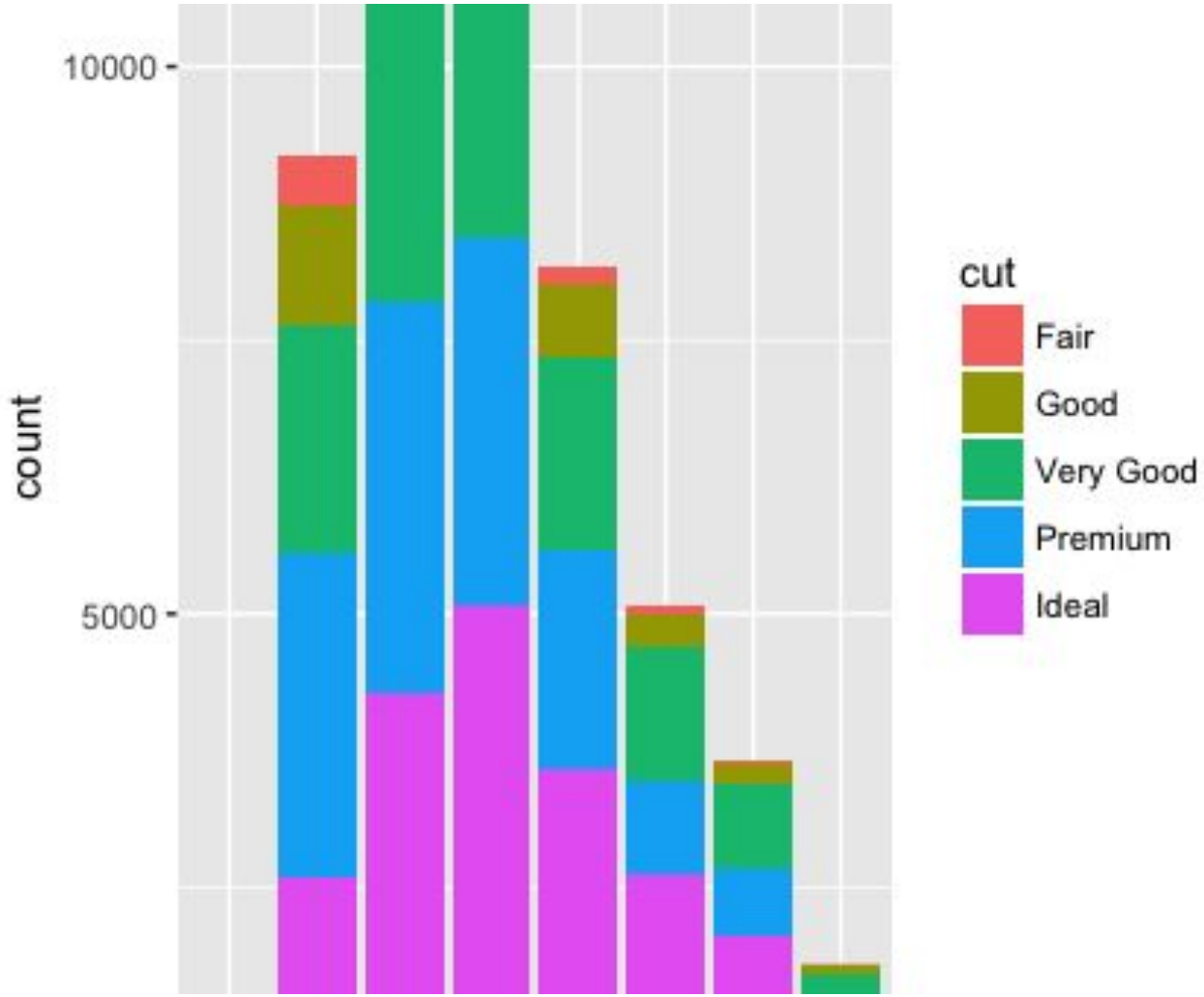
##Load library

library(ggplot2)

##Comparison qplot vs ggplot

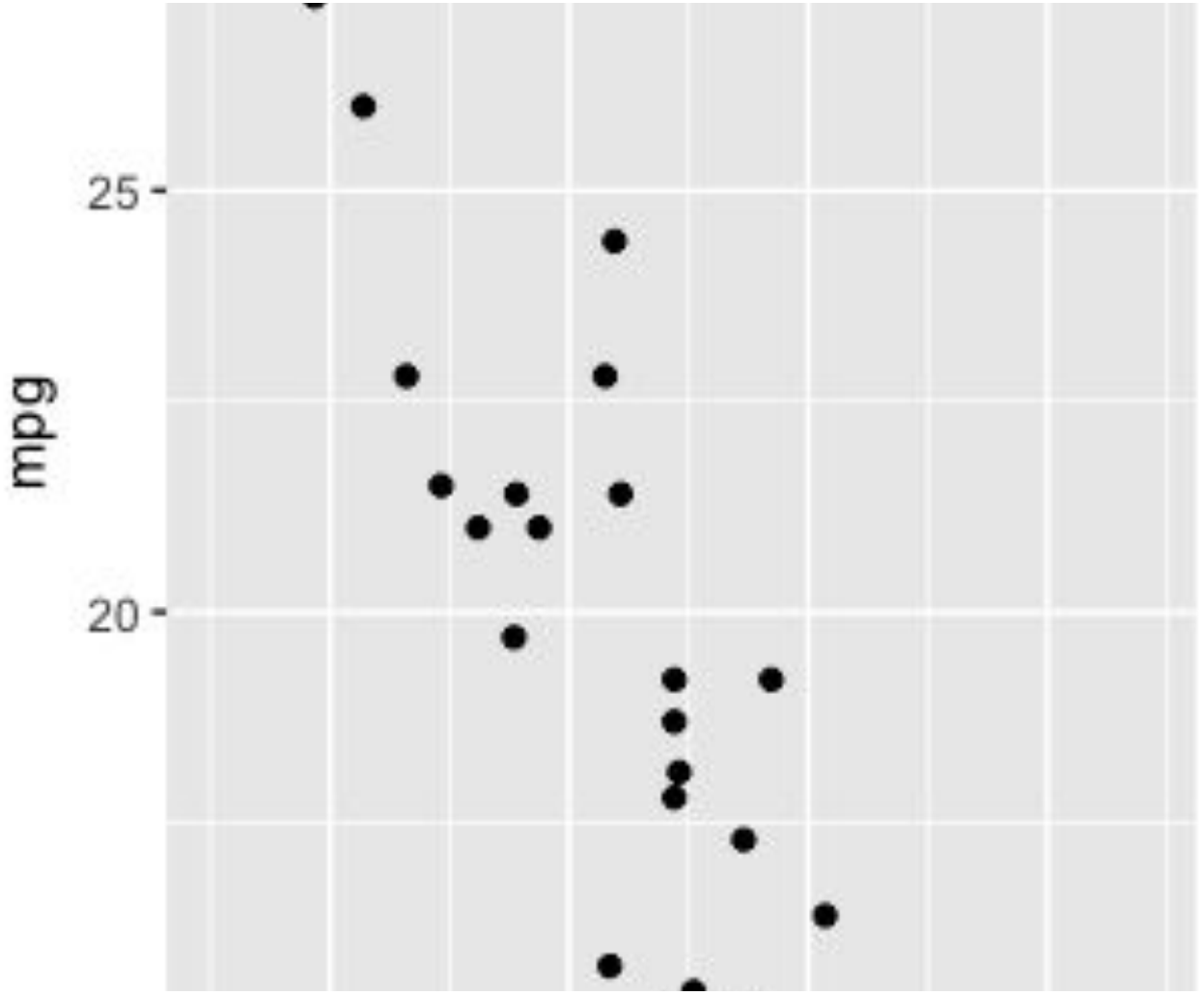
#qplot histogram

qplot(clarity, data=diamonds, fill=cut, geom="bar")



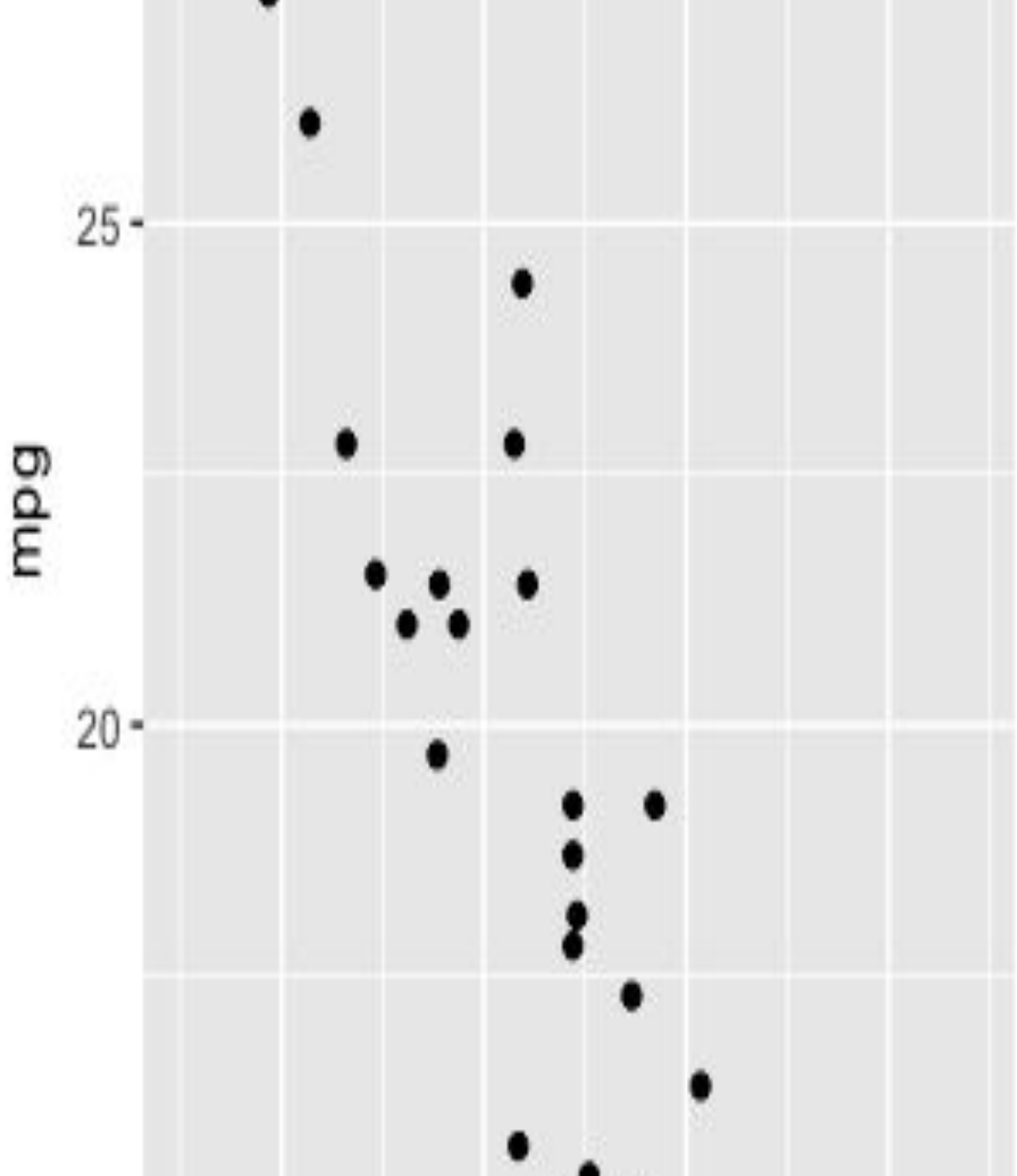
Scatterplot

qplot(wt, mpg, data=mtcars)



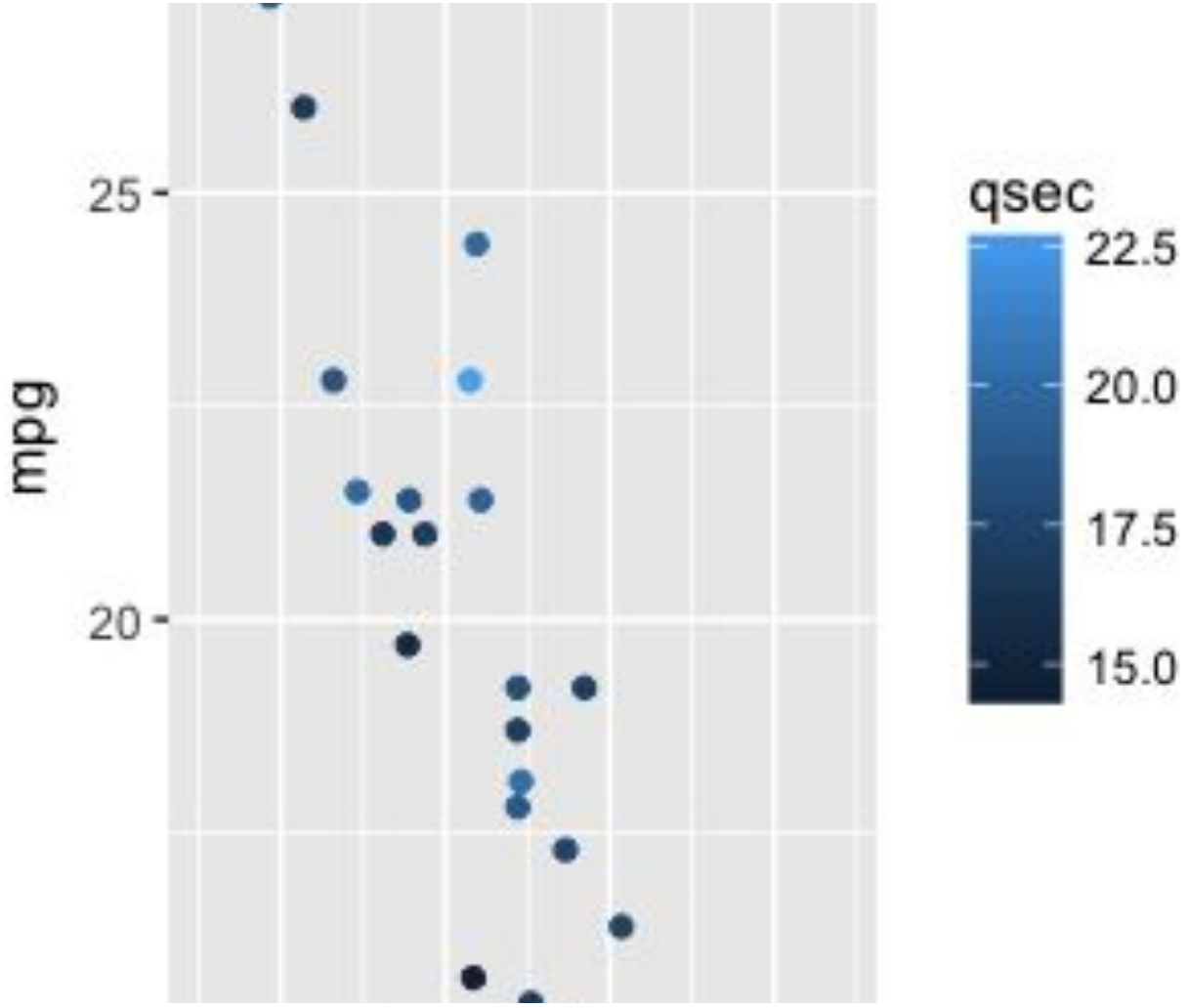
Transform input data with functions

qplot(log(wt), mpg-10, data=mtcars)



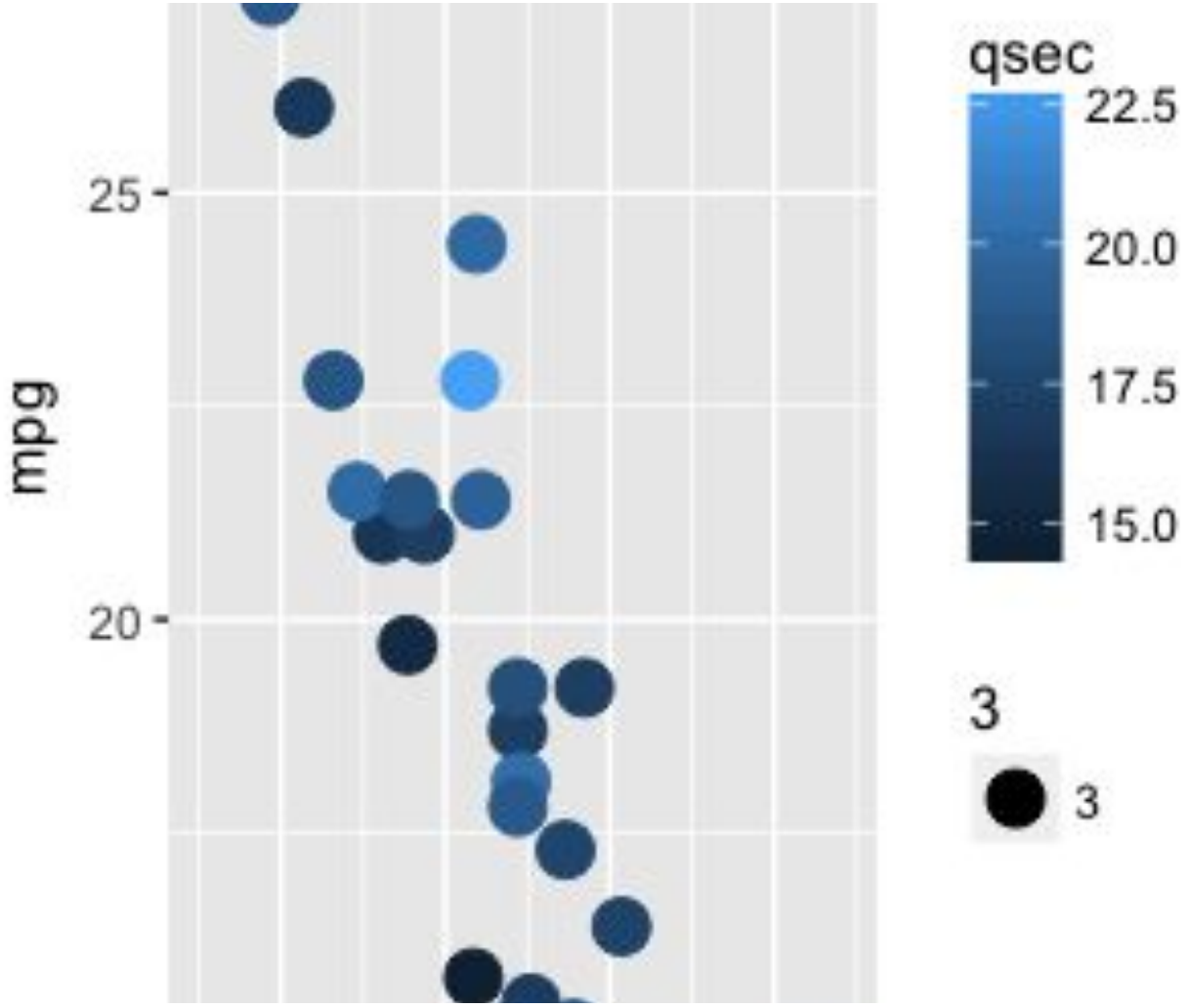
Add aesthetic mapping

qplot(wt, mpg, data=mtcars, color=qsec)



Change size of points

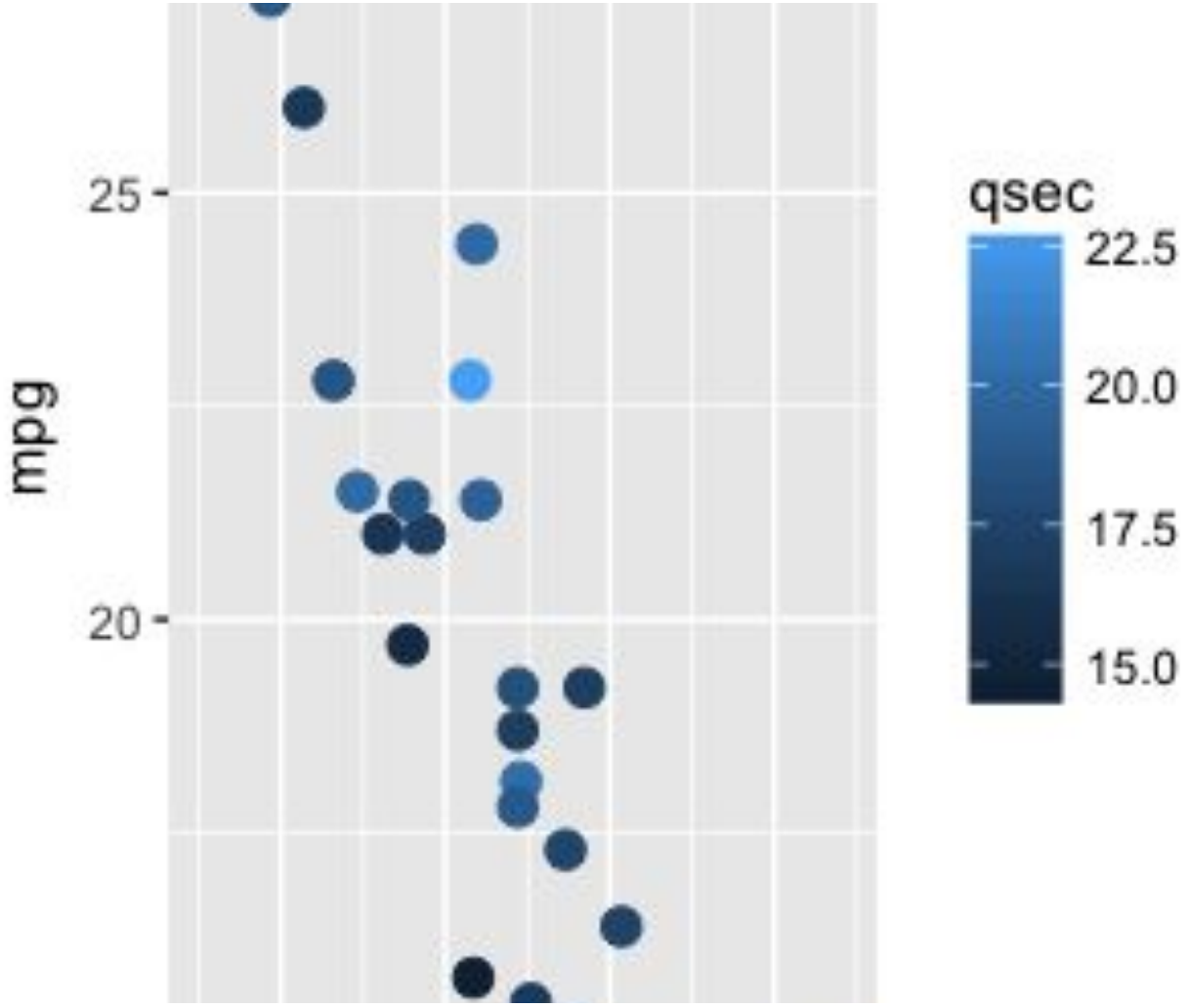
qplot(wt, mpg, data=mtcars, color=qsec, size=3)



Change size of points

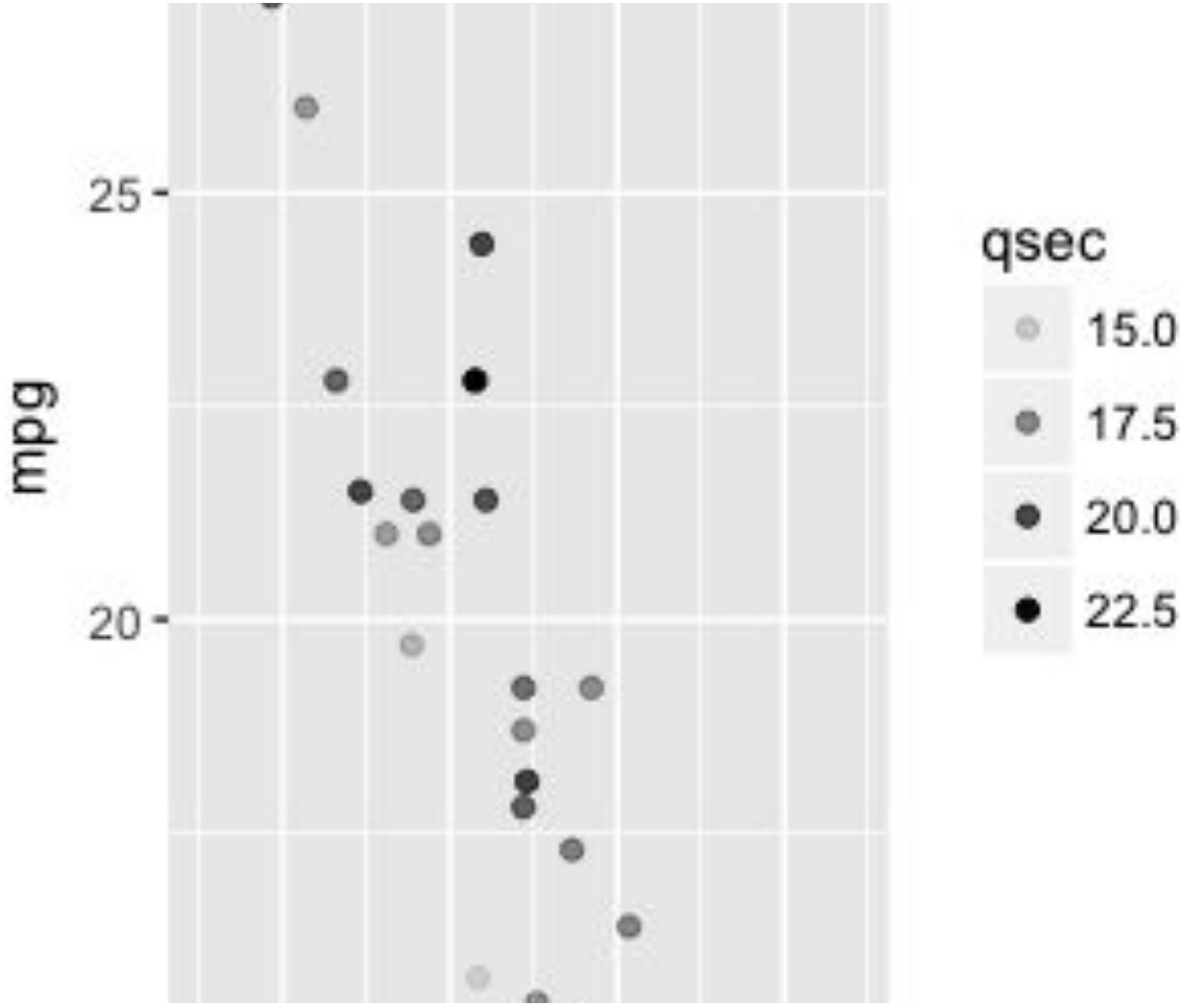
qplot(wt, mpg, data=mtcars, color=qsec, size=I(3))

I(3) can set as constant value instead of mapping



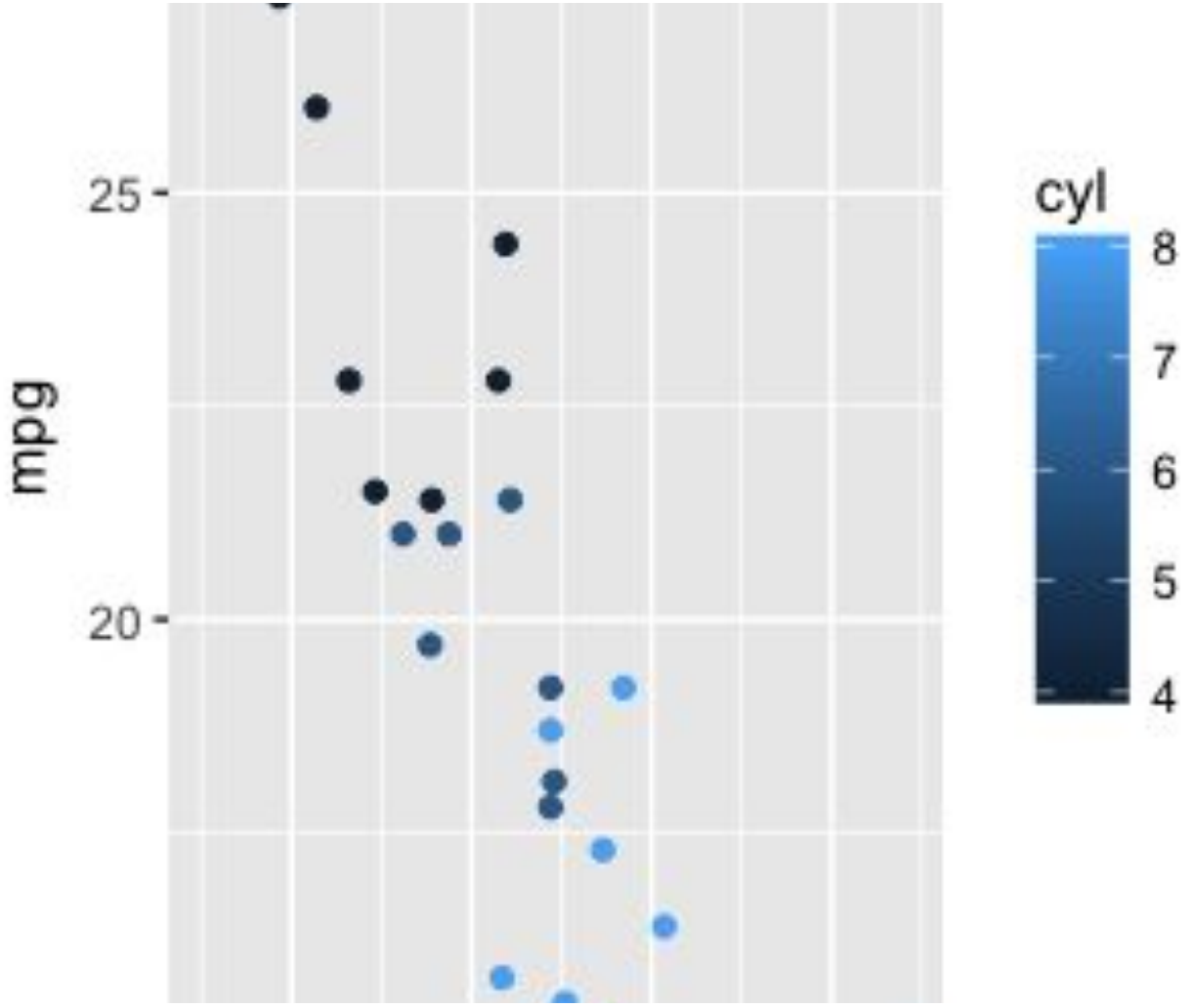
Use alpha blending

qplot(wt, mpg, data=mtcars, alpha=qsec)



Continuous scale

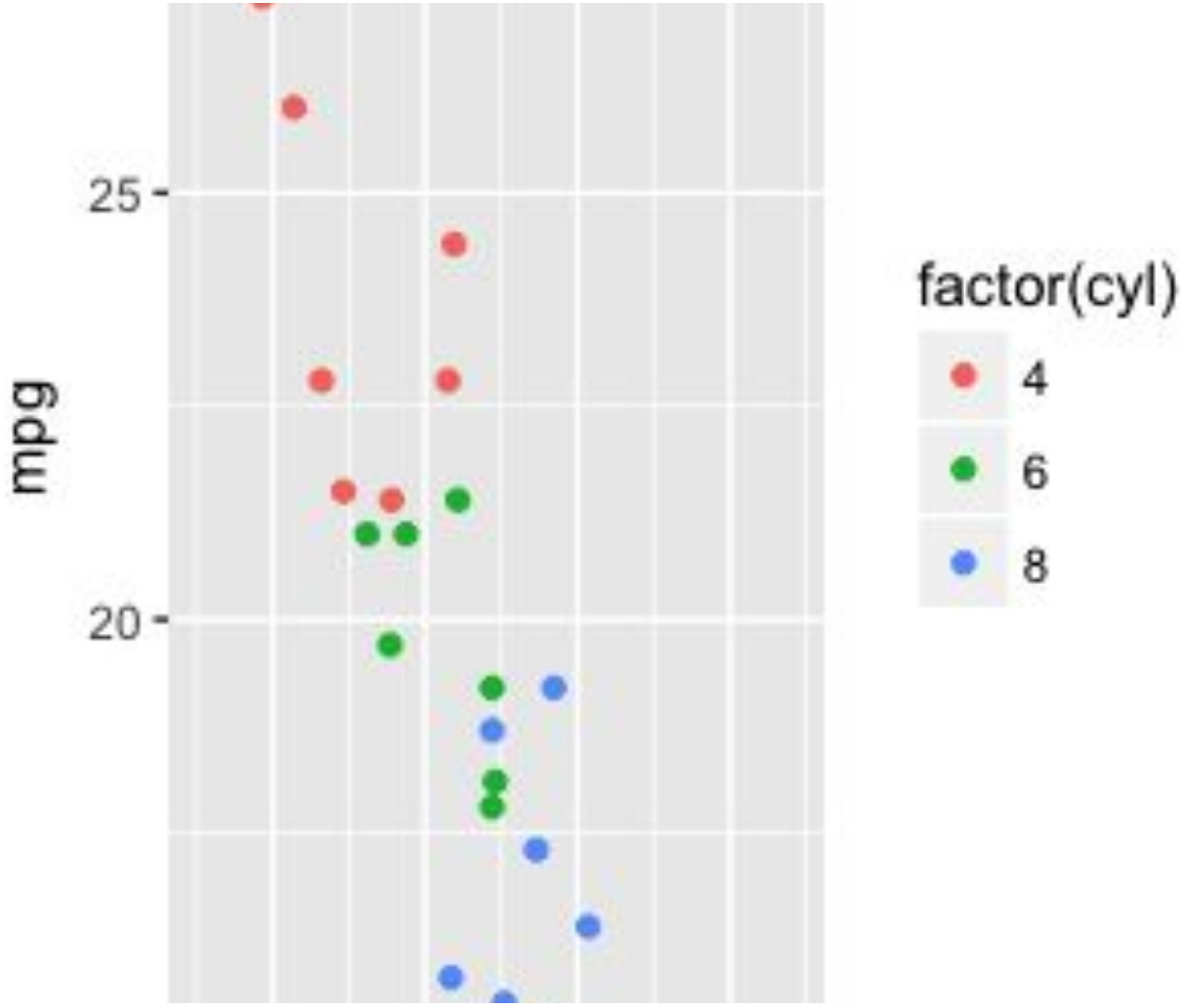
qplot(wt, mpg, data=mtcars, colour=cyl)



discrete scale

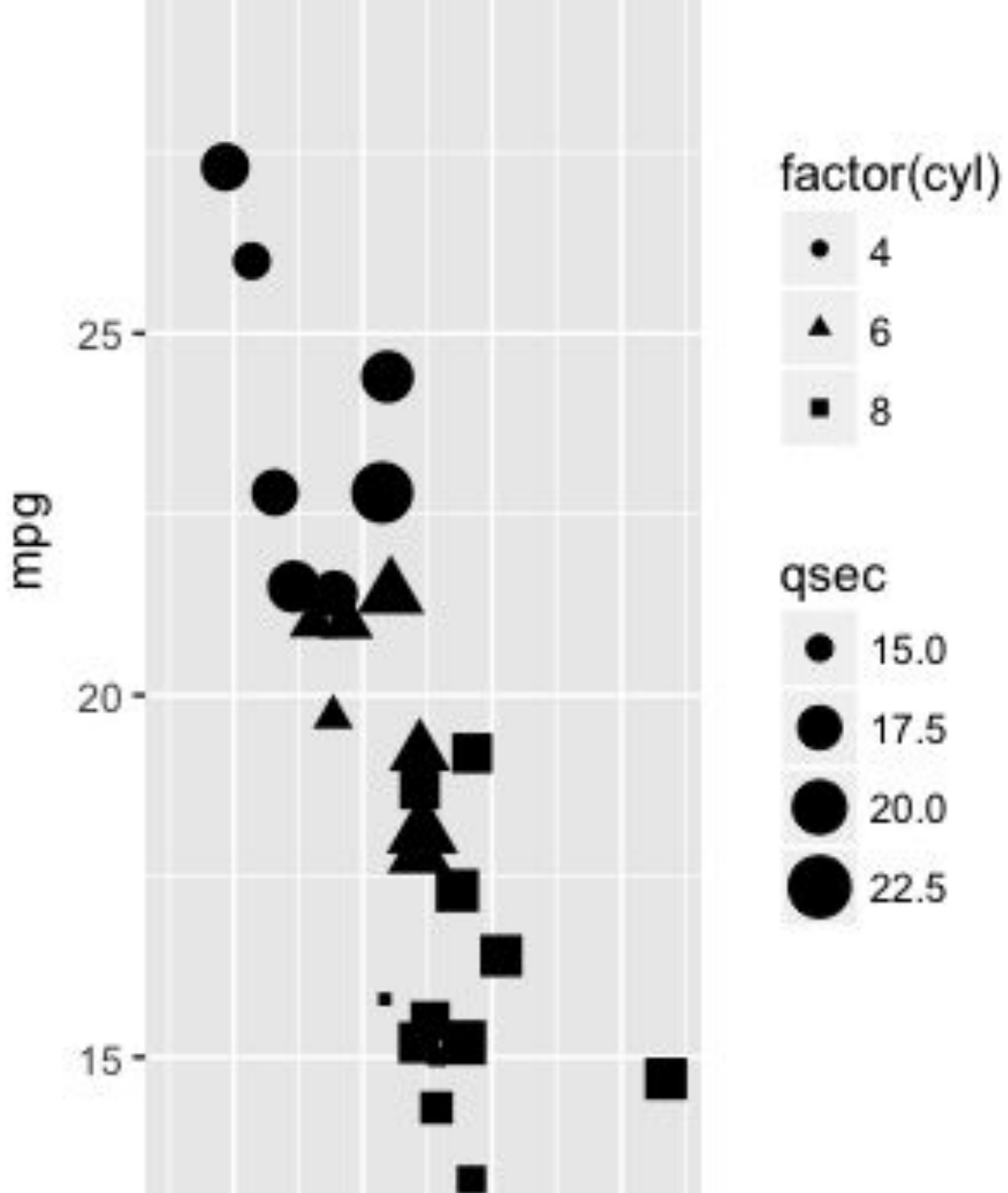
levels(mtcars$cyl)

qplot(wt, mpg, data=mtcars, colour=factor(cyl))



Use different aesthetic mappings

qplot(wt, mpg, data=mtcars,
shape=factor(cyl), size=qsec)

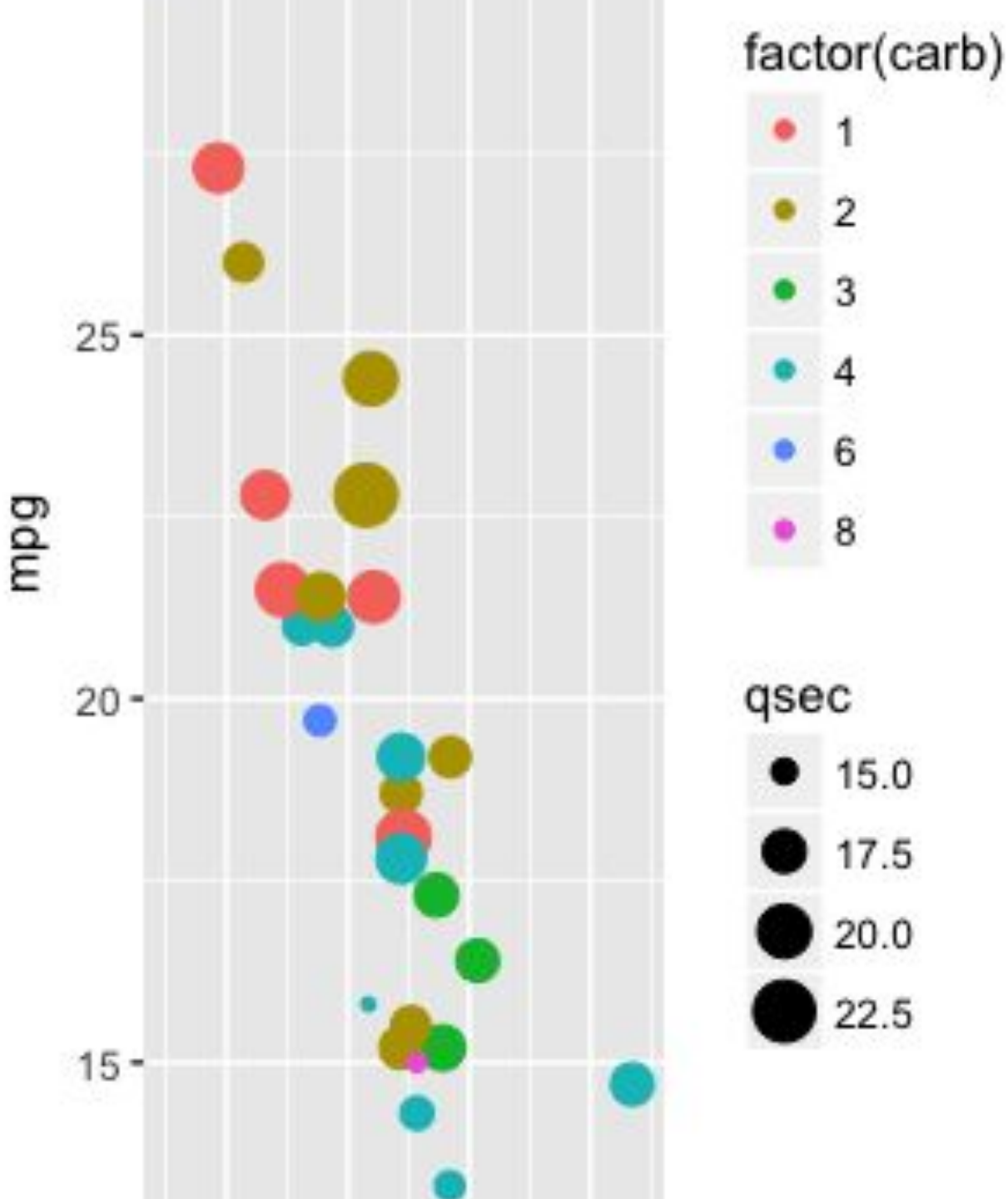


Combine mappings

qplot(wt, mpg, data=mtcars, size=qsec, color=factor(carb), geom = "point")

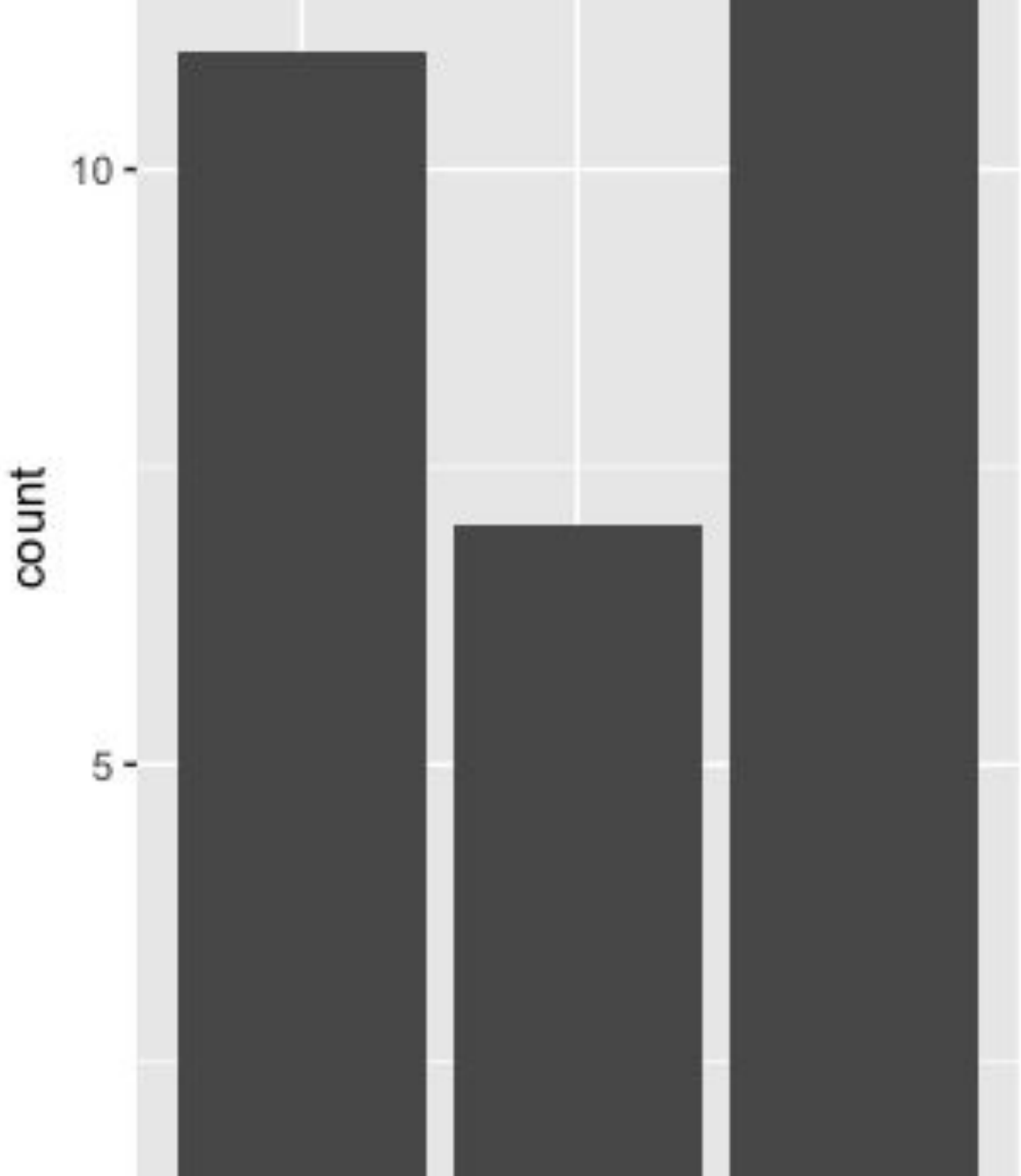
qplot(wt, mpg, data=mtcars, size=qsec, color=factor(carb))

qplot(wt, mpg, data=mtcars, size=qsec, color=factor(cyl), shape=I(1))



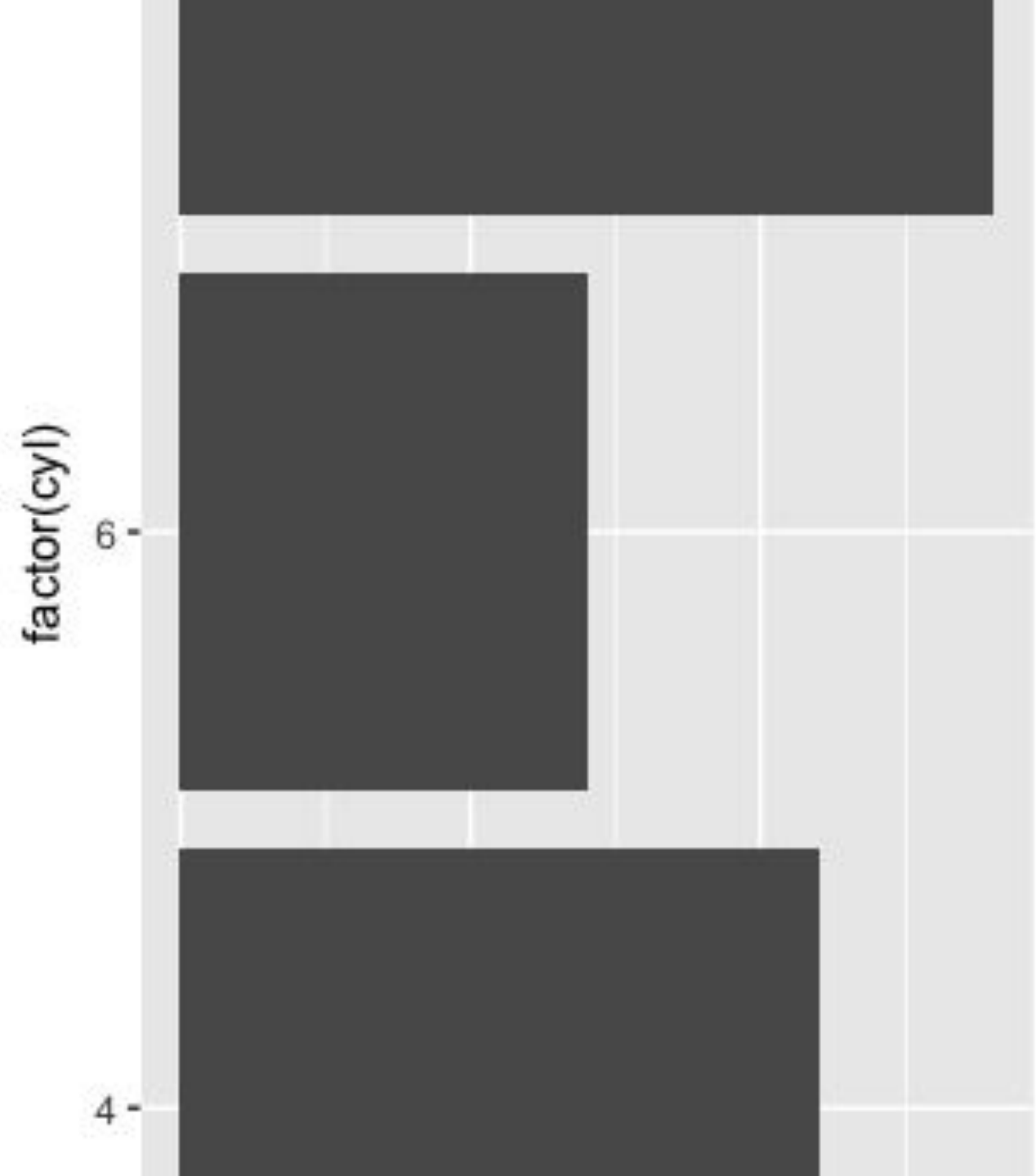
Bar plot

qplot(factor(cyl), data=mtcars, geom="bar")



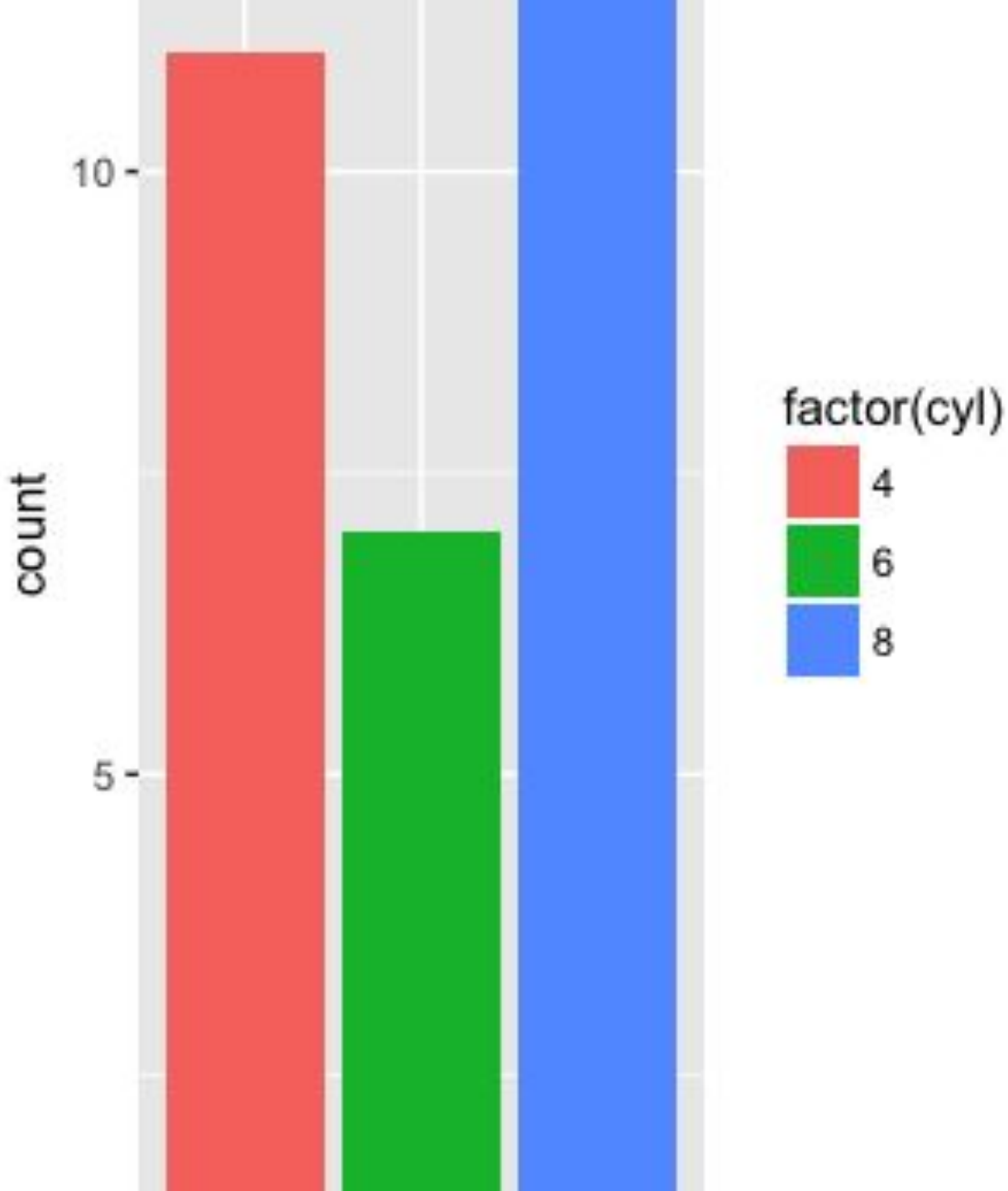
Flip plot by 90

qplot(factor(cyl), data = mtcars, geom="bar")+coord_flip()



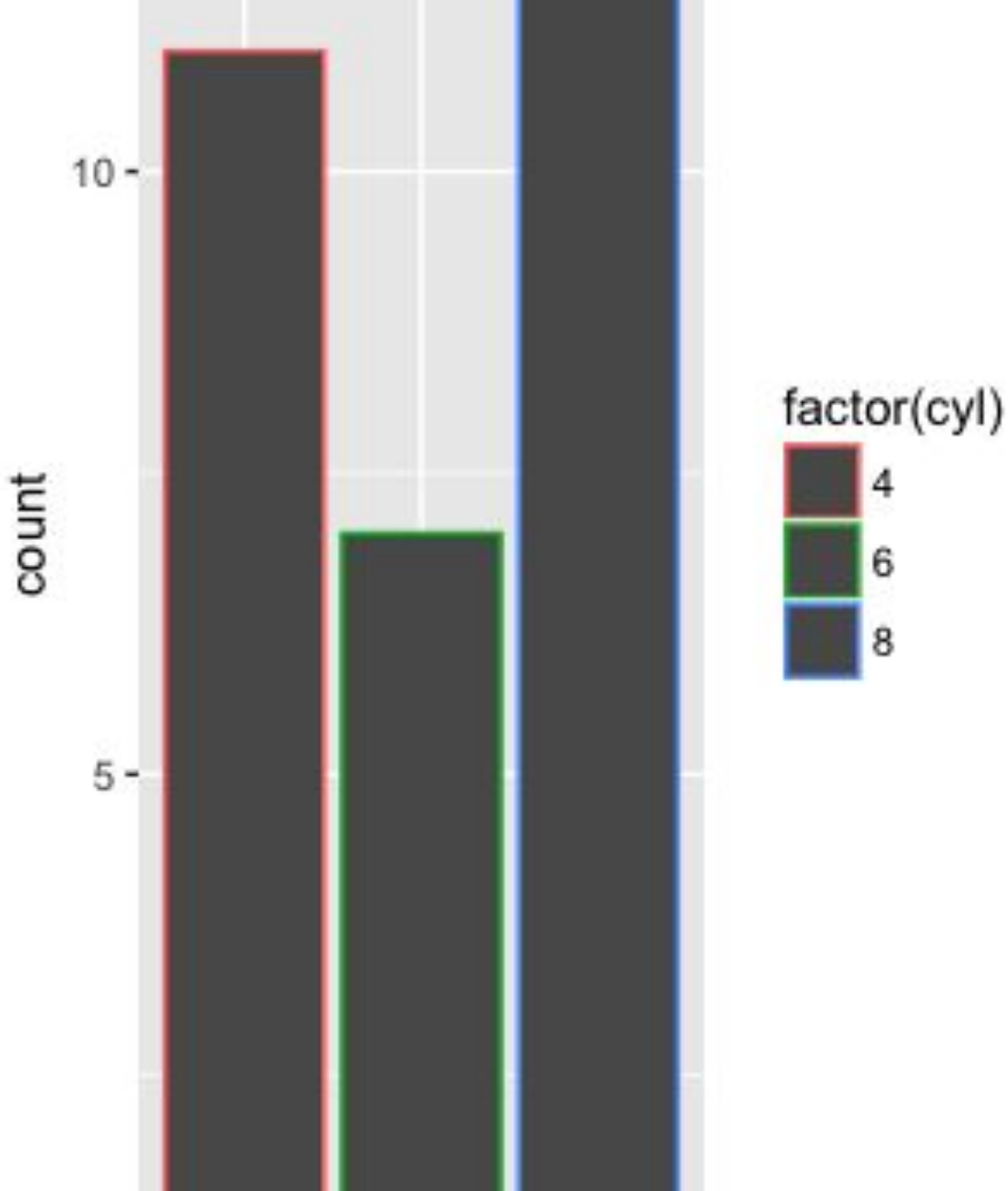
Fill Bar

qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(cyl))



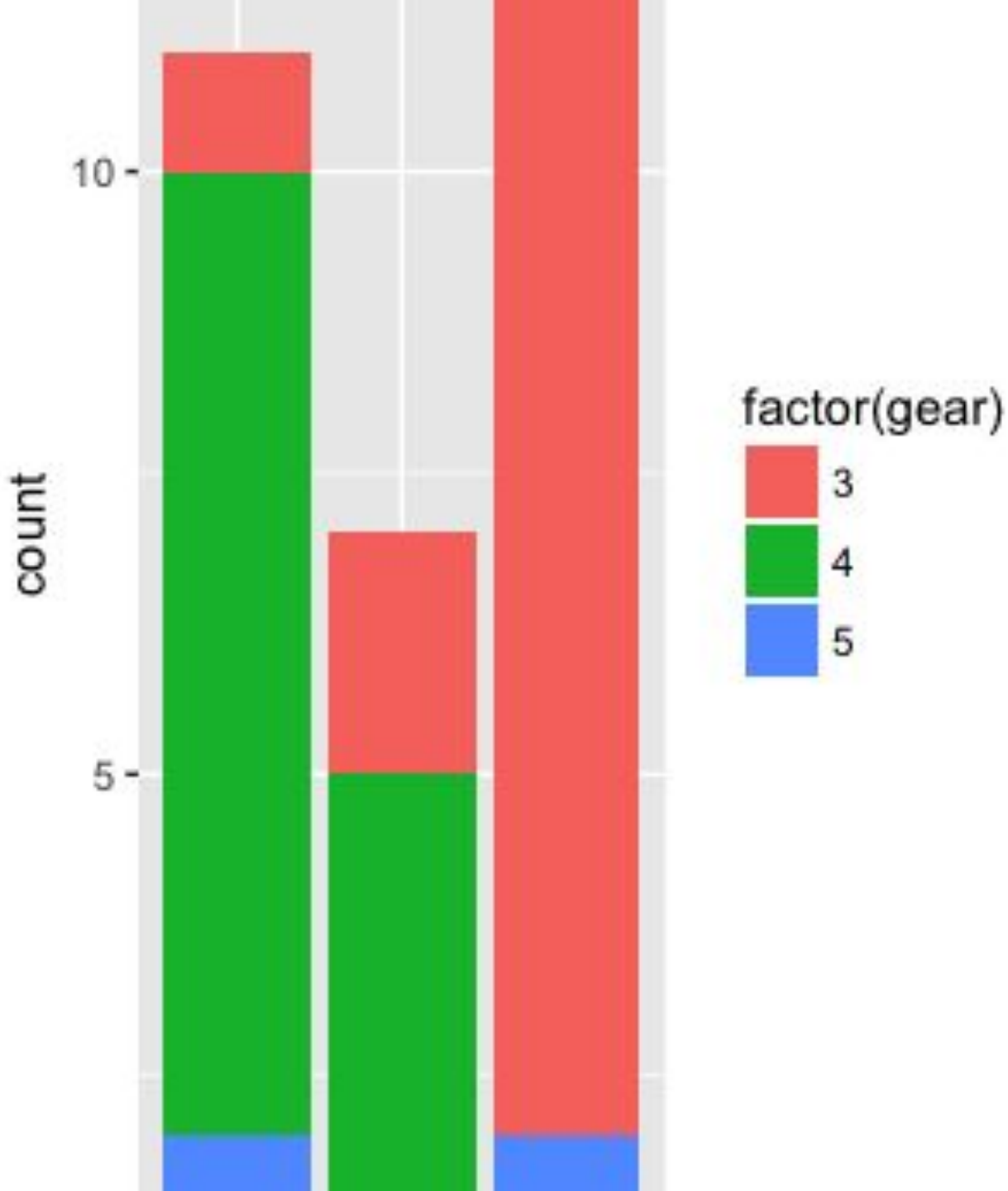
Color Bar

qplot(factor(cyl), data=mtcars, geom="bar", colour=factor(cyl))



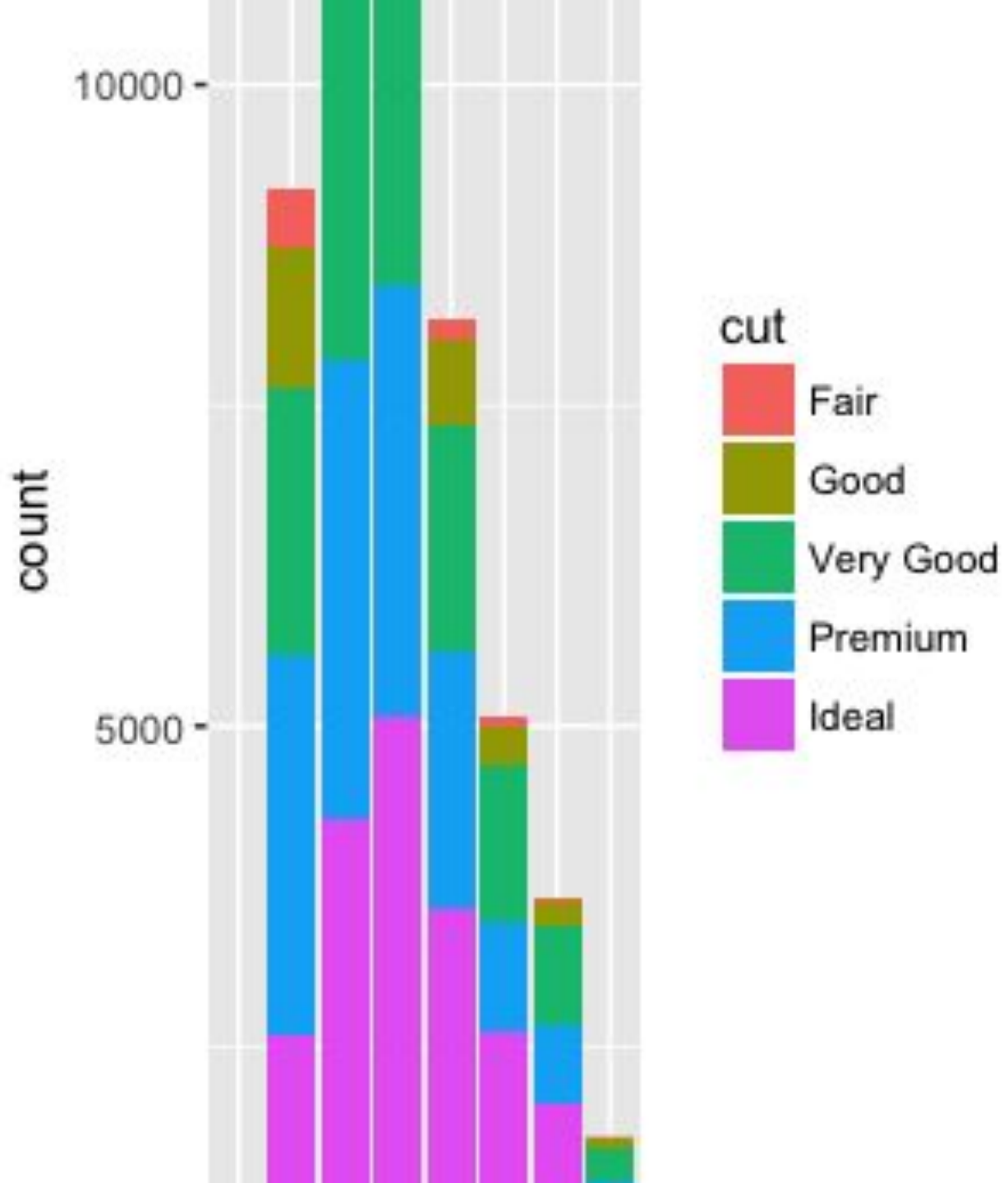
Bar fill by variable

qplot(factor(cyl), data=mtcars, geom="bar", fill=factor(gear))



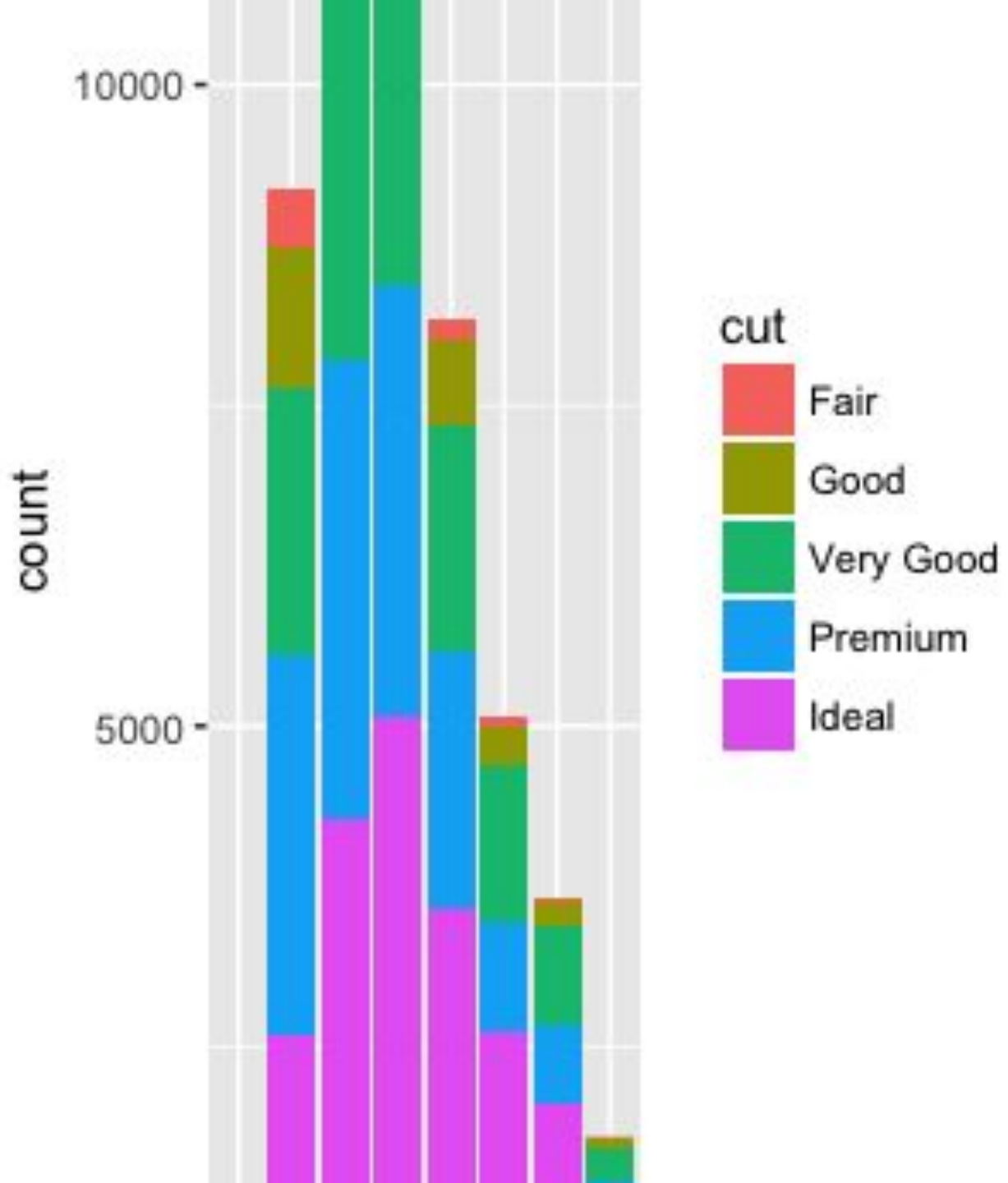
Stacked bars

qplot(clarity, data=diamonds, geom="bar", fill=cut, position="stack")



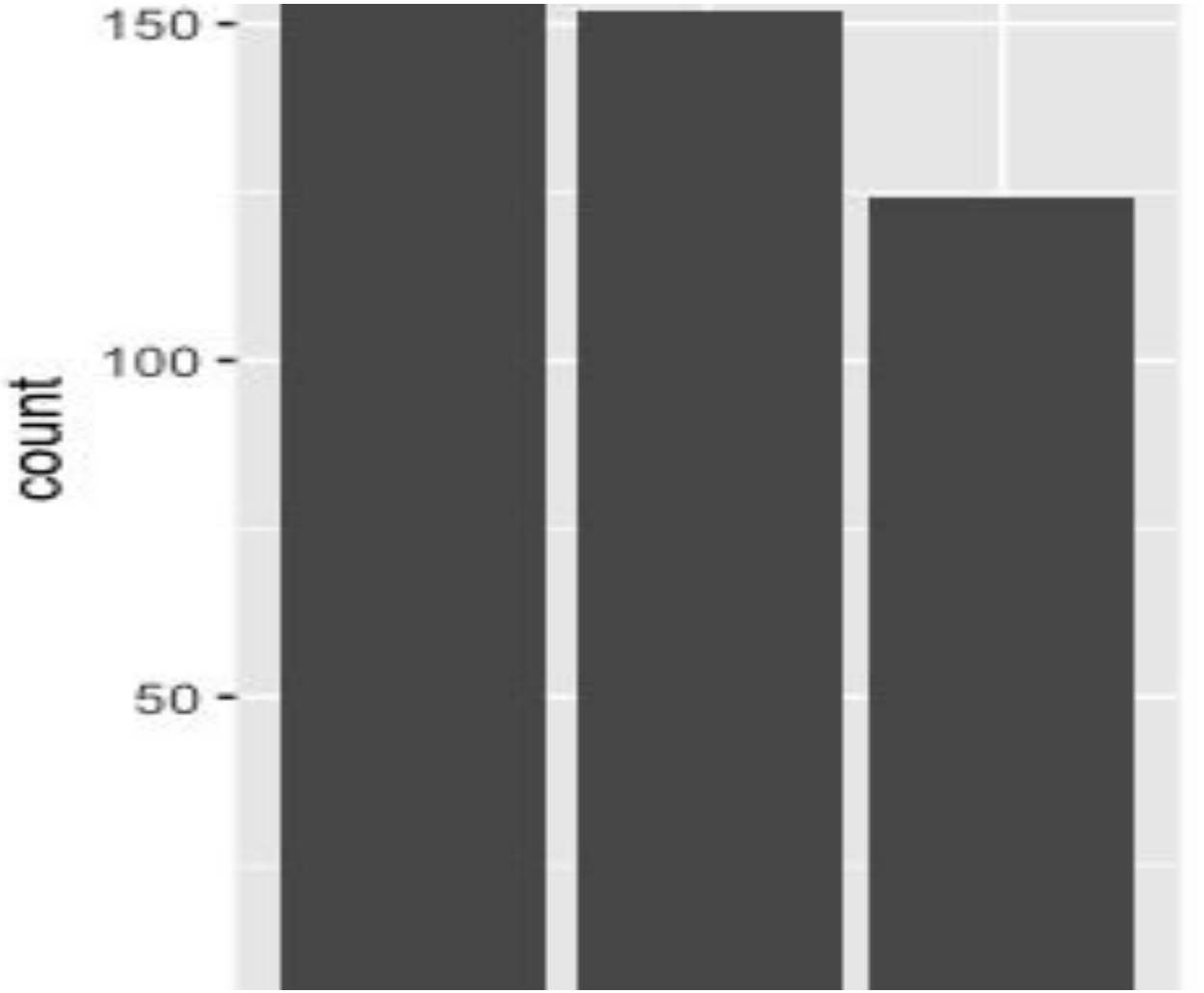
Dodged Bars

qplot(clarity, data=diamonds, geom="bar",
fill=cut, position = "dodge")



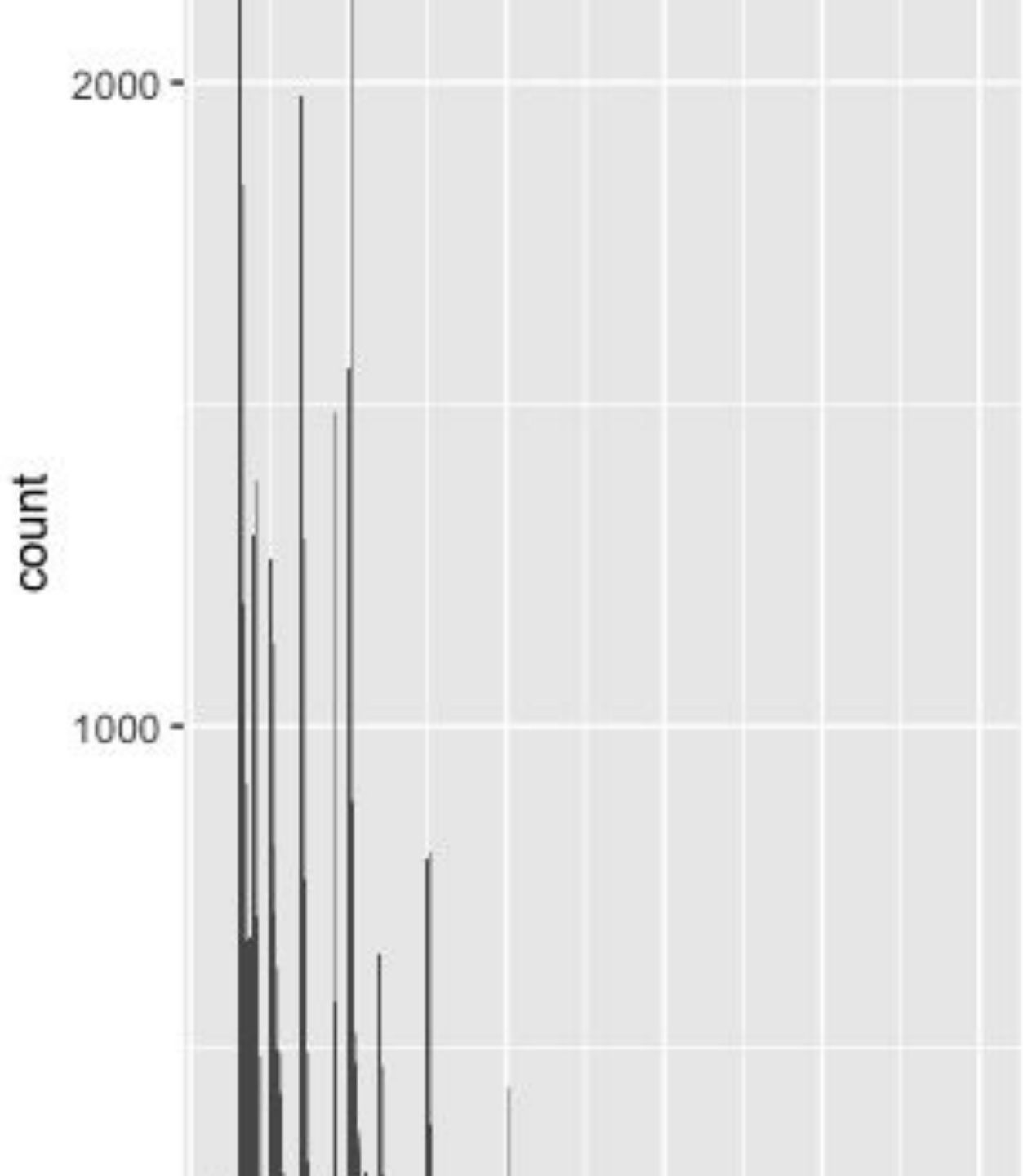
Histogram

qplot(substance, data=HELPrct)



Histogram with Change binwidth

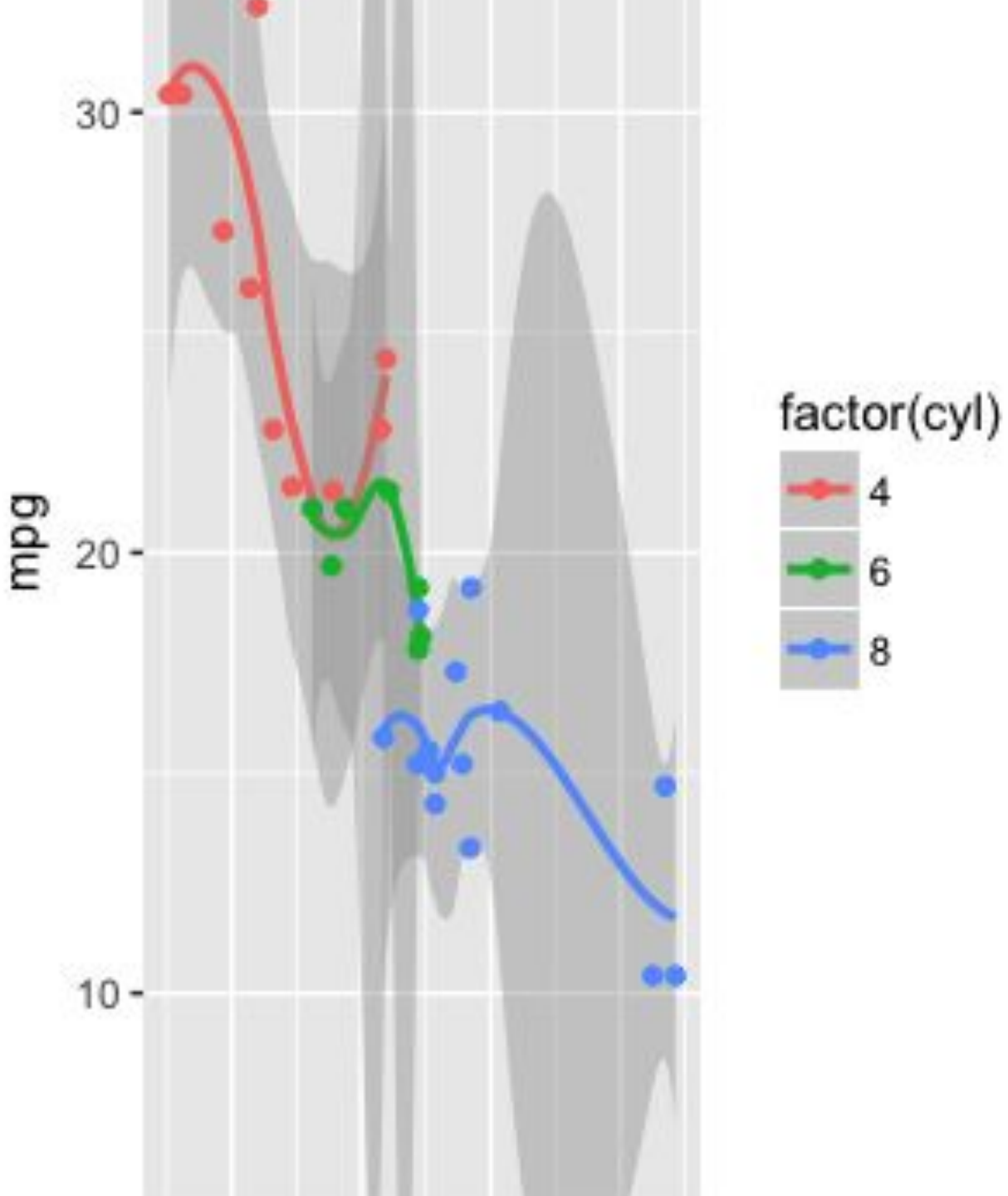
qplot(carat, data=diamonds, geom="histogram", binwidth=0.01)



Use geom to combine plots

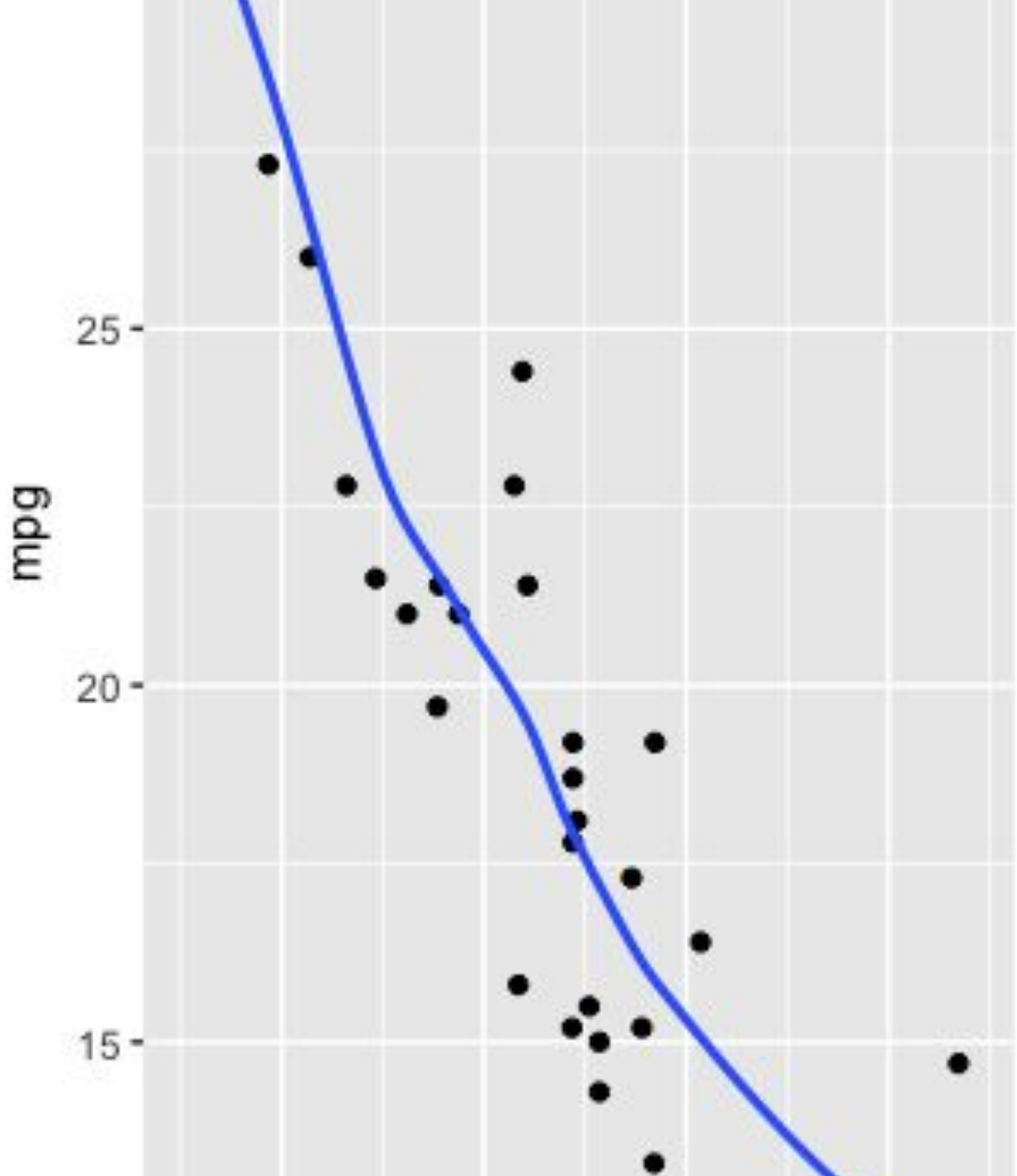
qplot(wt, mpg,
data=mtcars,color=factor(cyl),
geom=c("smooth","point"))

#Order of layer



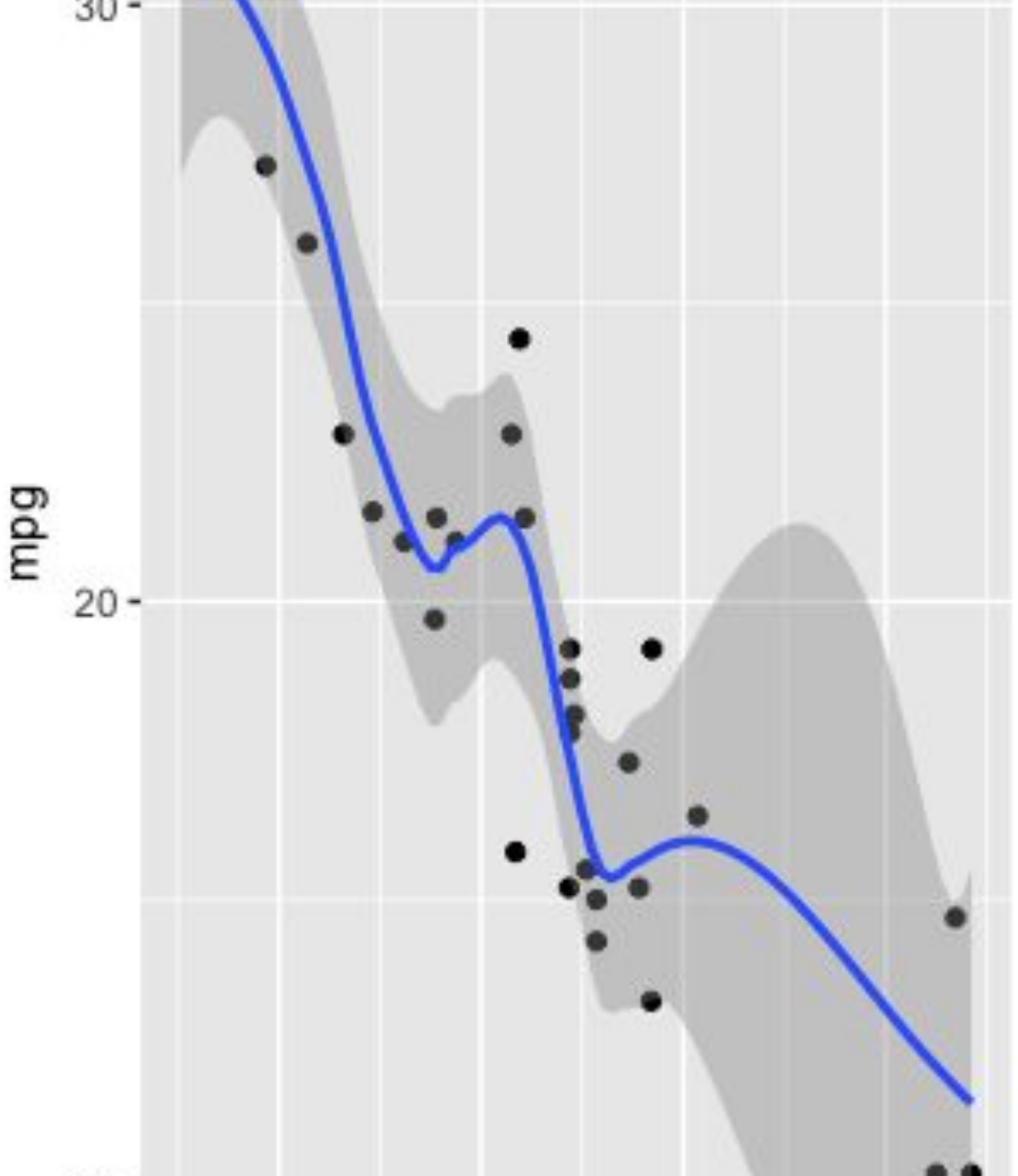
Removing standard error

qplot(wt, mpg, data=mtcars,
geom=c("point", "smooth"), se=FALSE)



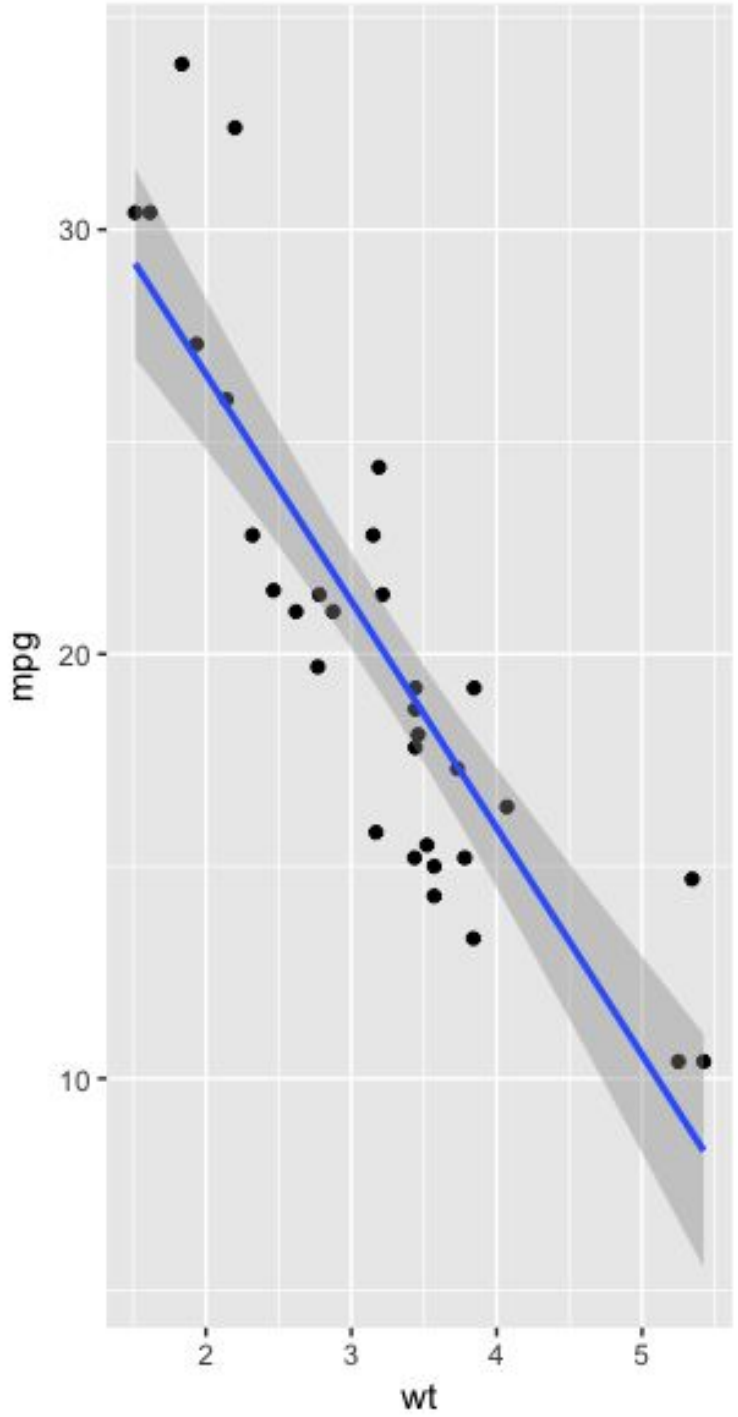
Making line more or less wiggly (span:0-1)

qplot(wt, mpg, data=mtcars,
geom=c("point", "smooth"), span=0.5)



Using linear modeling

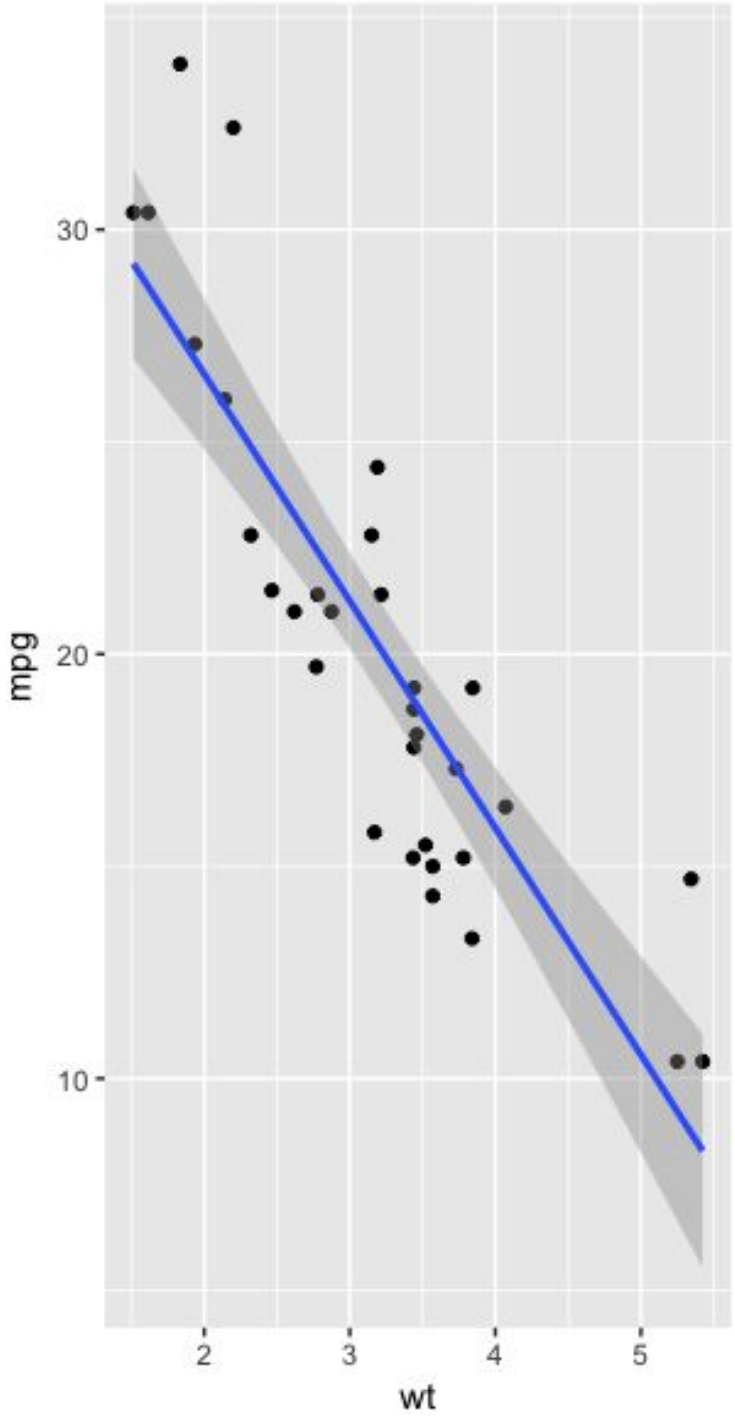
qplot(wt, mpg, data=mtcars,
geom=c("point","smooth"), method="lm")



Using a custom formula for fitting

library(splines)

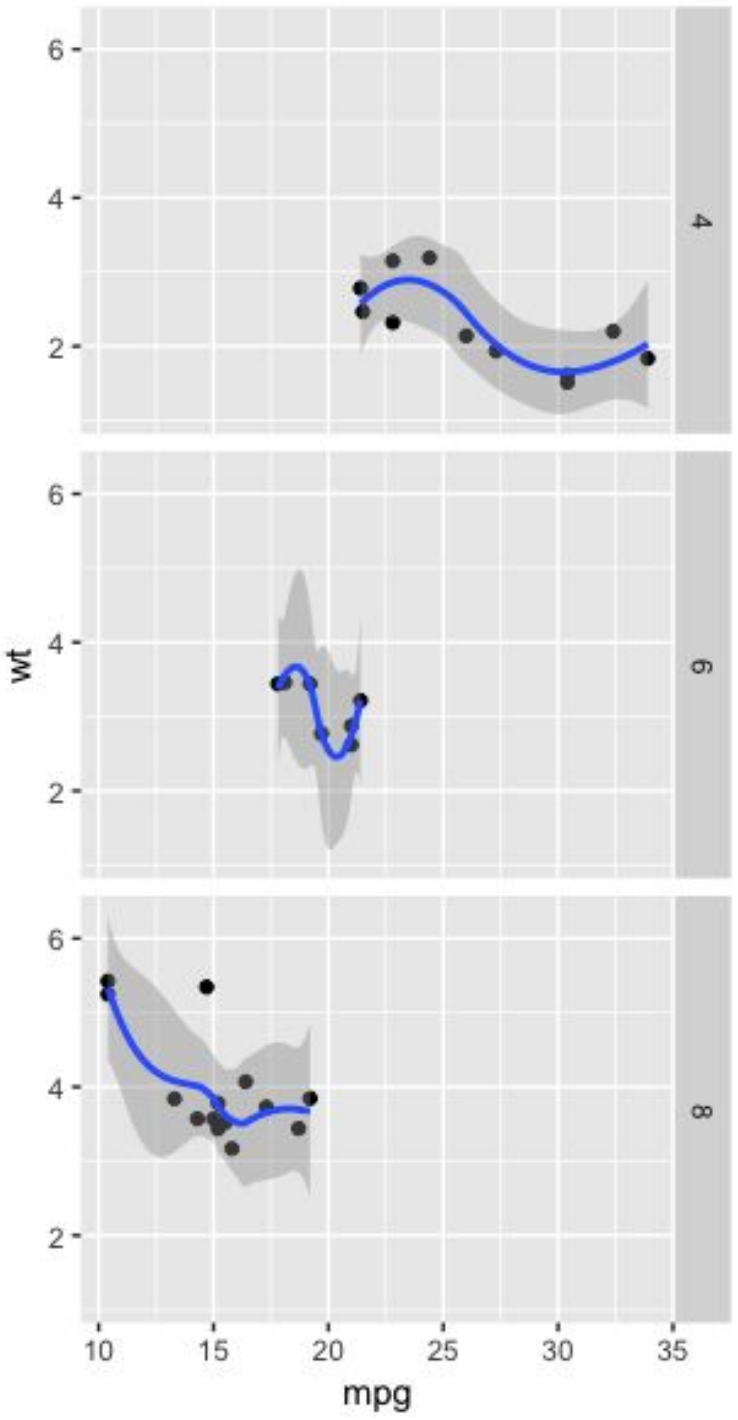
qplot(wt, mpg, data=mtcars,
geom=c("point","smooth"),
method="lm",fomular=y~ns(x,5))

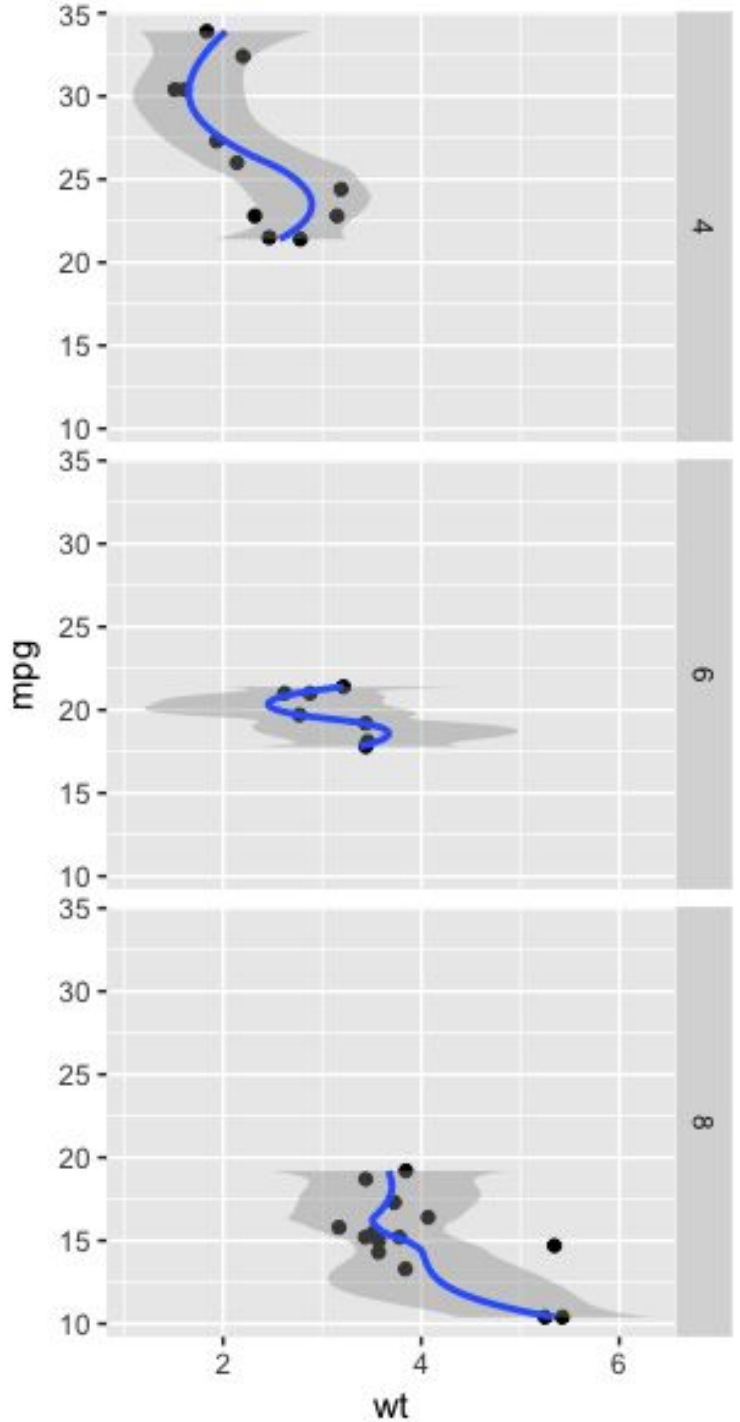


Illustrate flip versus changing of variable allocation

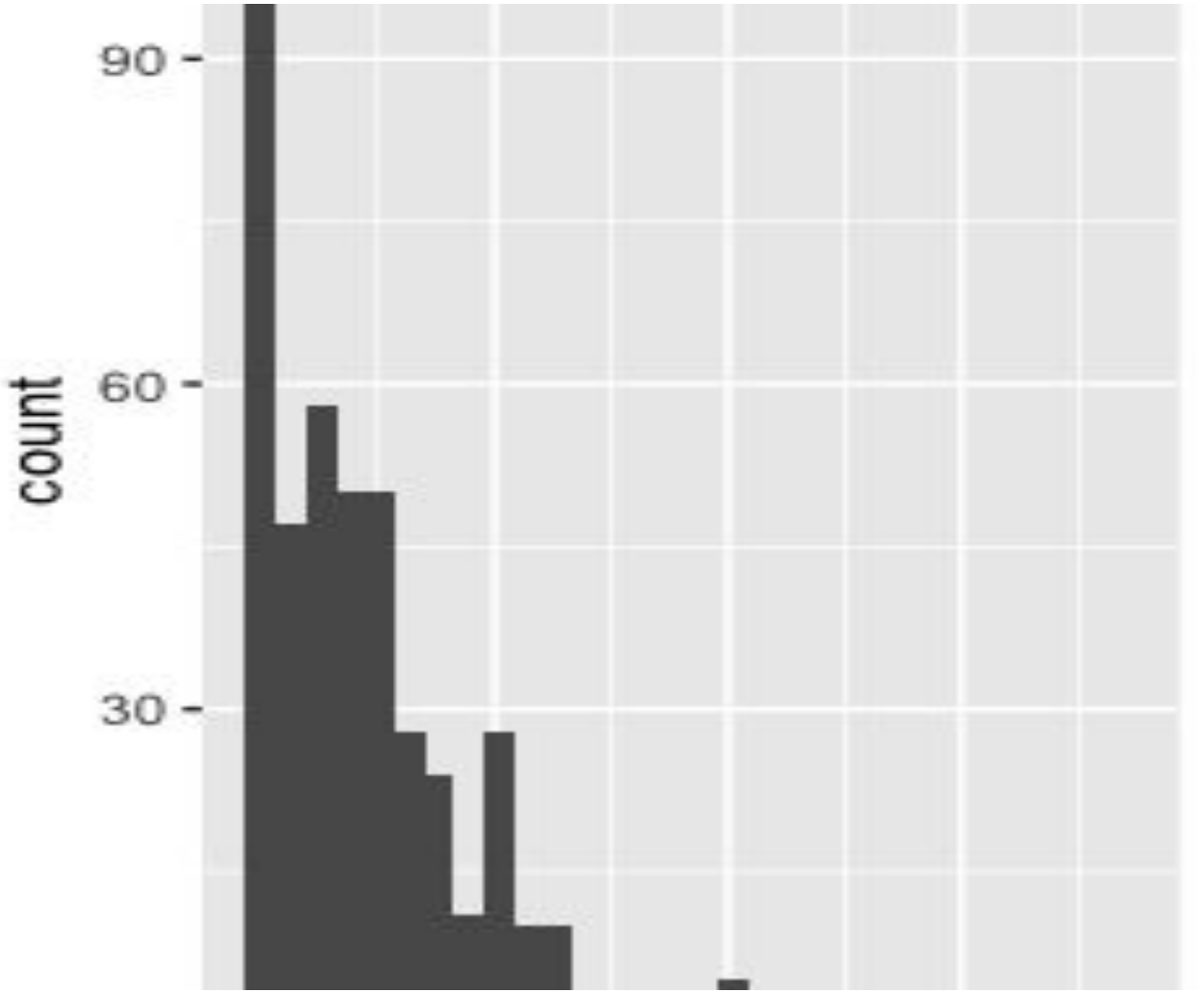
qplot(mpg, wt, data=mtcars, facets=cyl~.
geom=c("point","smooth"))

qplot(mpg, wt, data=mtcars, facets=cyl~.
geom=c("point","smooth"))+coord_flip()





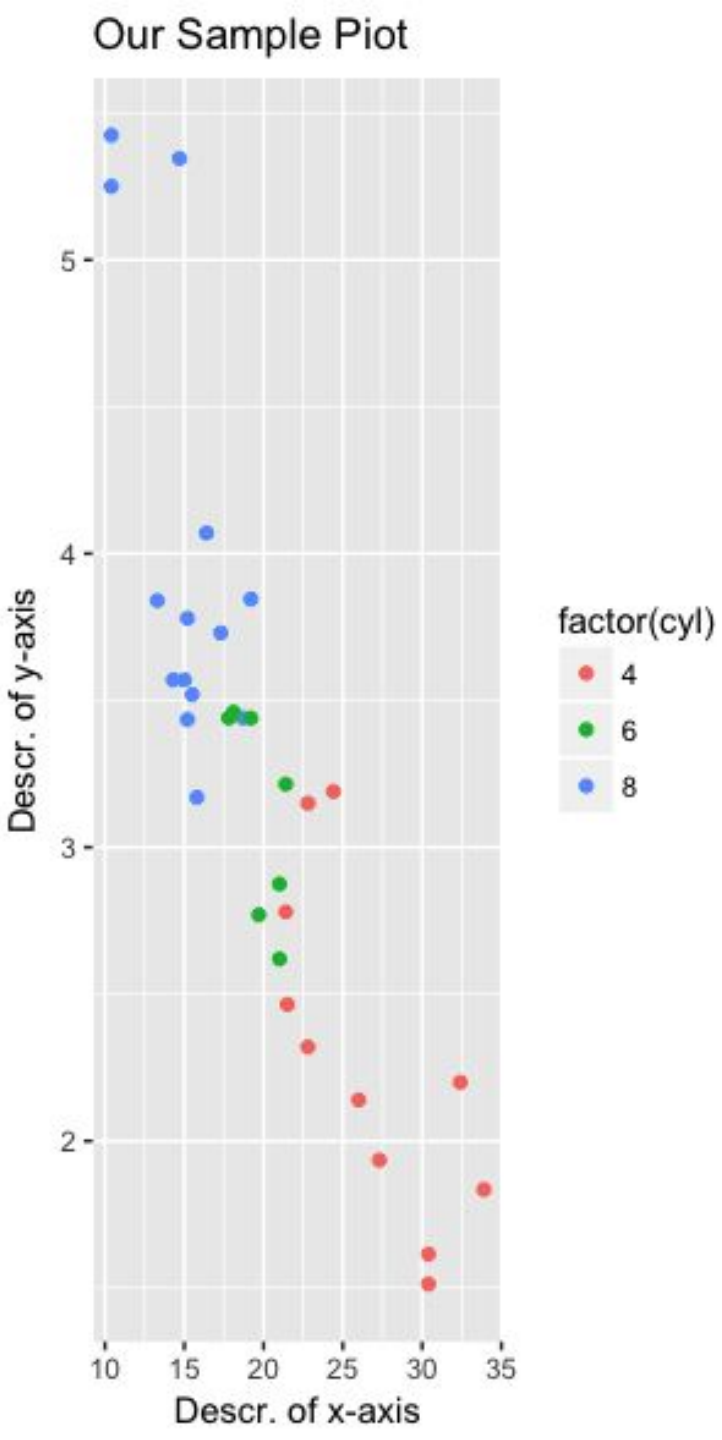
qplot(i2, data=HELPrct)



Changing text

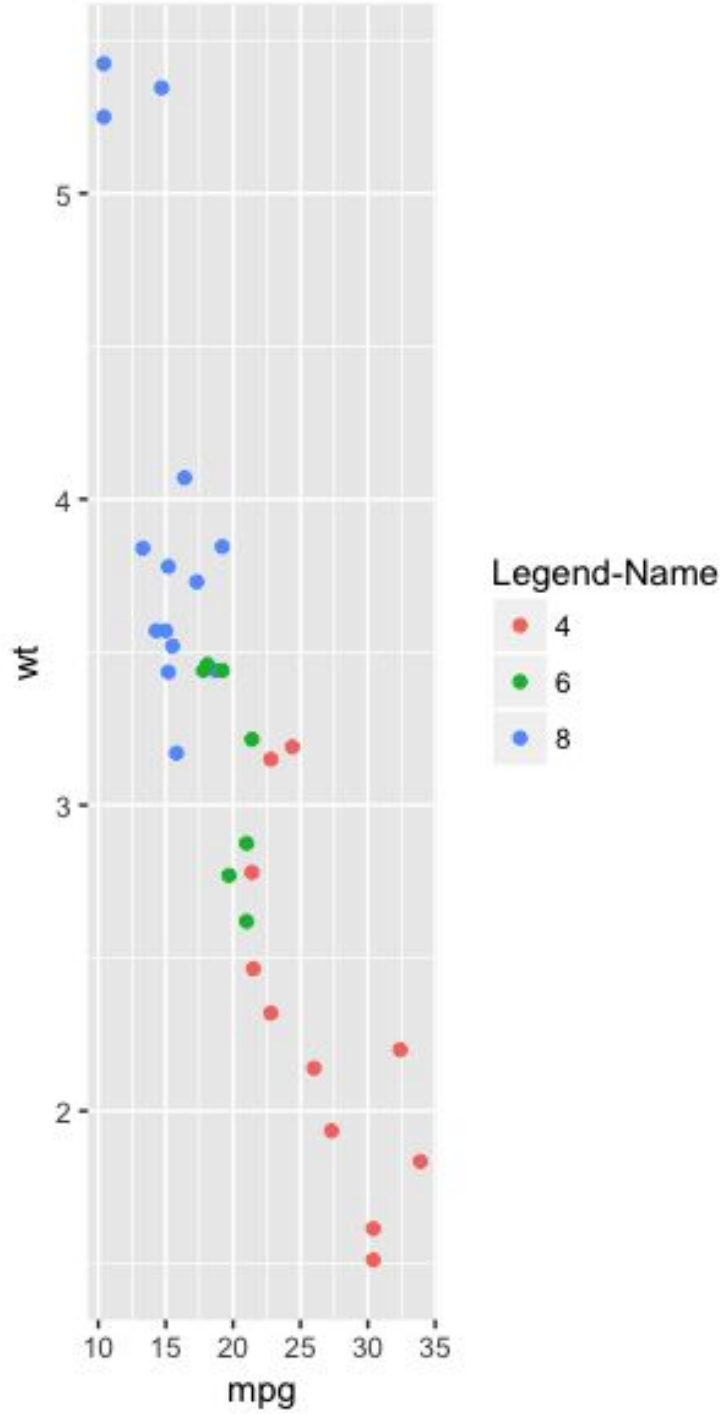
qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point", xlab = "Descr. of x-axis", ylab="Descr. of y-axis", main="Our Sample Piot")

qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point")+xlab("x-axis")



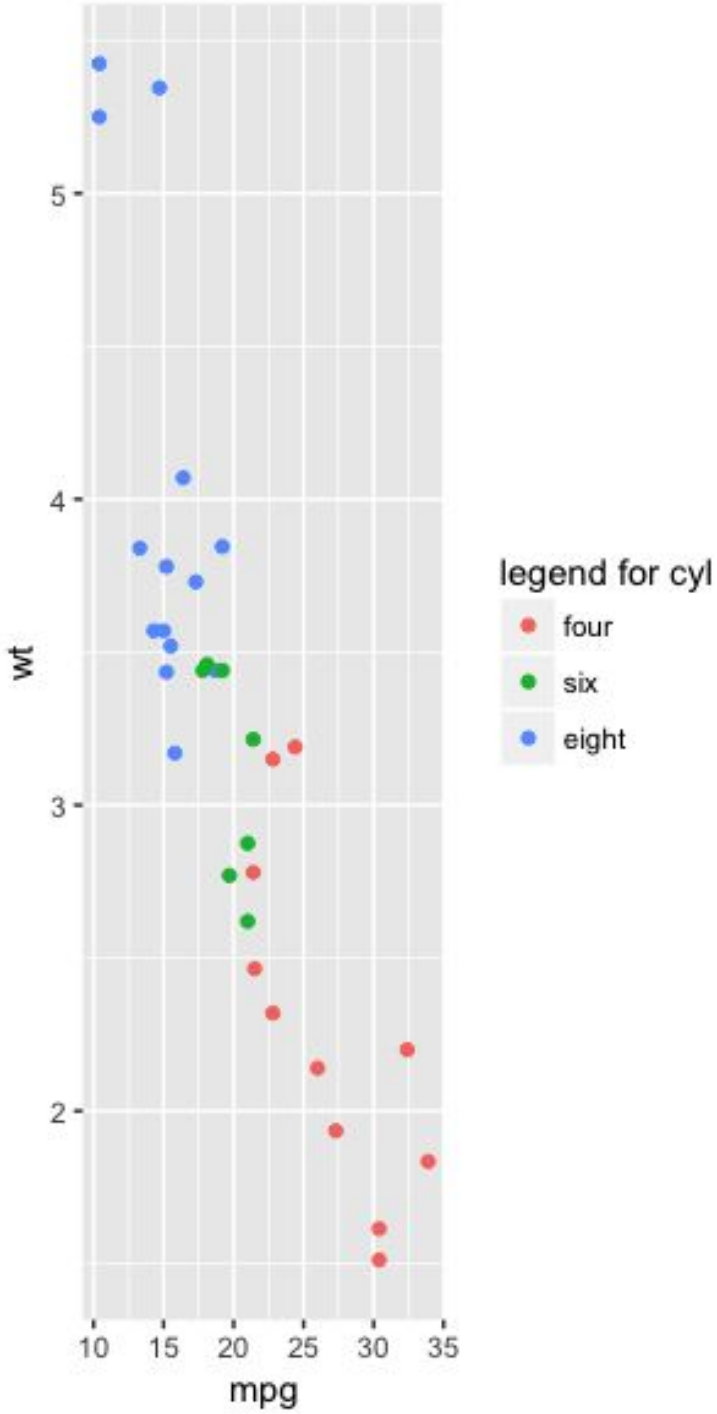
Changing name of legend

qplot(mpg, wt, data=mtcars,
colour=factor(cyl),
geom="point")+labs(colour="Legend-Name")



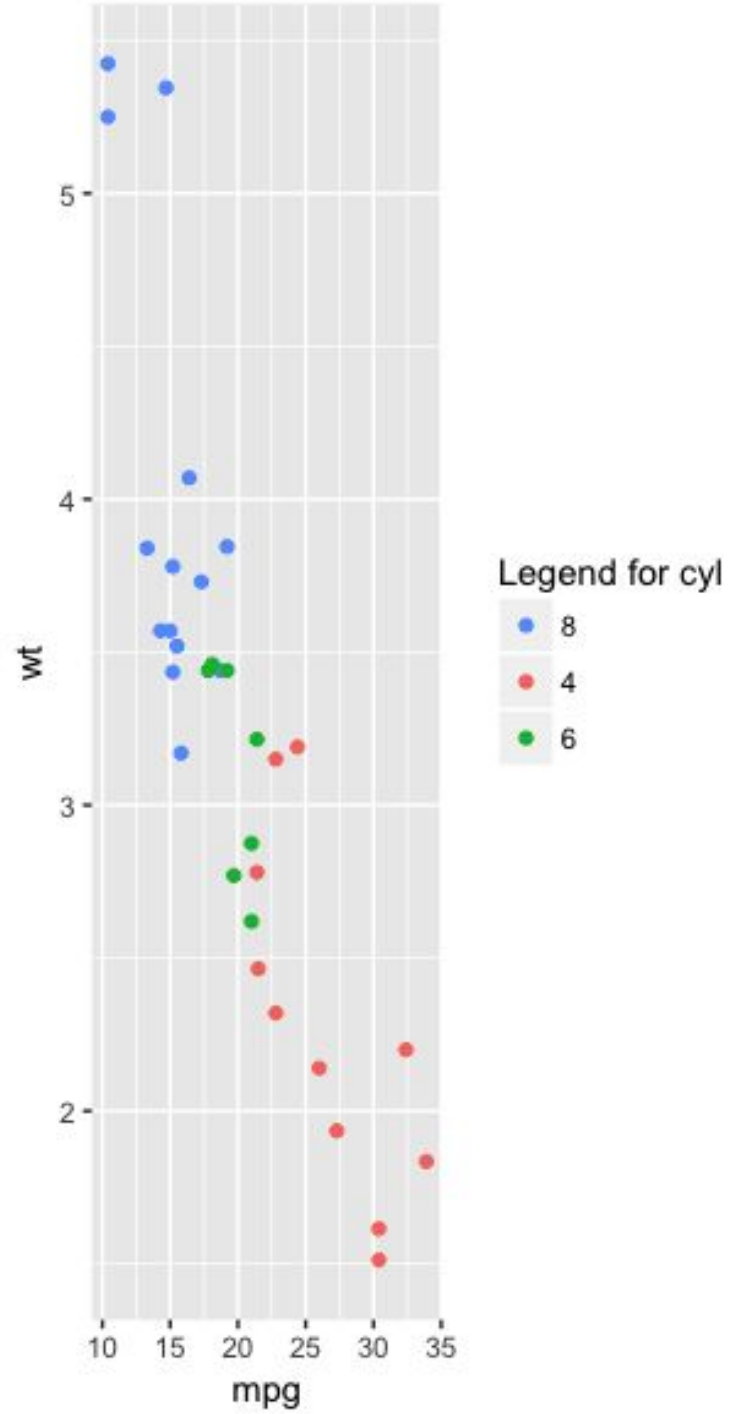
Changing labels on legend

qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point")+scale_colour_discrete(name= "legend for cyl", breaks=c("4","6","8"), labels=c("four","six","eight"))



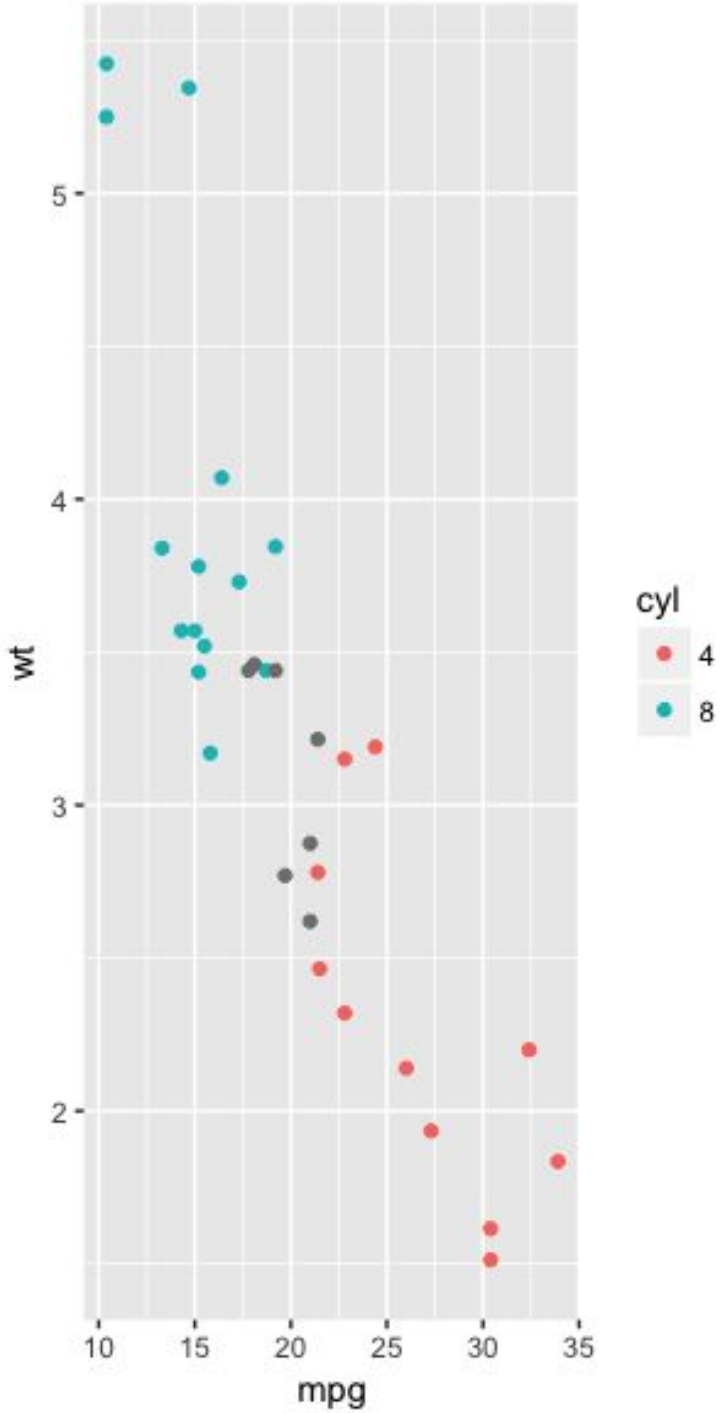
Reordering breaks

qplot(mpg, wt, data=mtcars, colour=factor(cyl), geom="point") + **scale_colour_discrete**(name="Legend for cyl", **breaks**=c("8","4","6"))



Dropping factors

```
mtcars2 <- transform(mtcars, cyl=factor(cyl))
levels(mtcars2$cyl)
qplot(mpg, wt, data=mtcars2, colour=cyl,
geom="point") +
scale_colour_discrete(limits=c("4", "8"))
```



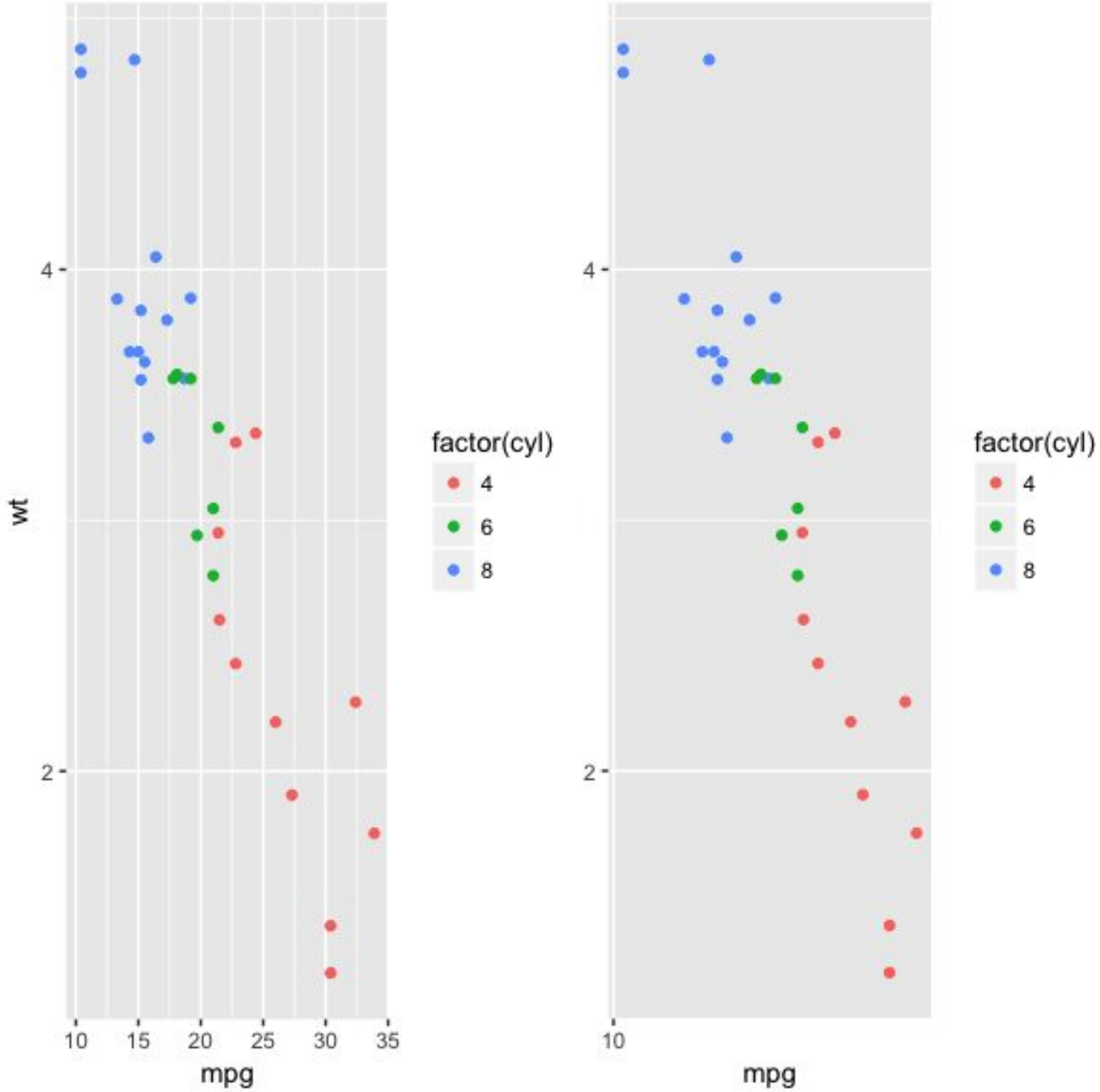
Using transformation

qplot(mpg, wt, data=mtcars,
colour=factor(cyl), geom="point")

**qplot(mpg, wt, data=mtcars,
colour=factor(cyl), geom="point") +**

scale_y_continuous(trans="log2")

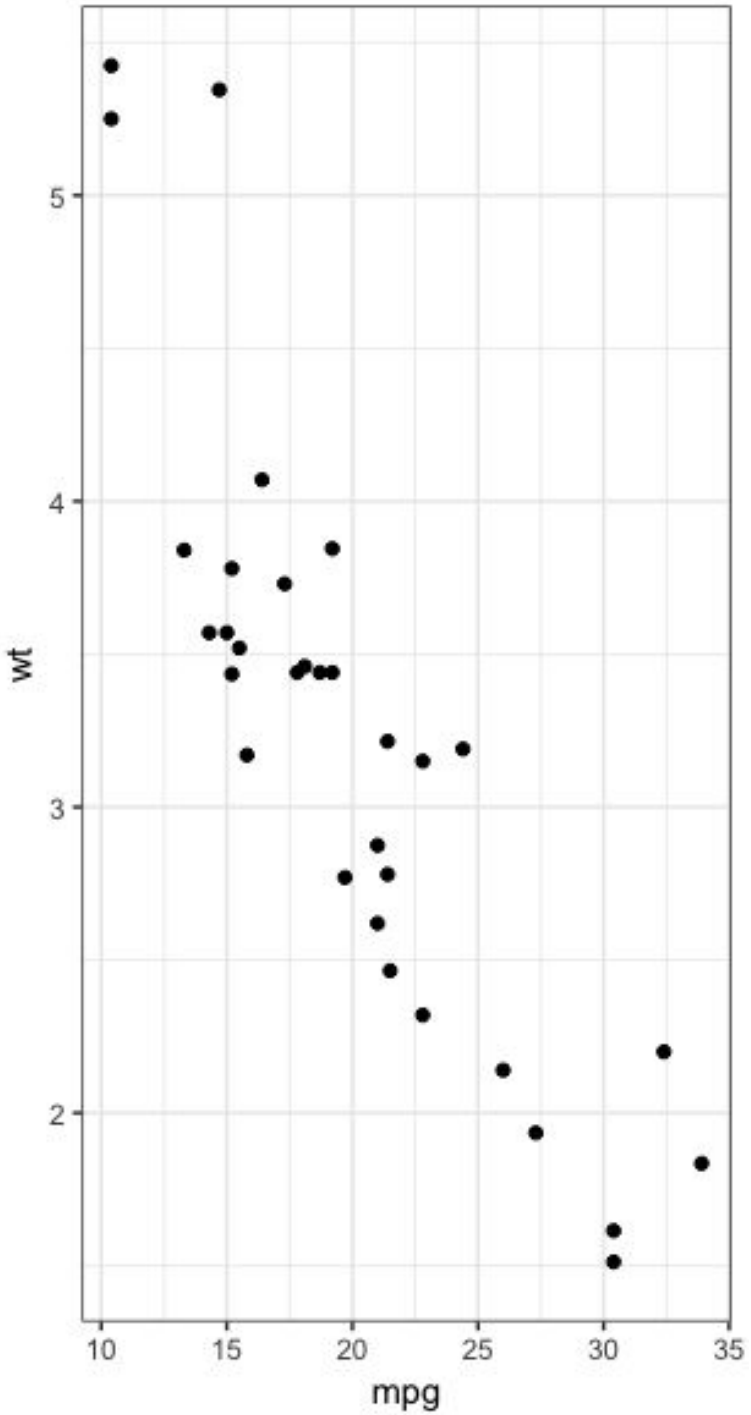
**qplot(mpg, wt, data=mtcars,
colour=factor(cyl), geom="point") +
scale_y_continuous(trans="log2") +
scale_x_log10()**

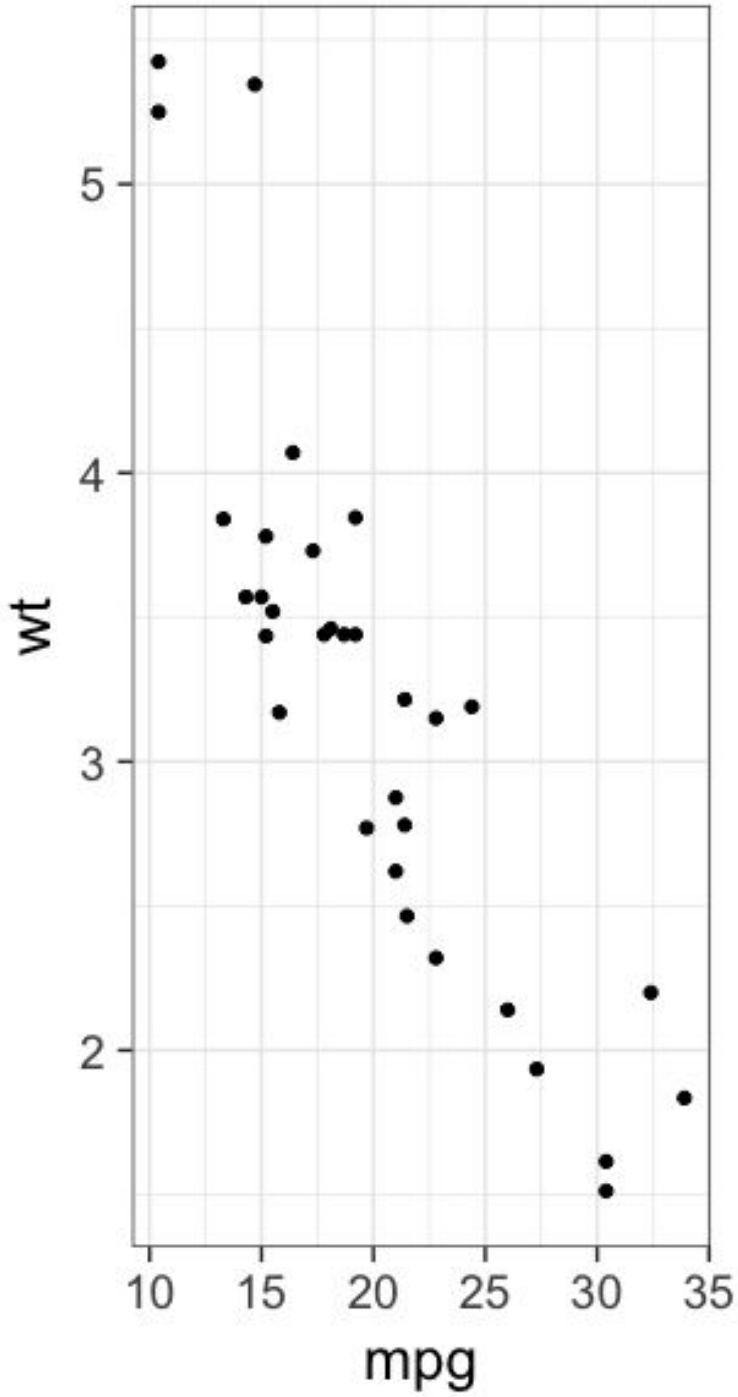


Use theme for plot only

qplot(mpg, wt, data=mtcars, geom="point")
+ theme_bw()

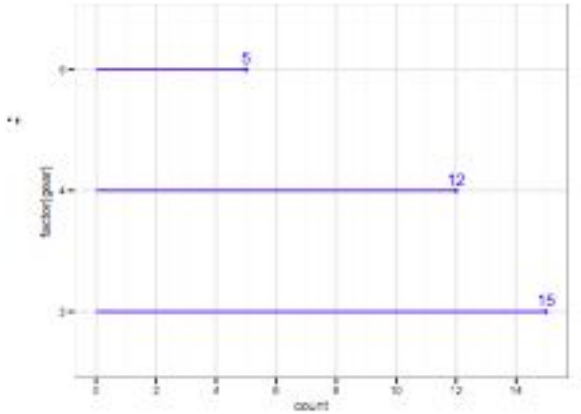
qplot(mpg, wt, data=mtcars, geom="point")
+ theme_bw(18)





Create barplot like lattice

qplot(x=factor(gear), ymax=..count.., ymin=0, ymax=..count.., label=..count.., data=mtcars, geom=c("pointrange", "text"), stat="bin", vjust=-0.5, color=I("blue")) + coord_flip() + theme_bw()



Create a pie-chart, rador-chart

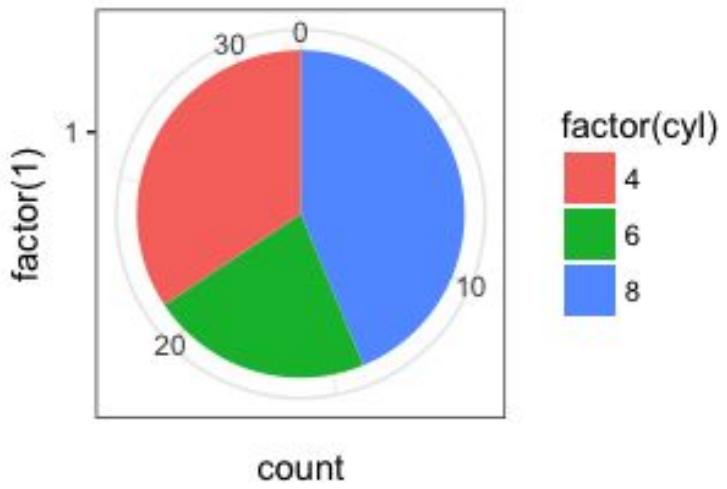
p.tmp <- ggplot(mtcars, aes(x=factor(1), fill=factor(cyl))) + geom_bar(width=1)

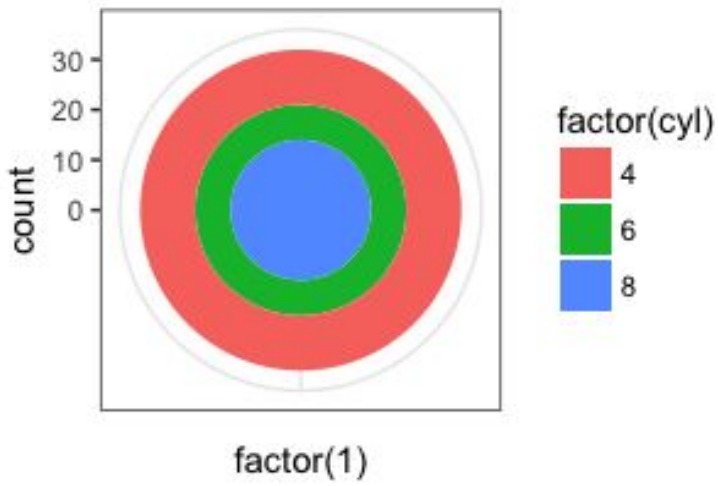
p.tmp

p.tmp + coord_polar(theta="y")

p.tmp + coord_polar()

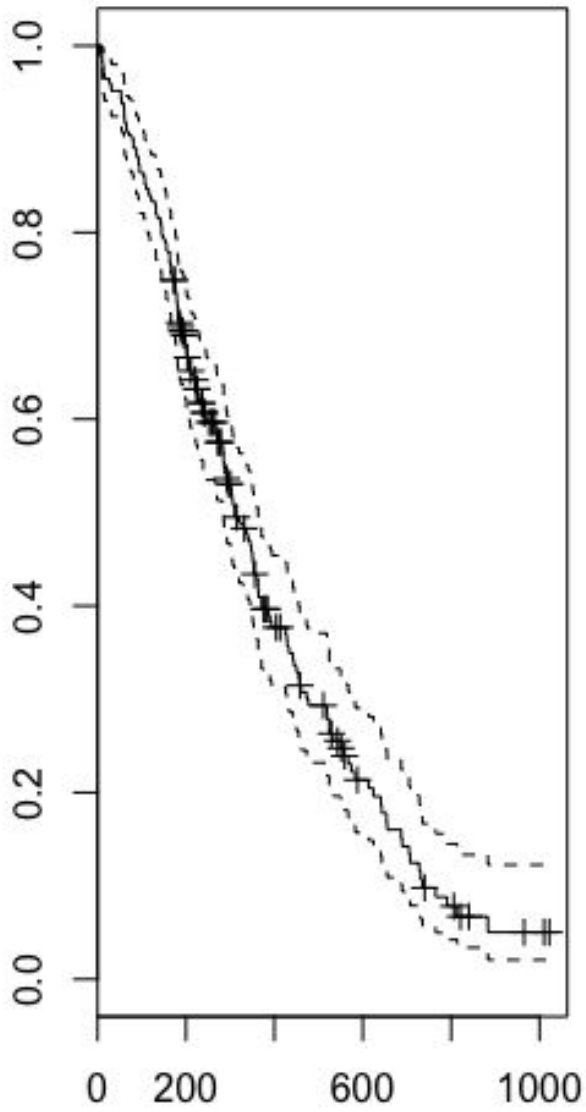
ggplot(mtcars, aes(factor(cyl), fill=factor(cyl))) + geom_bar(width=1) + coord_polar()





Create a kaplan-meier plot with survival package

```
t.Surv <- Surv(lung$time, lung$status)
t.survfit <- survfit(t.Surv~1, data=lung)
plot(t.survfit, mark.time=TRUE)
```



GGPLOT

The grammar of graphics of ggplot2

geom: the geometric "shape" used to display data-bar, point, line, ribbon, text

aesthetic: an attribute controlling how geom is displayed-x position, y position, color, fill, shape, size

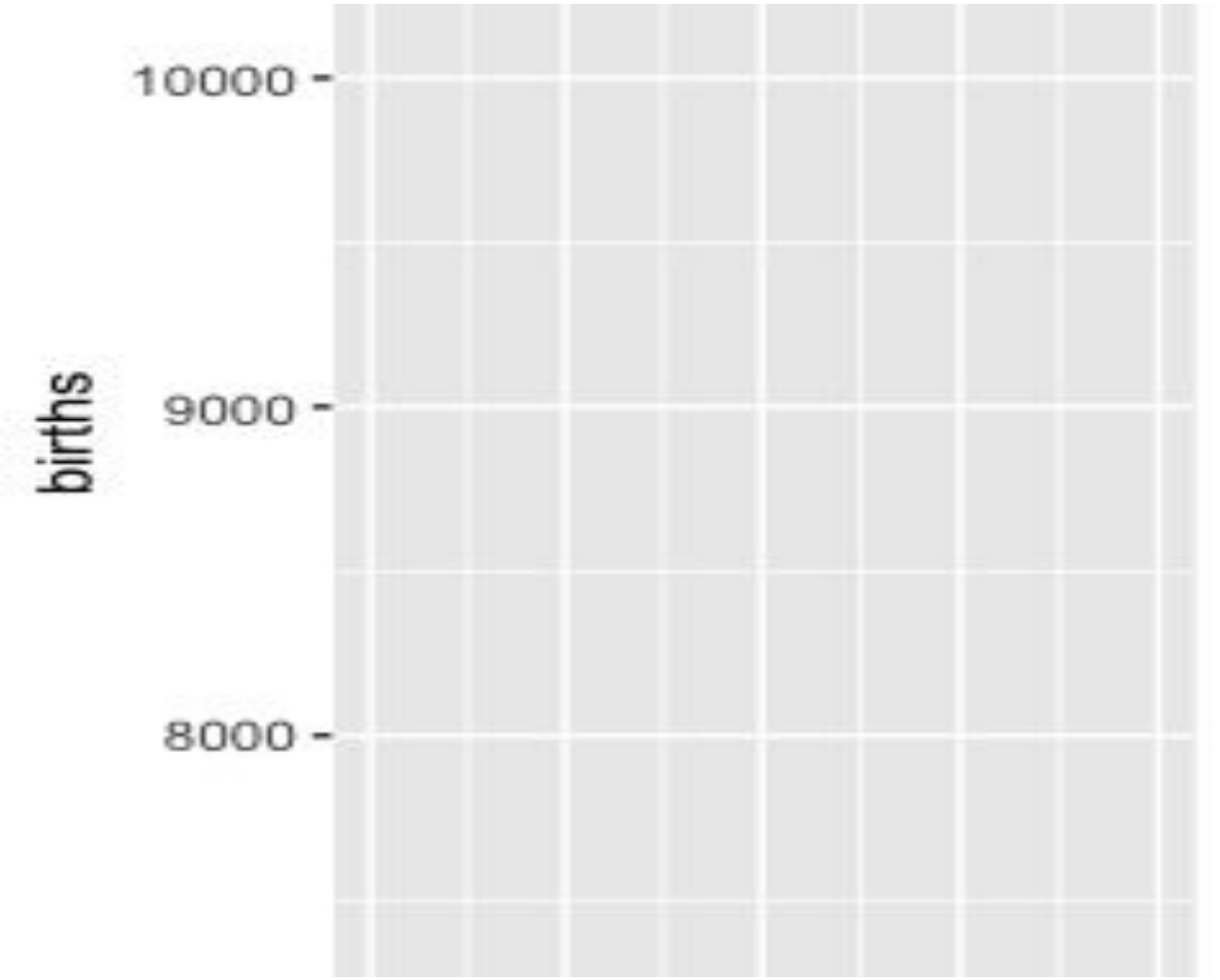
stat: a transformation applied to data before geom gets it-histograms work on binned

scale: conversion of raw data to visual display-particular assignment of colors, shapes, sizes

ggplot() establishes the default data and aesthetics for plot

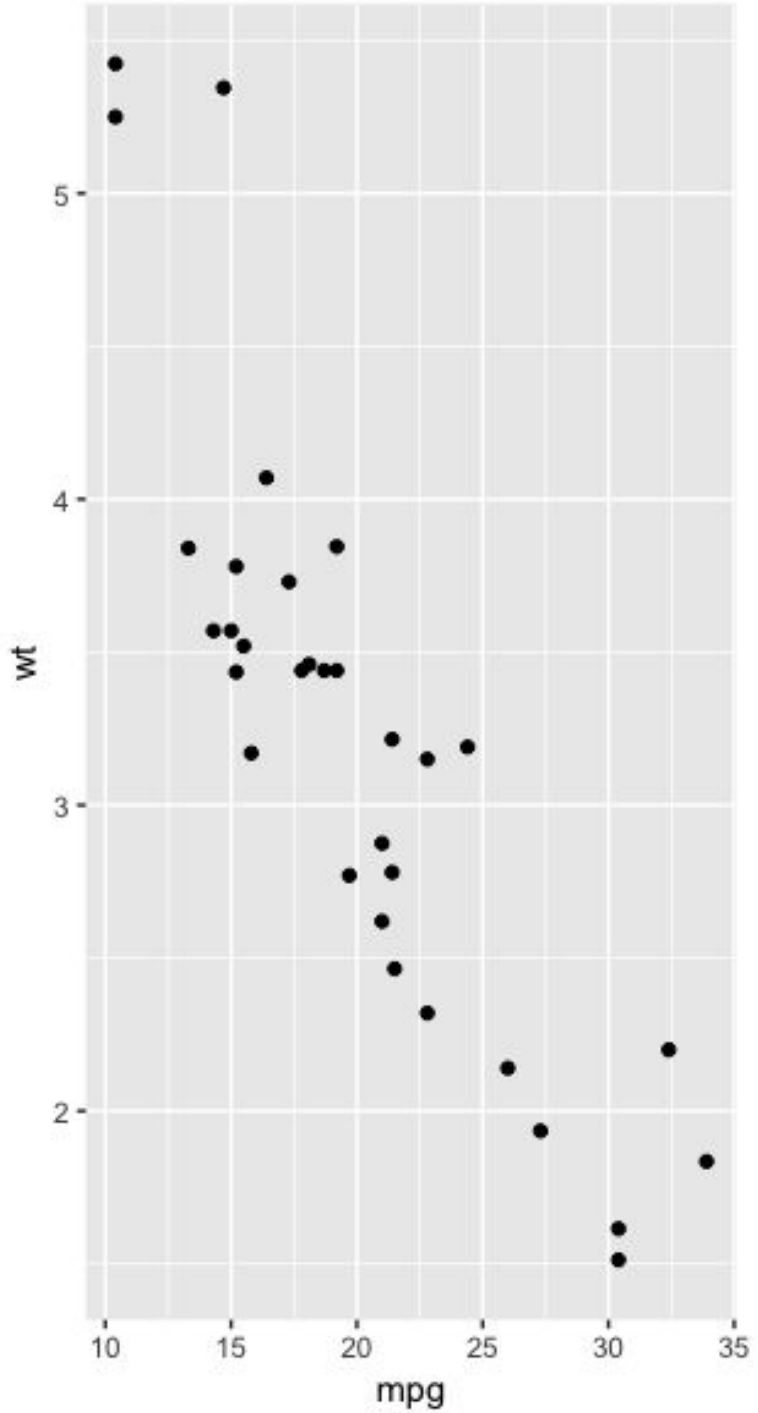
##It means for creating a sheet for graphs

ggplot(data=Births, aes(x=date, y=births, color=wday))



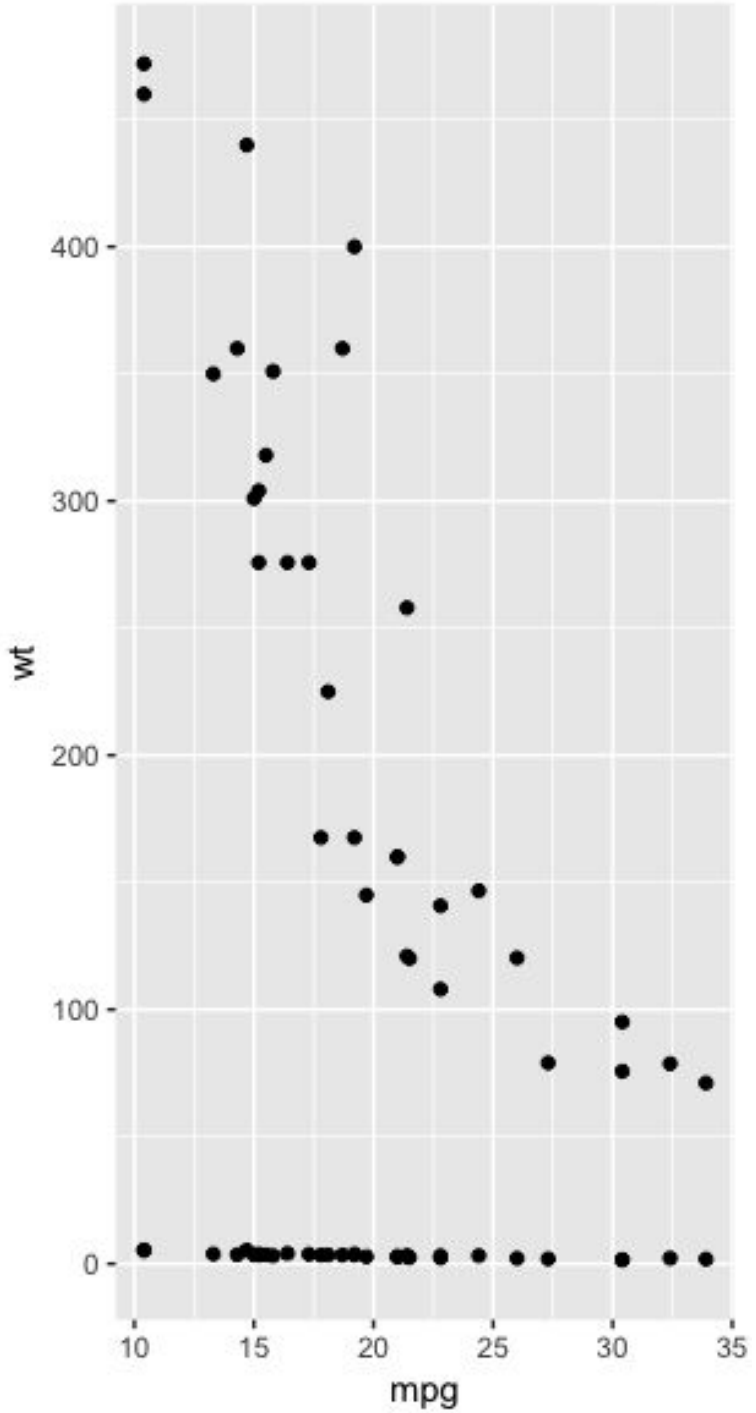
Scatter plot

p.tmp+geom_point()



Add an additional layer with different mapping

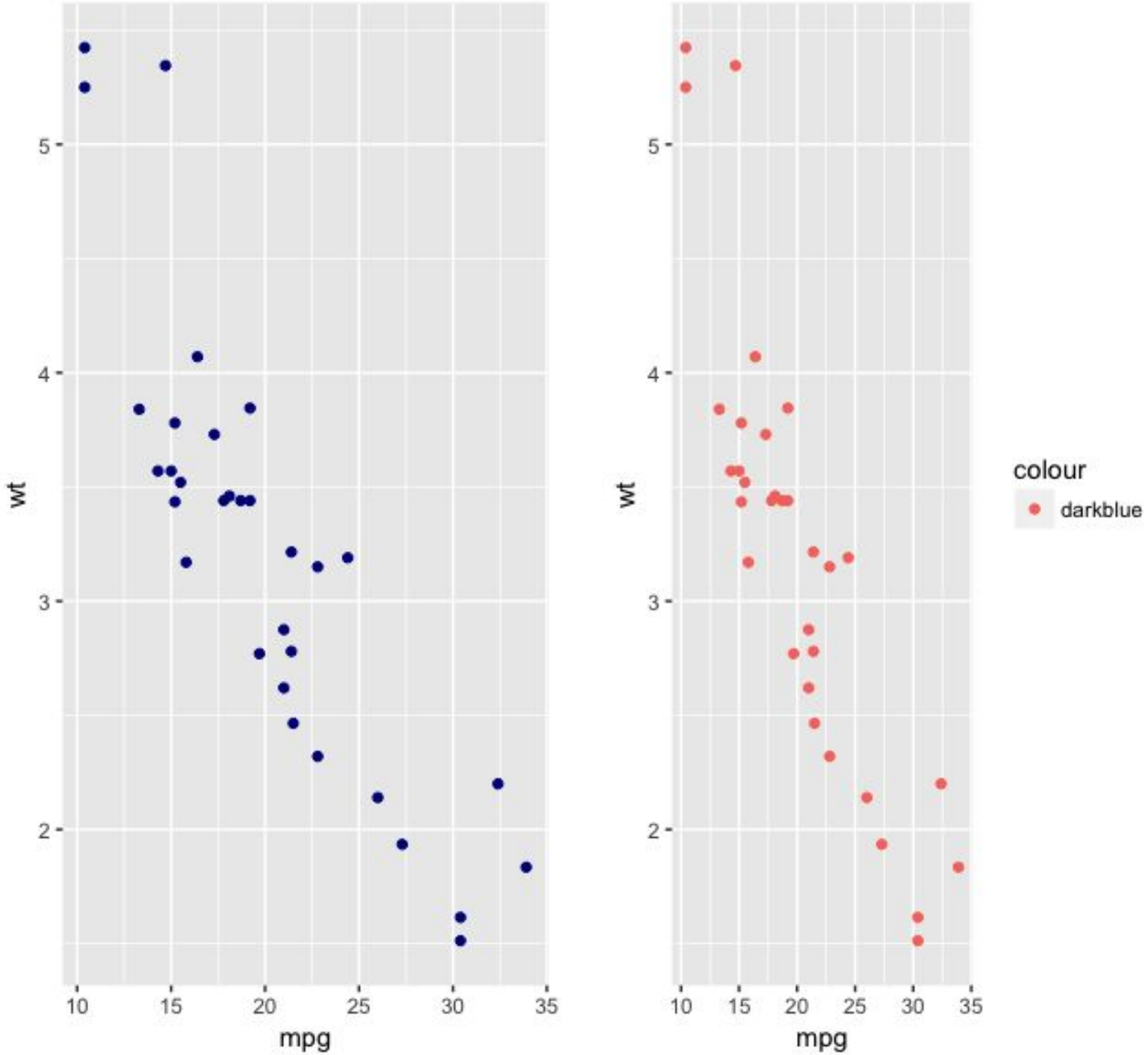
p.tmp+geom_point()+geom_point(aes(y=
))



Setting aesthetics instead of mapping

p.tmp+geom_point(color="darkblue")

p.tmp+geom_point(aes(color="darkblue"



Dealing with overplotting(hollow points, pixel points, alpha(0-1))

t.df=data.frame(x=rnorm(2000), y=rnorm(2000))

p.norm=ggplot(t.df, aes(x,y))

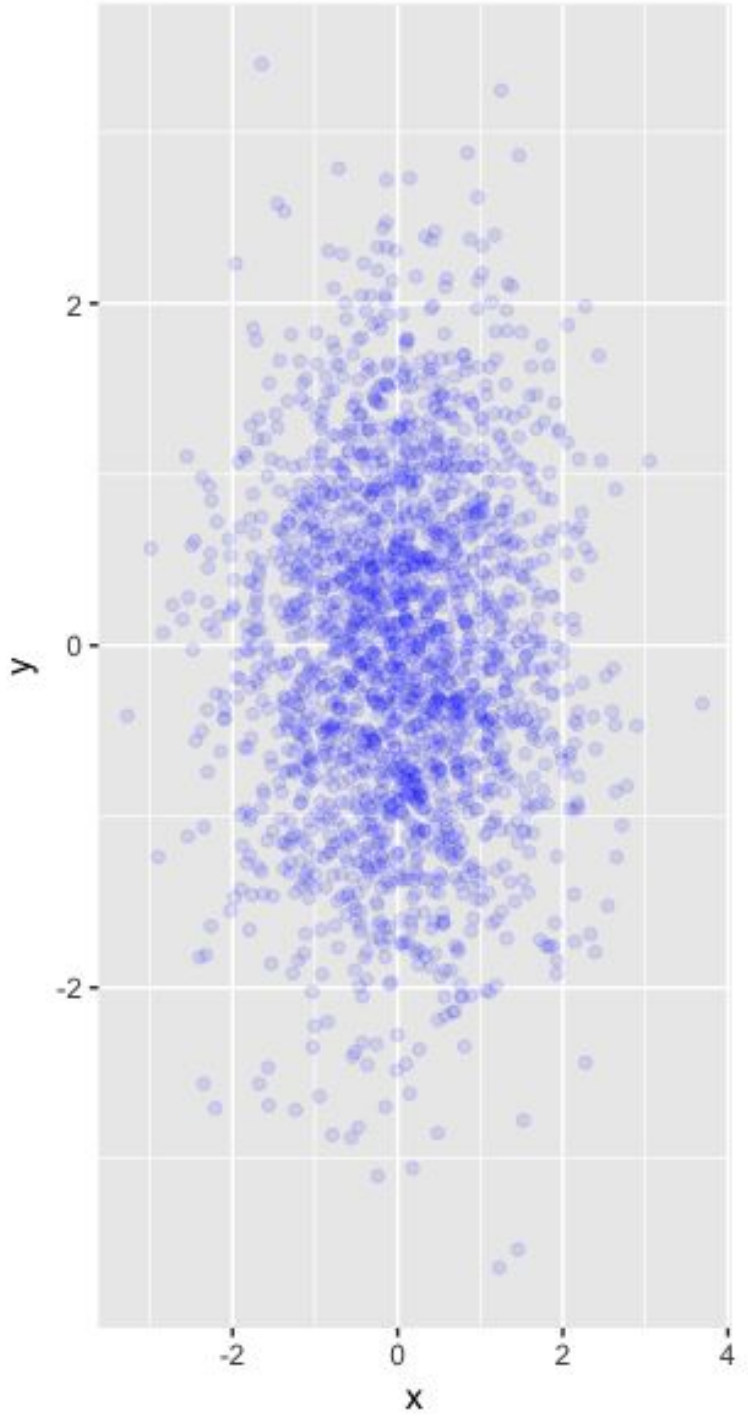
p.norm+geom_point()

p.norm+geom_point(shape=1)

p.norm+geom_point(shape=".")

p.norm+geom_point(colour=alpha("blac
1/2))

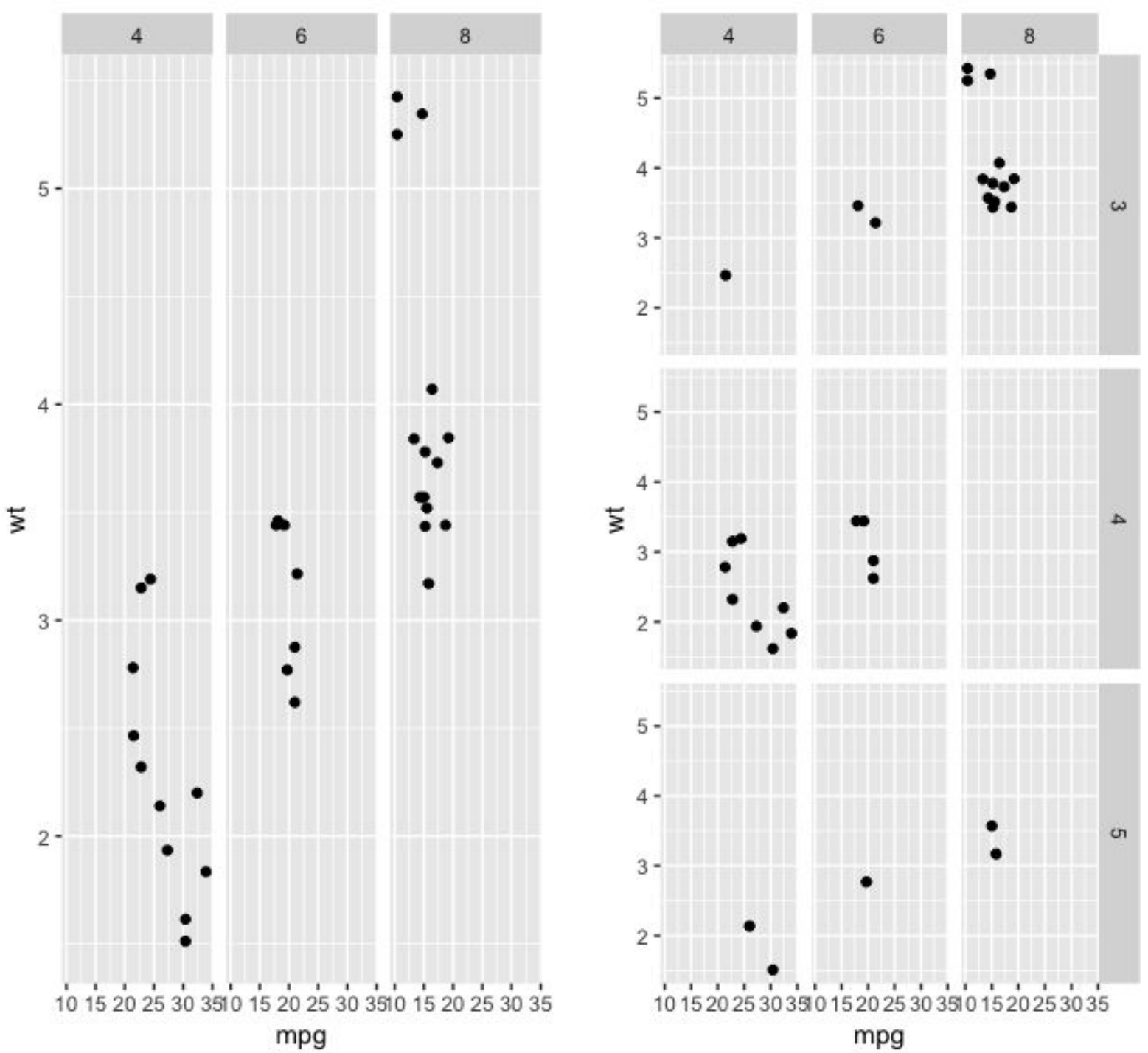
p.norm+geom_point(colour=alpha("blue
10))



Using facets

qplot(mpg, wt, data=mtcars, facets=.~cyl
geom="point")

qplot(mpg, wt, data=mtcars, facets=gear
geom="point")



Facet_wrap/facet_grid

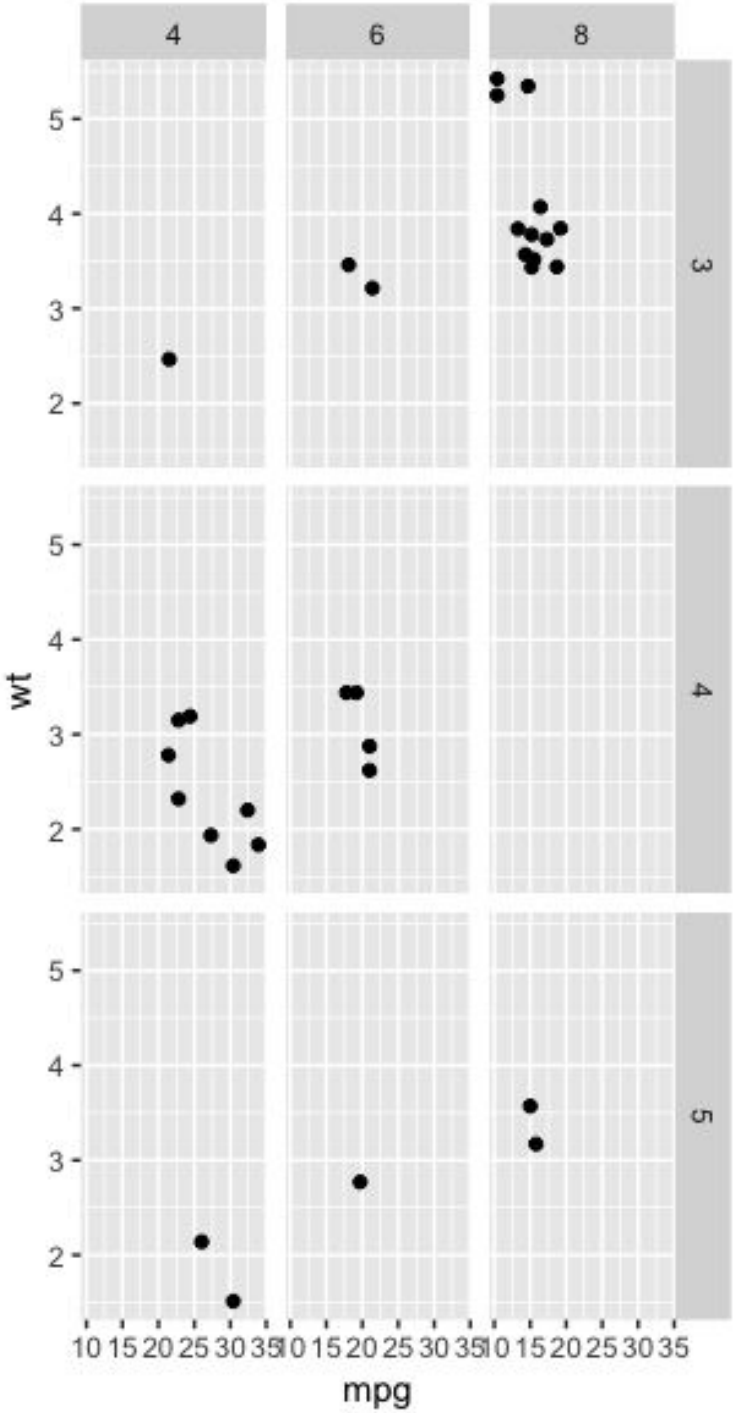
p.tmp=ggplot(mtcars, aes(mpg, wt))+geom_point()

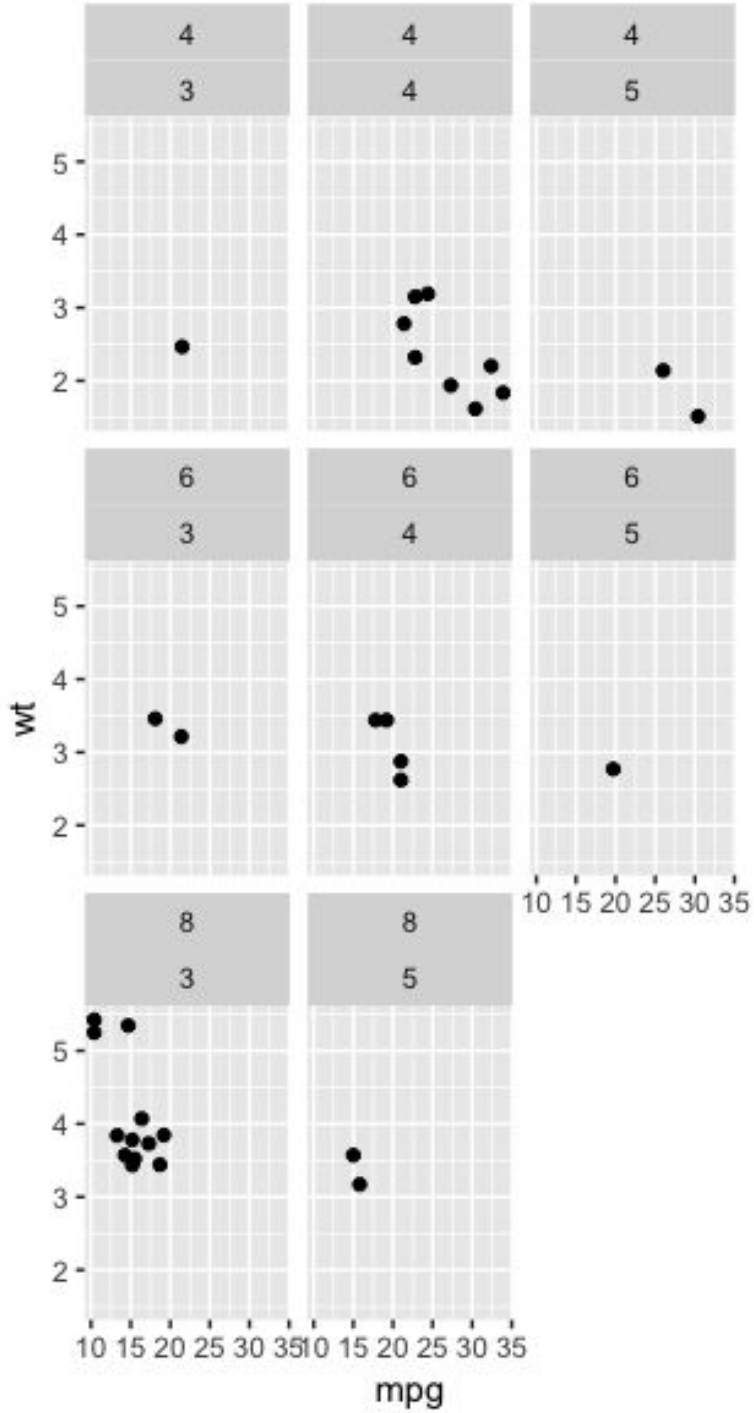
p.tmp+facet_wrap(~cyl)

p.tmp+facet_wrap(~cyl, ncol=3)

p.tmp+facet_grid(gear~cyl)

p.tmp+facet_wrap(~cyl+gear)



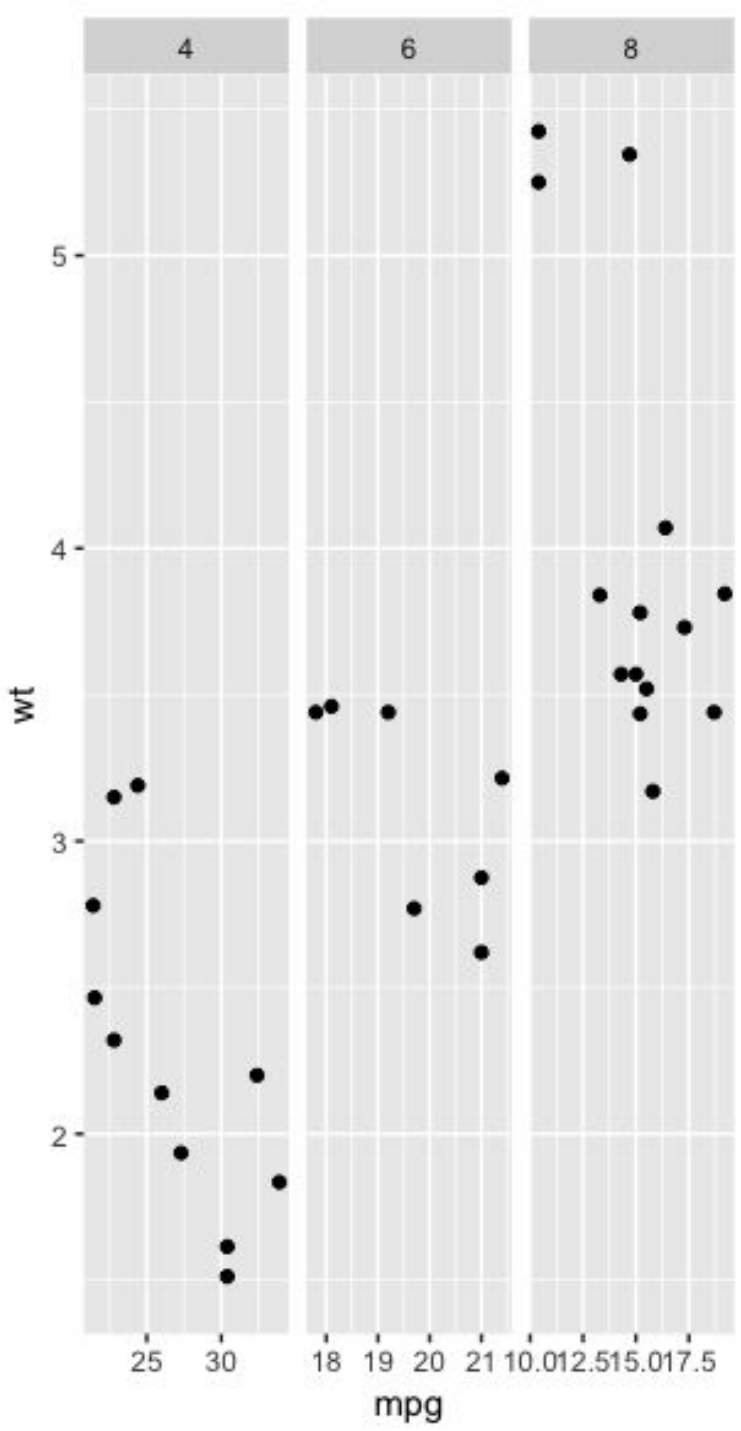


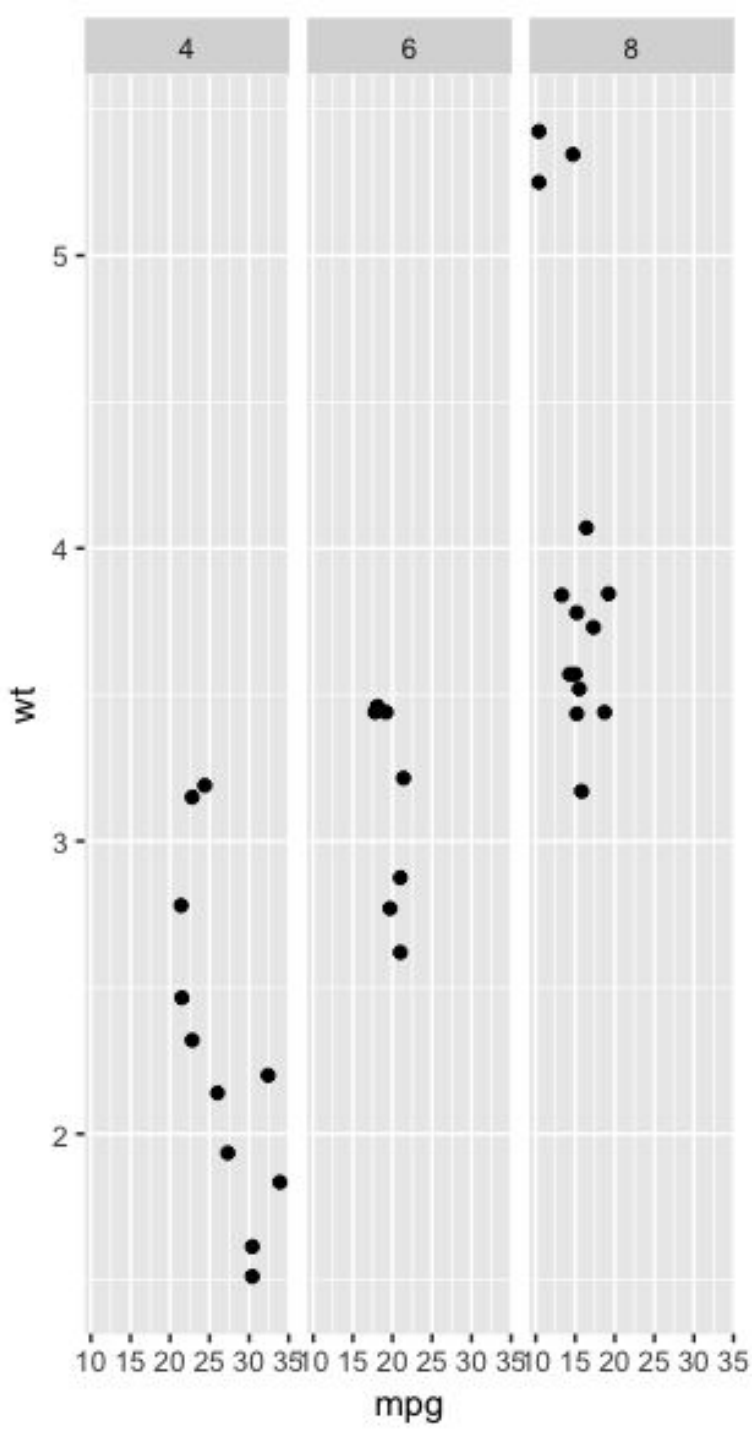
Controlling scales in facets

p.tmp+facet_wrap(~cyl, scales="free")

p.tmp+facet_wrap(~cyl, scales="free_x"

p.tmp+facet_wrap(~cyl, scales="fixed")





Using scales

p.tmp=qplot(cut, data=diamonds, geom="bar", fill=cut)

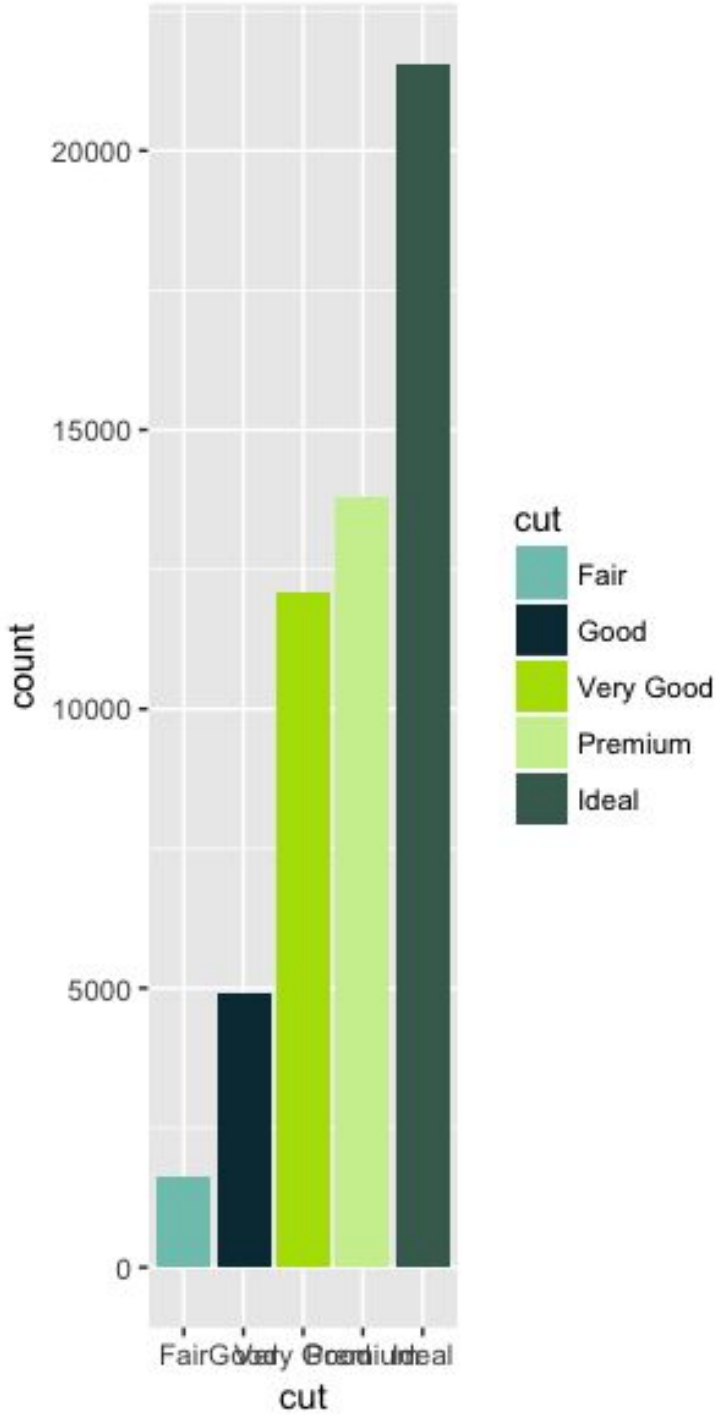
p.tmp

p.tmp+scale_fill_brewer()

p.tmp+scale_fill_brewer(palette = "Paired")

RColorBrewer::display.brewer.all()

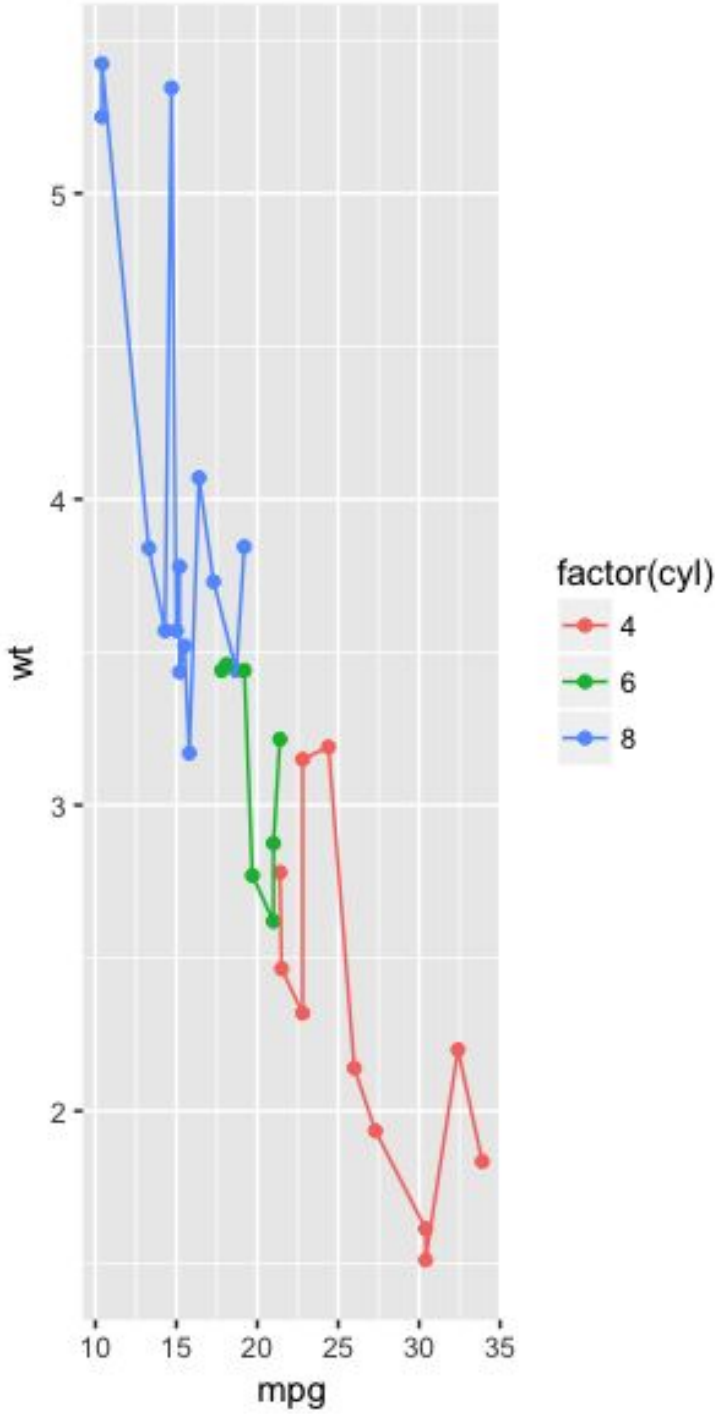
p.tmp + scale_fill_manual(values=c("#7fc6bc","#083642","#b1df01","#cdef9c","#466b5d"))



Using ggplot-syntax with qplot

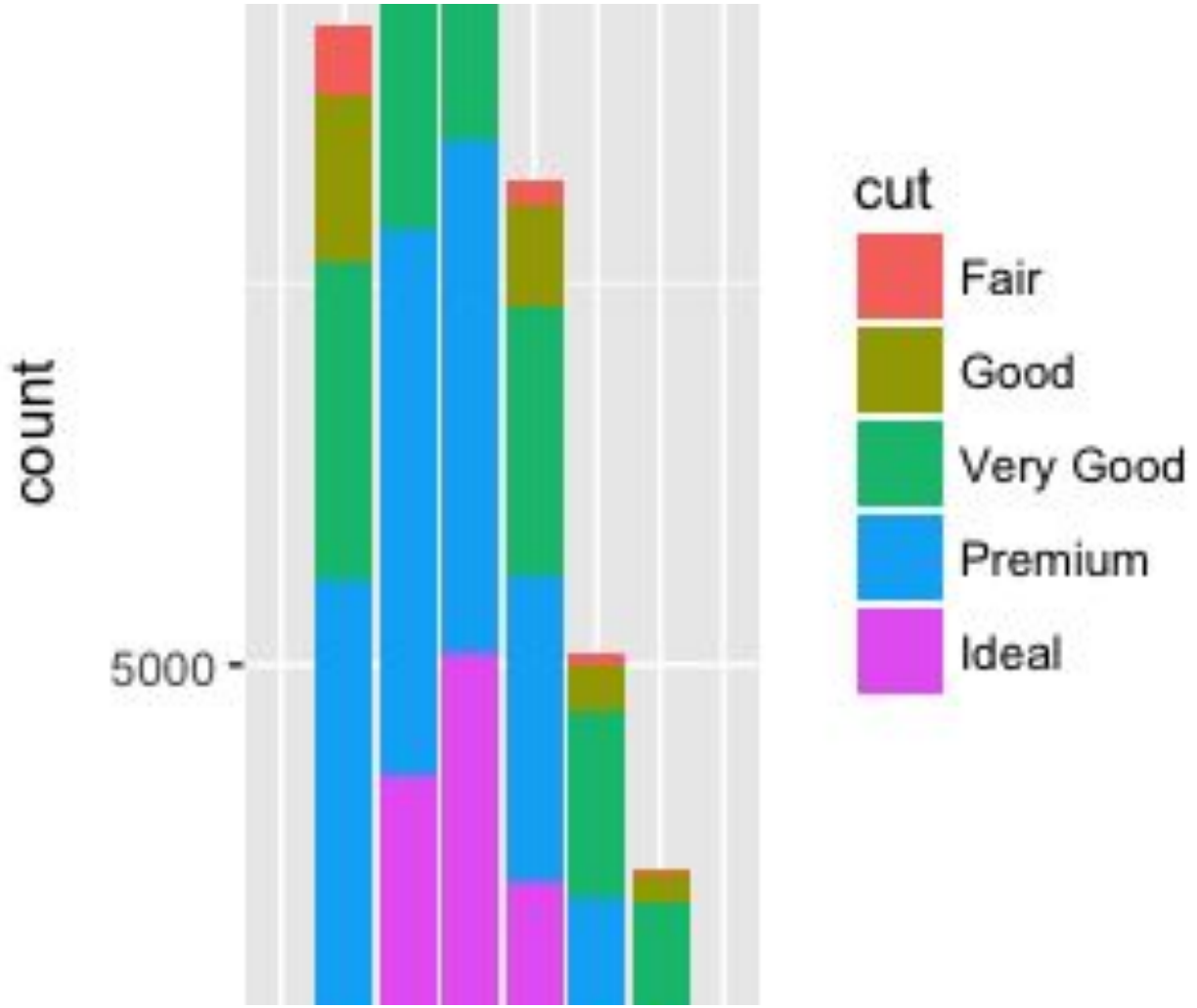
qplot(mpg, wt, data=mtcars,
color=factor(cyl),
geom="point")+geom_line()

qplot(mpg, wt, data=mtcars,
color=factor(cyl), geom=c("point","line"))



ggplot histogram

ggplot(diamonds, aes(clarity, fill=cut))+geom_bar()

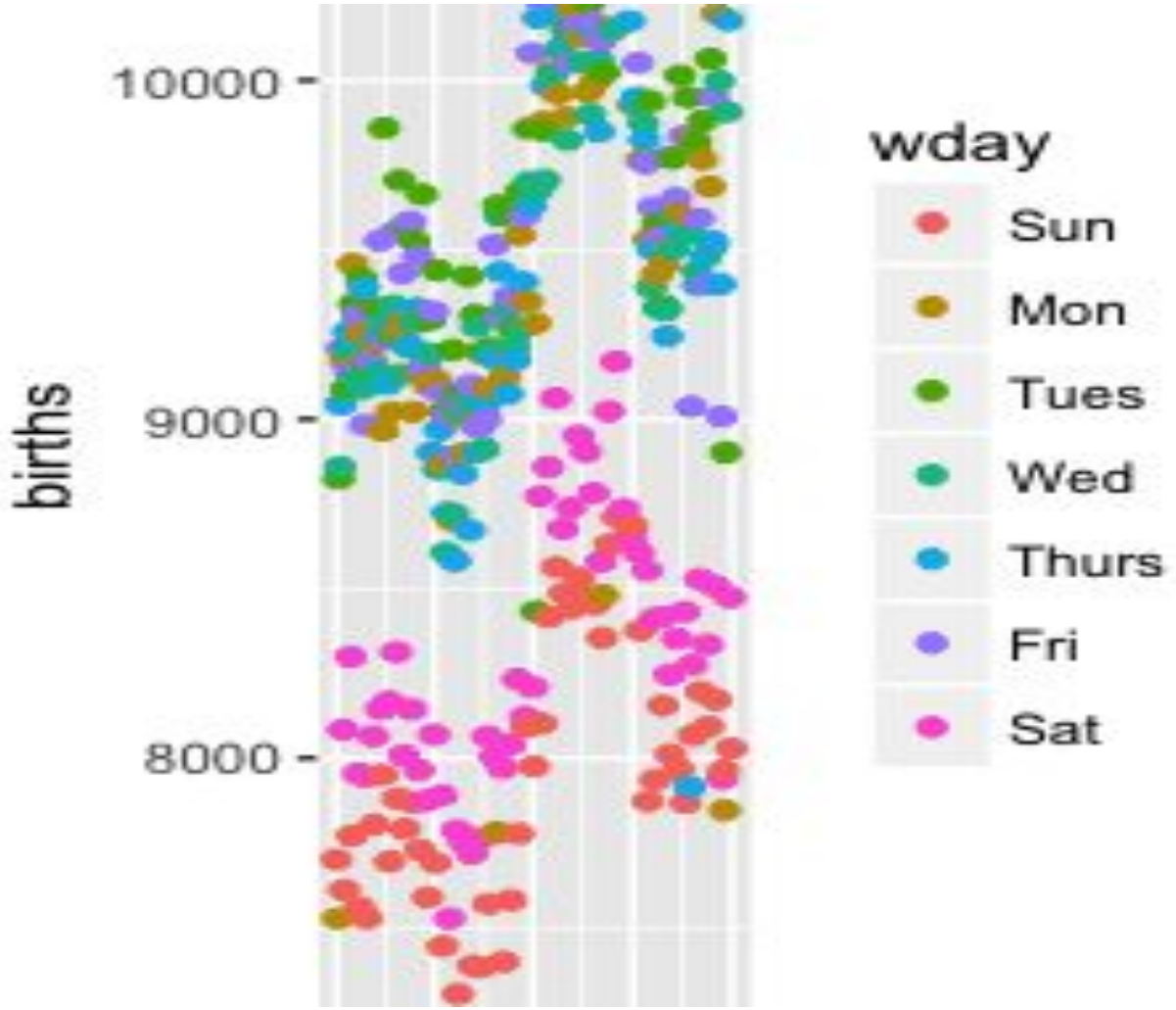


Add layers by supplying a geometry

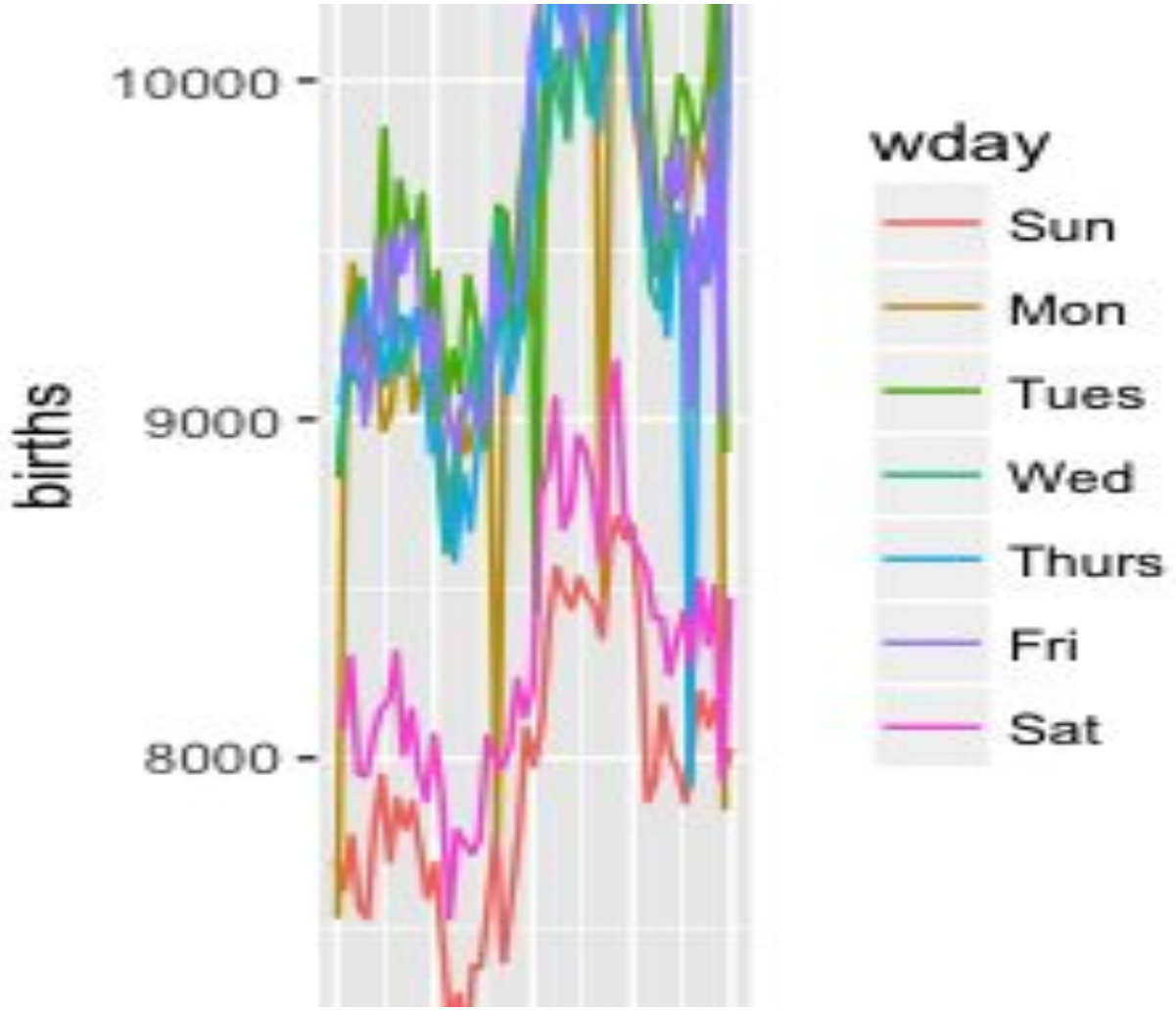
apropos("^geom_")

apropos("^stat_")

ggplot(data=Births, aes(x=date, y=births, color=wday)) + geom_point()

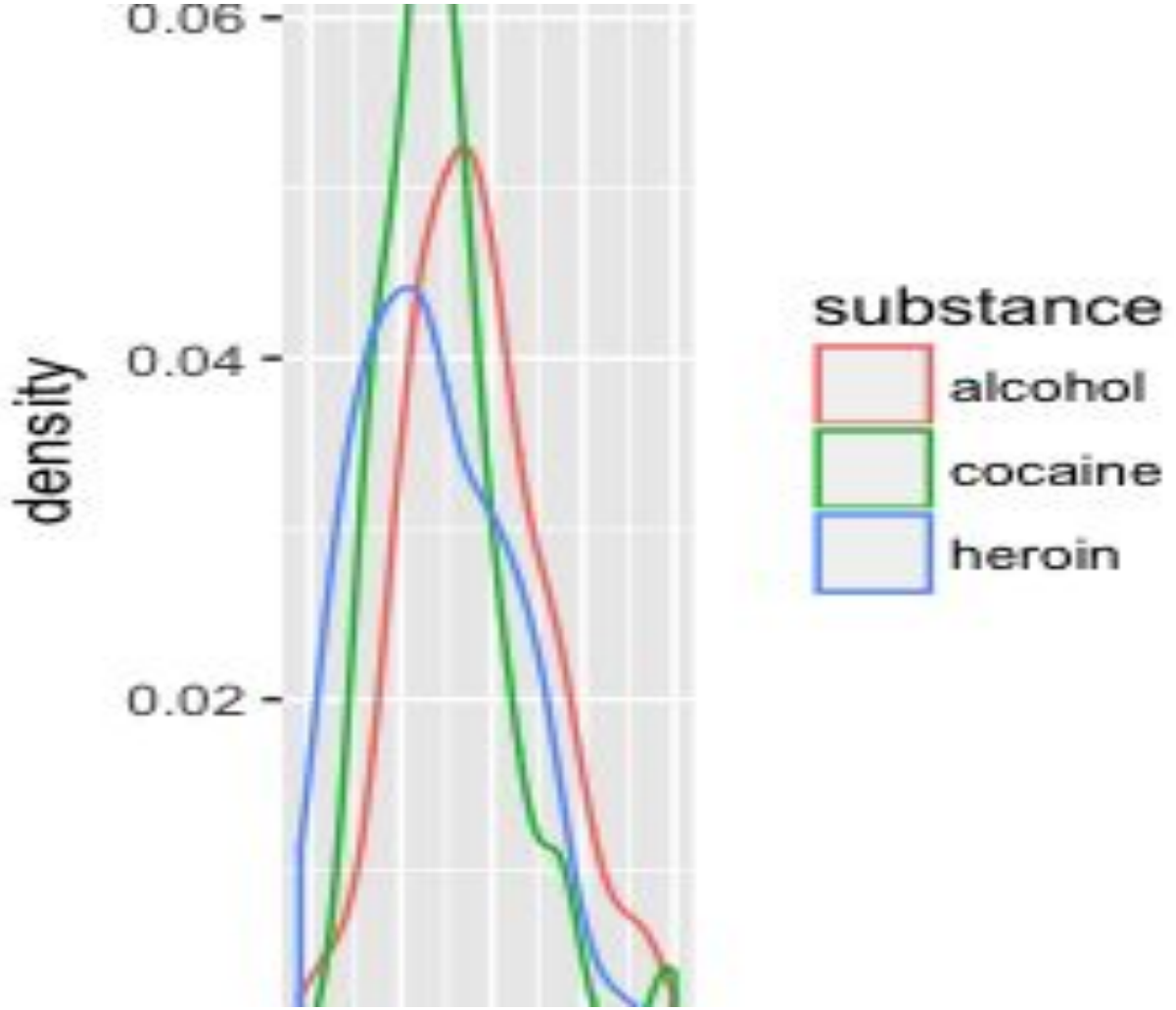


ggplot(data=Births, aes(x=date, y=births, color=wday)) + geom_line()



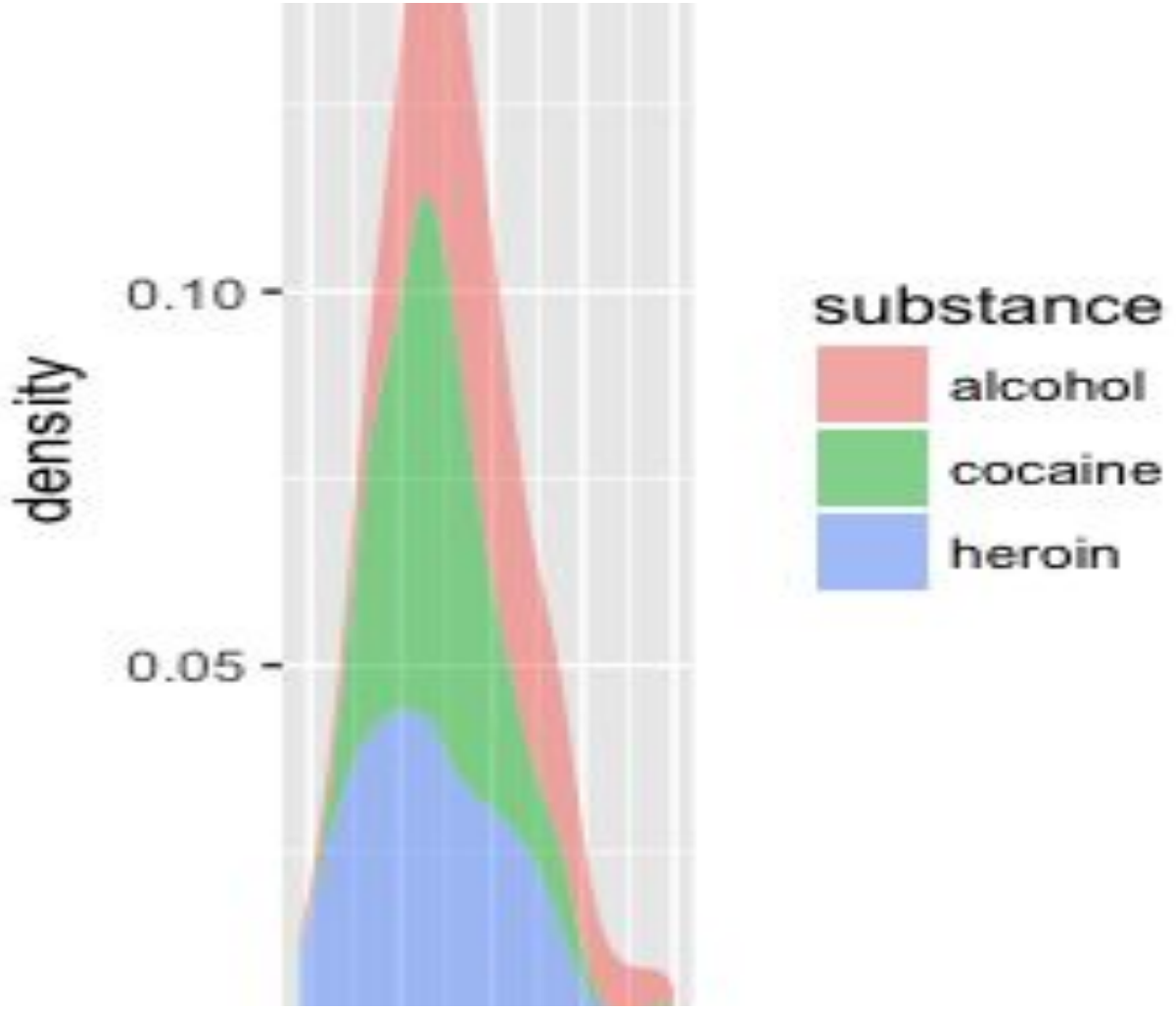
#Uses stat_density with this geom by default

ggplot(data=HELPrct, aes(x=age, color=substance))+geom_density()



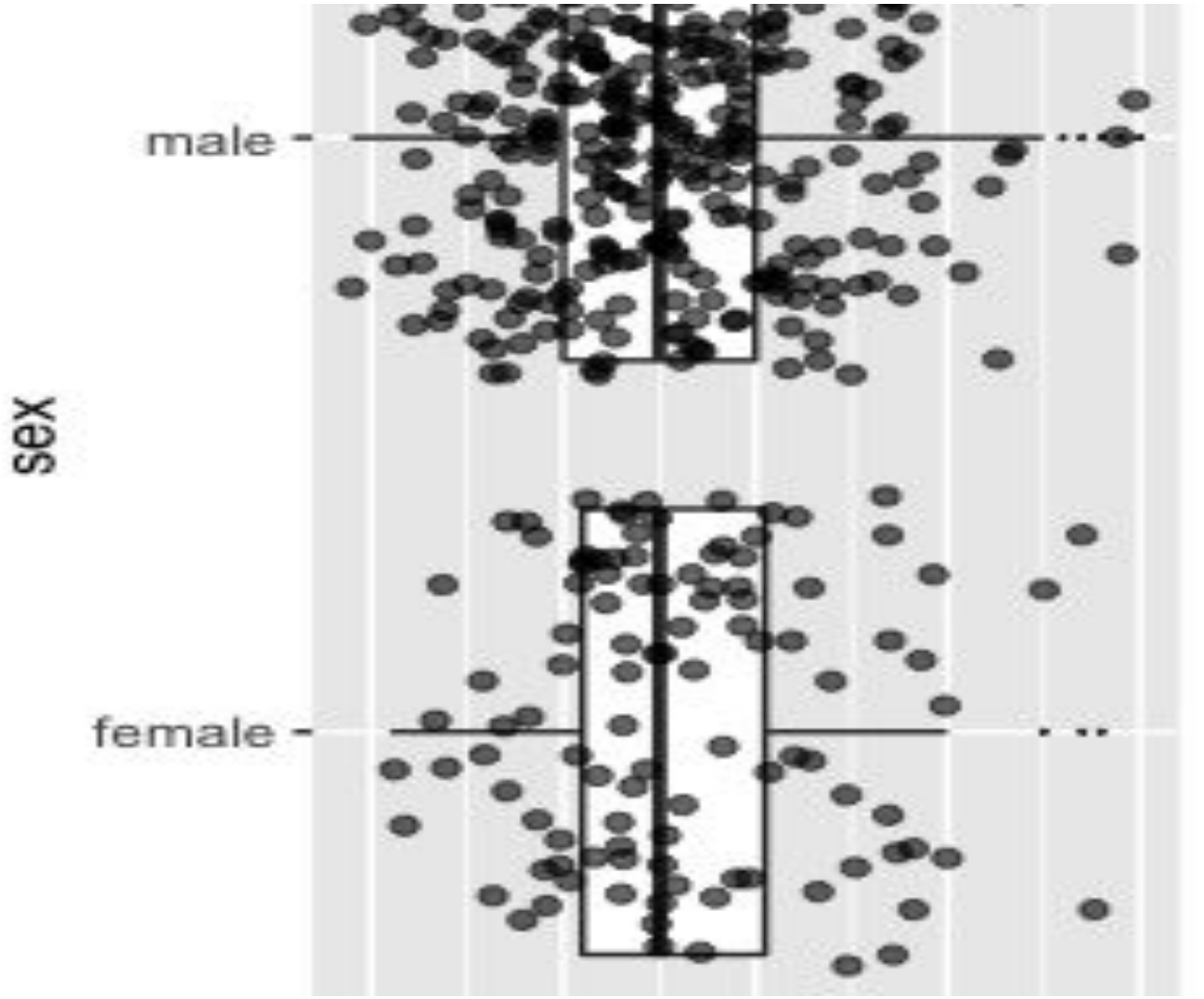
#Uses a geom_area by default for this stat

ggplot(data=HELPrct, aes(x=age, fill=substance)) + stat_density(alpha=0.5)



#Use multiple layers

```
ggplot(data=HELPrct, aes(x=sex,
y=age))+geom_boxplot(outlier.size=0) +
geom_jitter(alpha=0.6)+coord_flip()
```

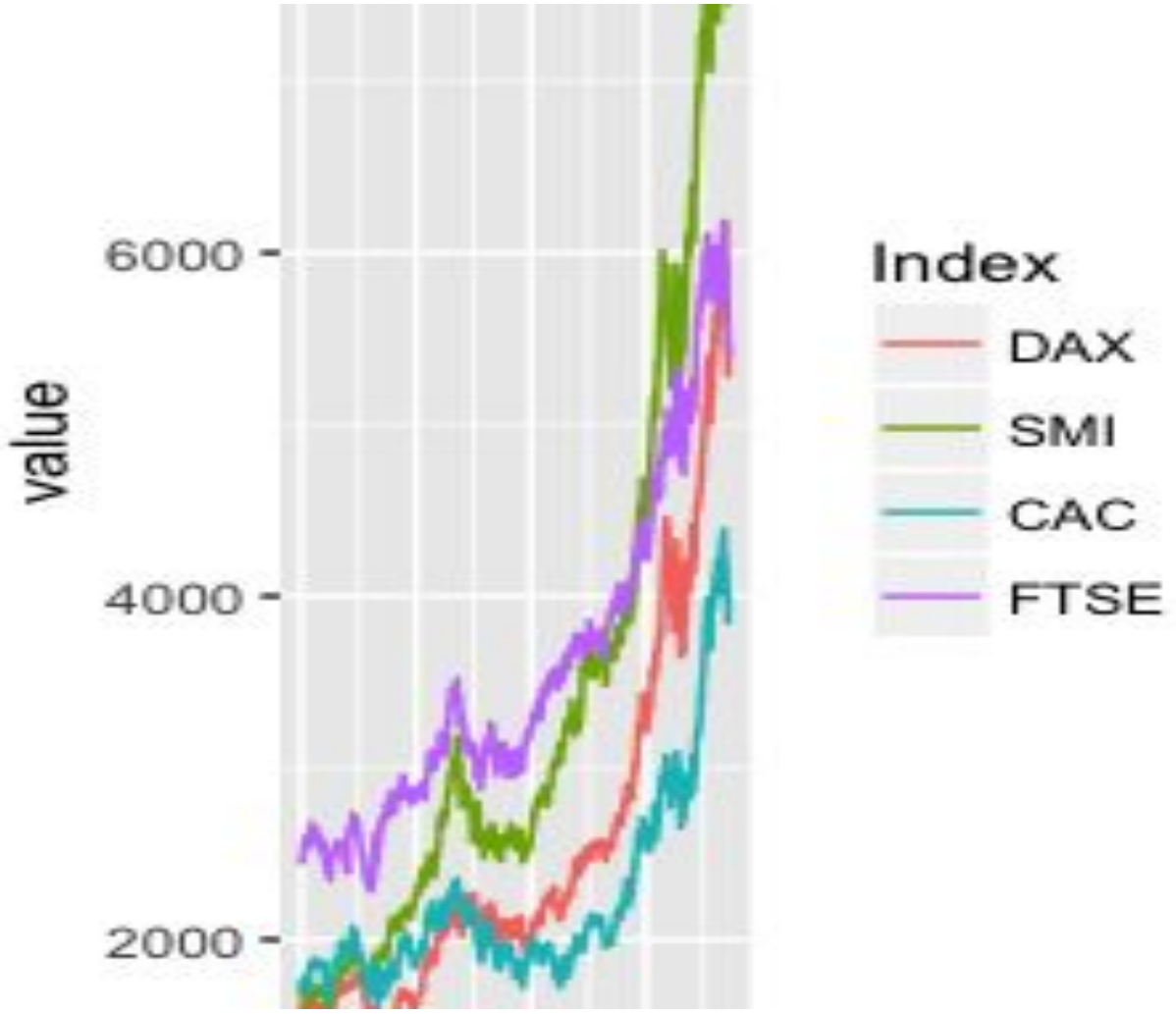


MELITING DATA--PIVOTS

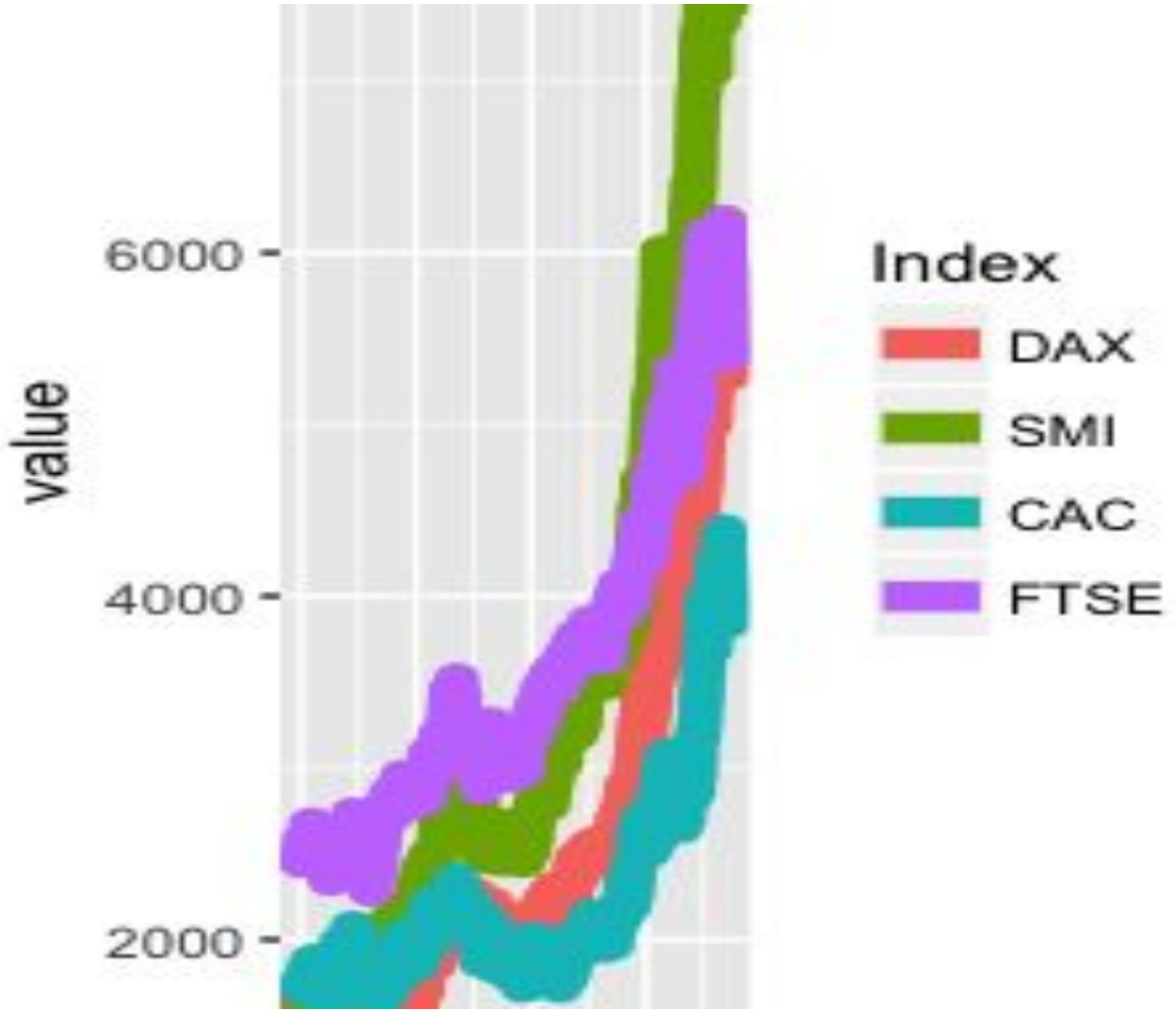
```
#Load the package
library(ggplot2)
head(EuStockMarkets)

#Load the package for the melt command
library(reshape)

#Make copy of the dataset
ds=as.data.frame(EuStockMarkets)

#Original data does not have day information
ds$day=seq(nrow(ds))
head(ds)

#Melt collapses clumns into one field
EuStock=melt(ds, id=c("day"))

#Rename the new field with the column name to "Index"
names(EuStock)[2]="Index"
head(EuStock)
```

#Plot the stock values with color

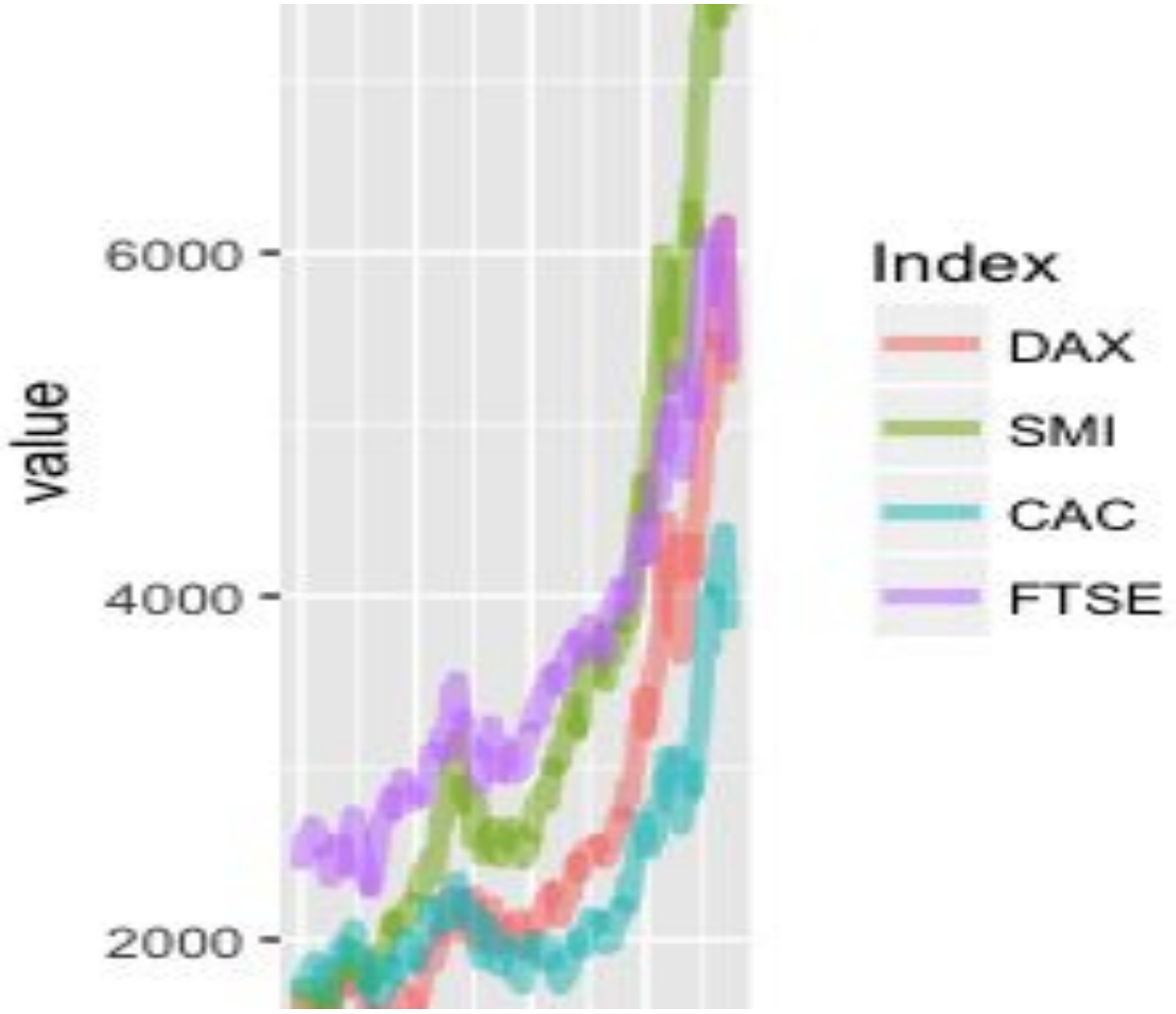
ggplot(EuStock, aes(day, value, color=Index))+geom_line()



```
ggplot(EuStock, aes(day, value,
color=Index))+geom_line(size=3)
```

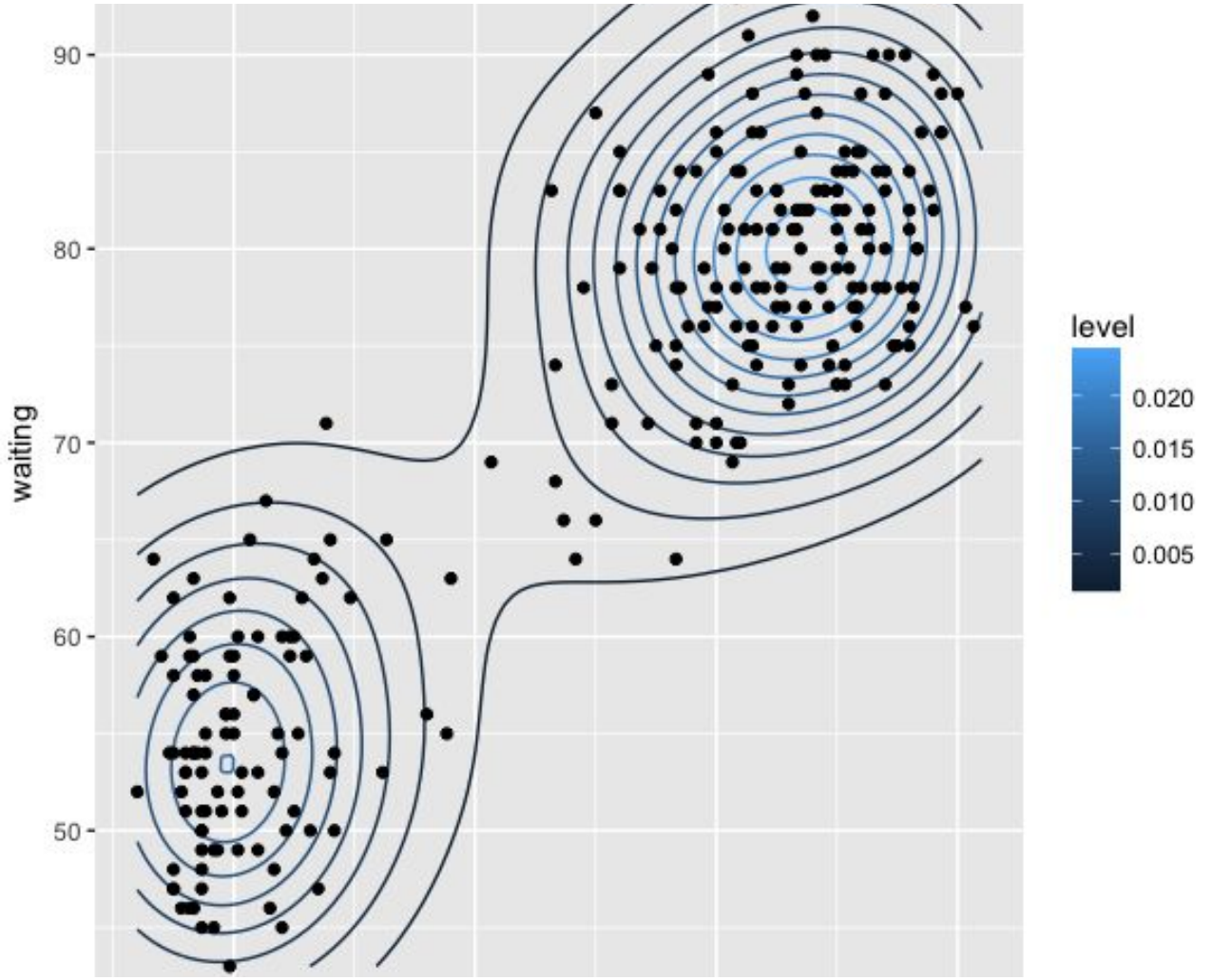


```
ggplot(EuStock, aes(day, value,
color=Index))+geom_line(size=1.5,
alpha=0.5)
```



Base Plot

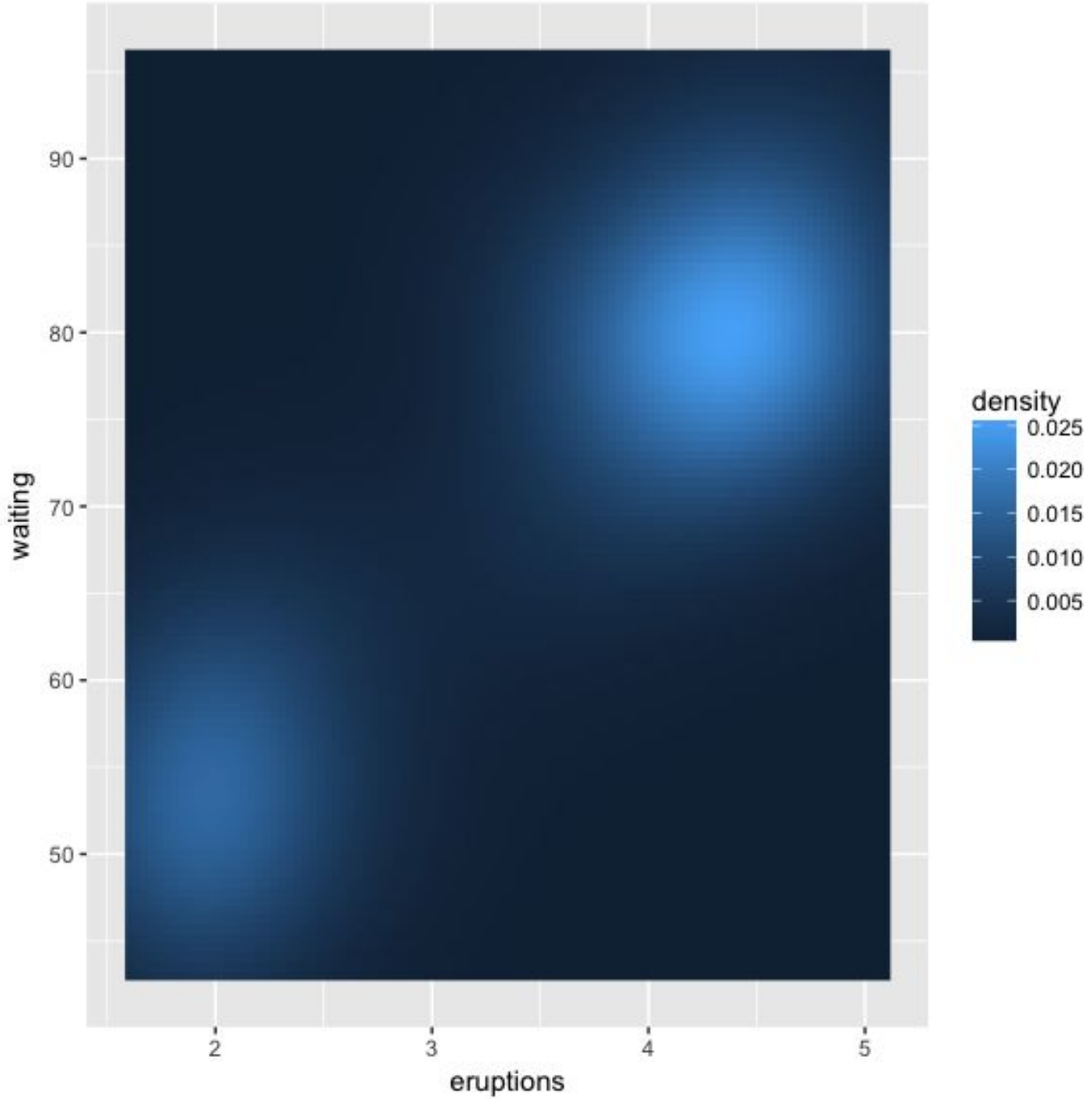
```
# The base plot

p <- ggplot(faithful, aes(x=eruptions, y=waiting))

p +  stat_density2d()

# can map the 'height' also, with ..level..

p + stat_density2d(aes(colour=..level..)) + geom_point()
```



Density Plot

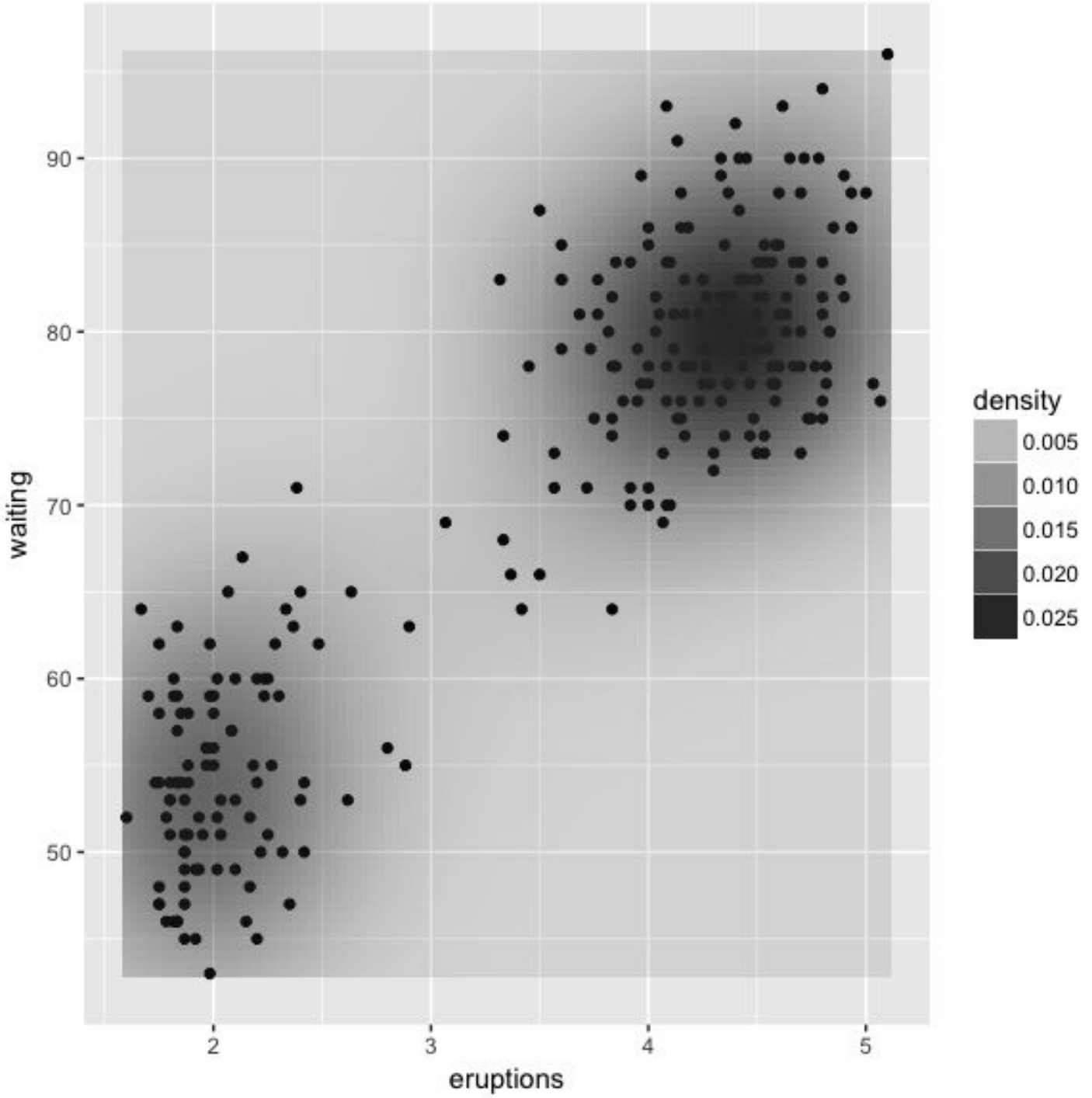
Map density estimate to fill color

p + stat_density2d(aes(fill=..density..), geom="raster", contour=FALSE)

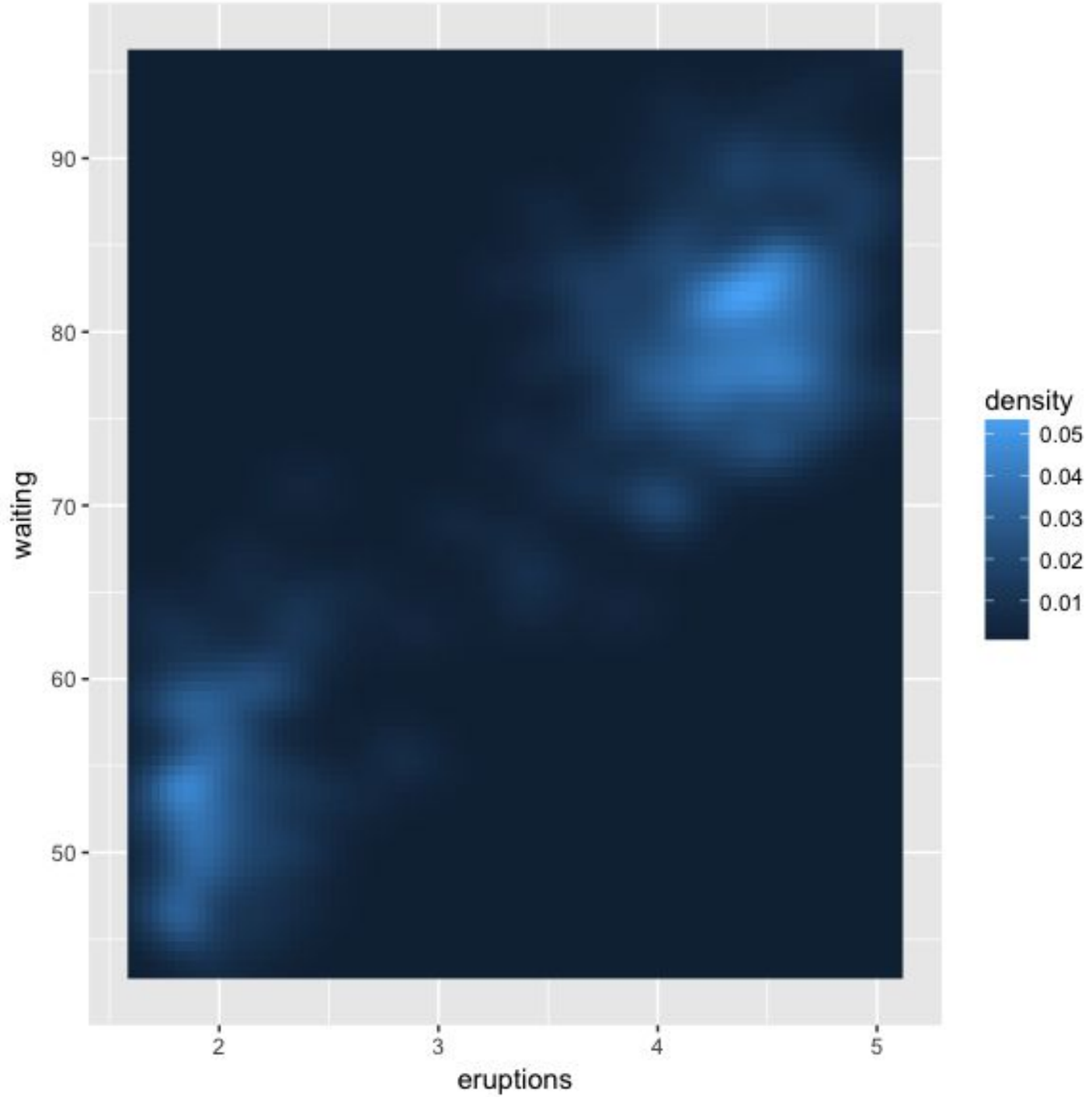


With points, and map density estimate to alpha

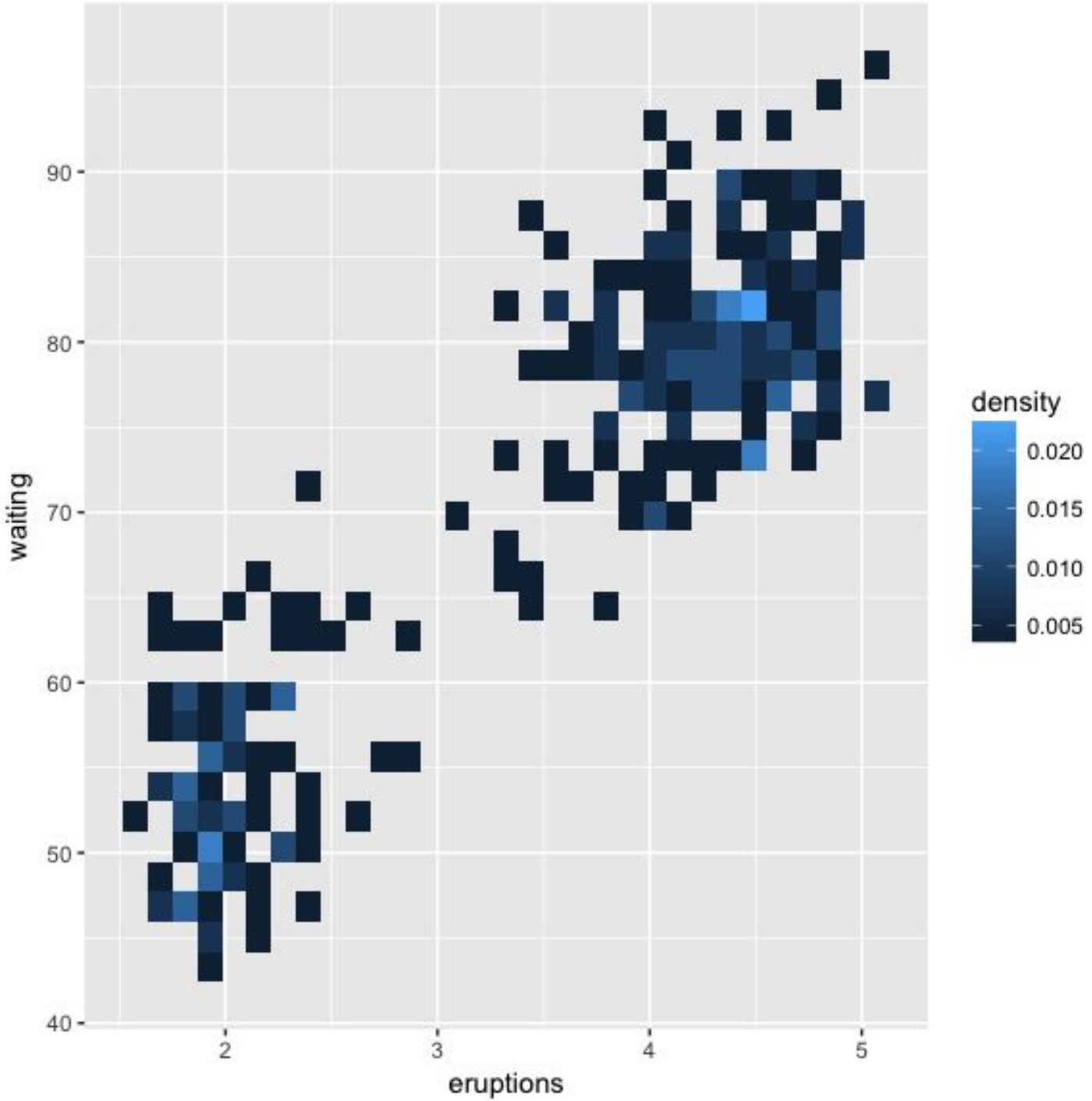
p + geom_point() +
stat_density2d(aes(alpha=..density..),
geom="tile", contour=FALSE)



the h argument allows you to control the fineness of the grid



p + stat_bin2d(aes(fill=..density..))



Shiny

Server

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

ui

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

R Visualization Example

```r
library(foreign)  # Allows us to read spss files!
library(corrplot)
library(ggplot2)

# Read in the hbat spss "hbat" dataset from the book by Hair, et. al.
hbat = read.spss("HBAT.sav", to.data.frame=T)
head(hbat)
```

```
> names(hbat) = c("id", "CustomerType", "IndustryType", "Size", "Region", "DistributionSystem", "
ProductQuality", "E-Commerce", "TechSupport", "ComplaintResolution", "Advertising", "ProductLine"
, "SalesforceImage", "CompetitivePricing", "Warranty", "NewProducts", "OrderingAndBilling", "Pric
eFlexibility", "DeliverySpeed", "CustomerSatisfaction", "Recommending", "FuturePurchase", "Purcha
sePercentage", "FutureRelationship")
> names(hbat)
```

```r
# Pull out just the numeric fields and place customer satisfaction
# at the front because it will be our "Y".  It makes it easier to
# interpret correlation matrices!
hbatNumeric = hbat[, c(20, 7:19)]
head(hbatNumeric)
plot(hbatNumeric)

# Compute the correlation matrix and visualize it
cor.hbat = cor(hbatNumeric)
cor.hbat
corrplot(cor.hbat, method="ellipse")

# Let's look at the distributions of some of the parameters
par(mfrow = c(2, 2))  # divide the plot area into a 2x2 matrix of plots
hist(hbat$ProductQuality, col="red")
hist(hbat$`E-Commerce`, col="blue")
hist(hbat$Advertising, col="green")
hist(hbat$Warranty, col="magenta")

par(mfrow = c(1, 1)) # Switch back to a single plot

```

```r
par(mfrow = c(1, 1)) # Switch back to a single plot

# We can get a boxplot of all the numerical fields
boxplot(hbatNumeric)

# Let's look at the distributions of a specific parameter based on one of the categorical parameters
boxplot(CustomerSatisfaction ~ CustomerType, data=hbat, col="darkgreen")

# A simple scatter plot of two fields.  The "pch" option specifies the plot character 16 = filled circle
plot(hbat$ComplaintResolution, hbat$CustomerSatisfaction, col="blue", pch=16)

# run a regression
fit = lm(CustomerSatisfaction ~ ComplaintResolution, data=hbat)
summary(fit)

# Now create a scatter plot and overlay the fit
plot(hbat$ComplaintResolution, hbat$CustomerSatisfaction, pch=16)
abline(fit, col="red", lw=4)

par(mfrow=c(2, 2))
plot(fit)
par(mfrow=c(1, 1))

```

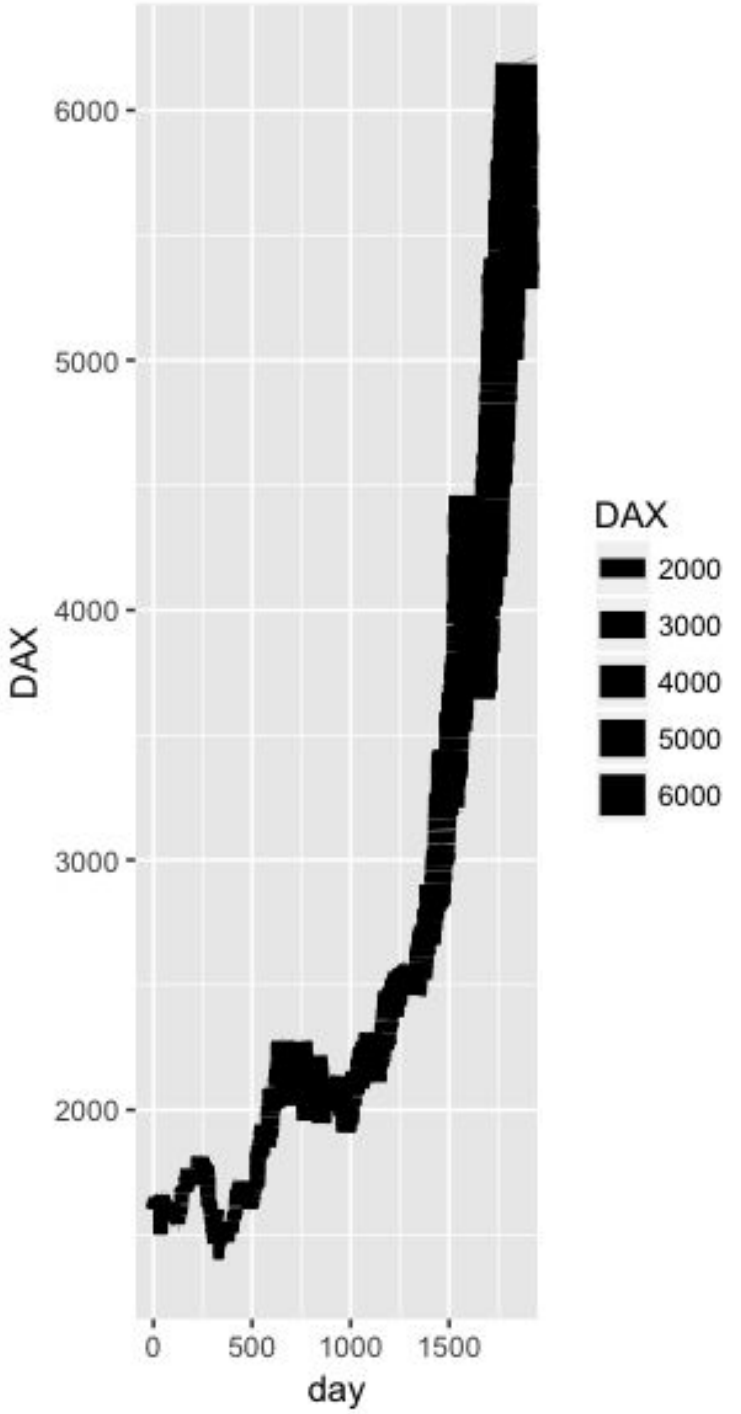
EU Stock Markets

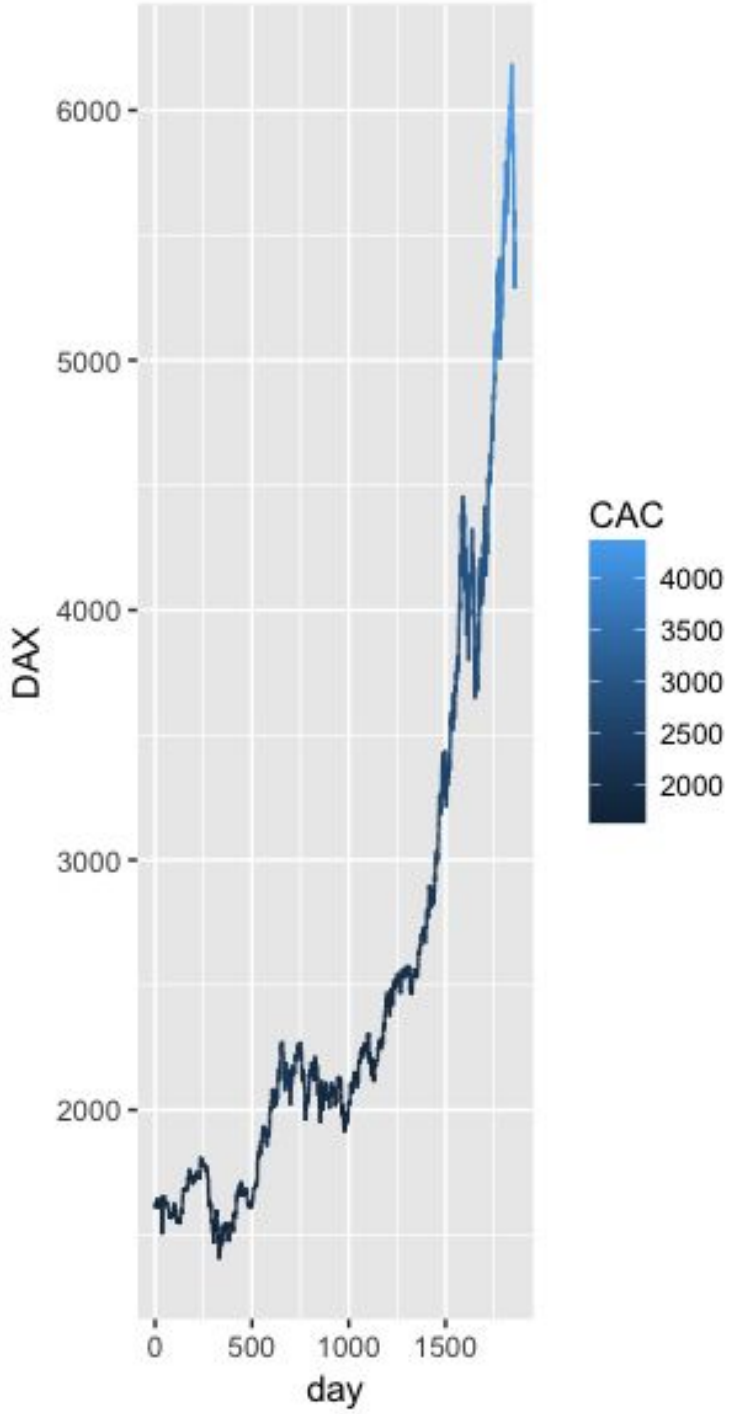
```
library(ggplot2)
head(EuStockMarkets)

# we want to add fields, so make a copy
ds = as.data.frame(EuStockMarkets)
ds$day = seq(nrow(ds))    # Add a sequential count of the records to indicate the "day"
# No, it is not totally correct, but in the absence of this information ...
library(reshape)

head(ds)

# Melt collapses columns into one field like the "pivot" feature in Tableau
# The id parameter here tells what fields to keep (i.e. not "melt")
EuStock = melt(ds, id=c("day"))
names(EuStock)[2] = "Index"
head(EuStock)
```

mapping the thickness of a line to a val

data from linegrpahs.r

```
ggplot(ds, aes(day, DAX)) +
geom_line(aes(size=DAX))

ggplot(ds, aes(day, DAX)) +
geom_line(aes(size=CAC))

ggplot(ds, aes(day, DAX)) +
geom_line(aes(colour=CAC))
```

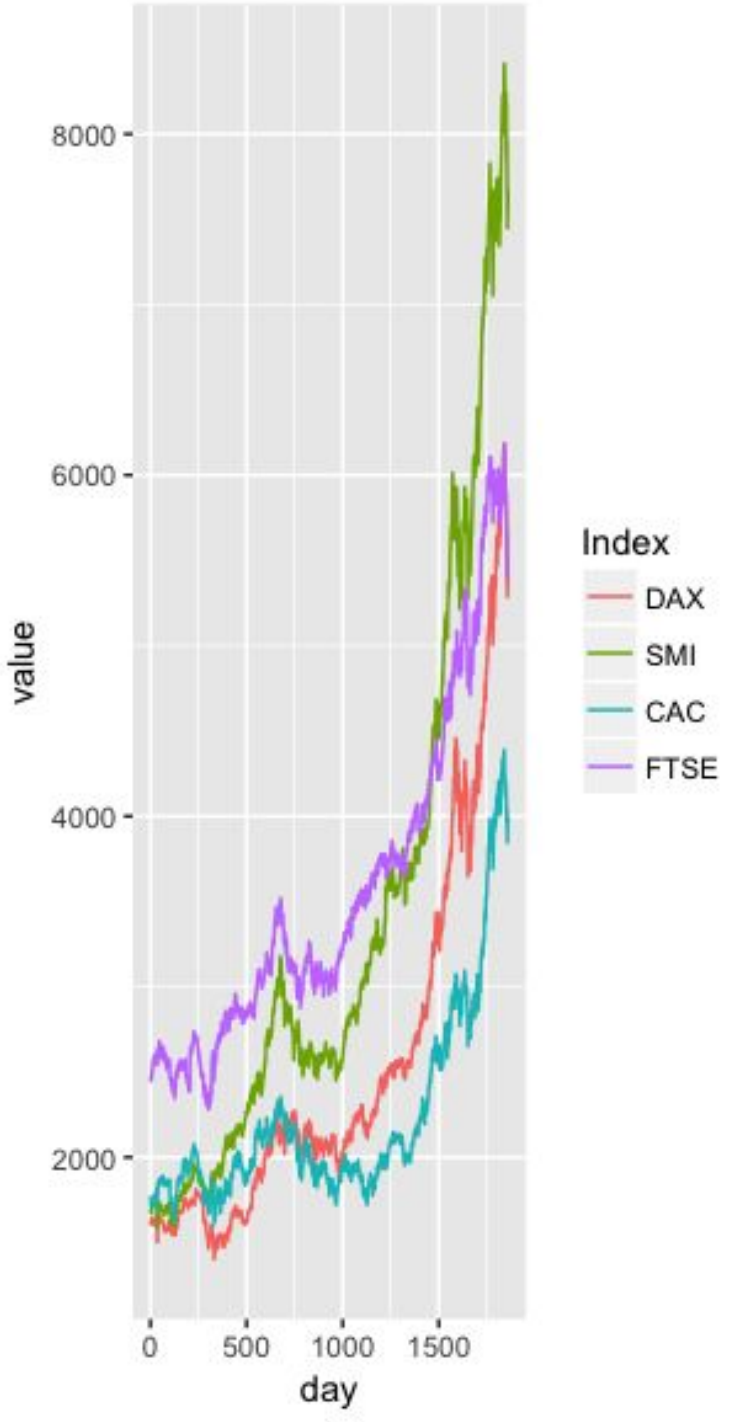


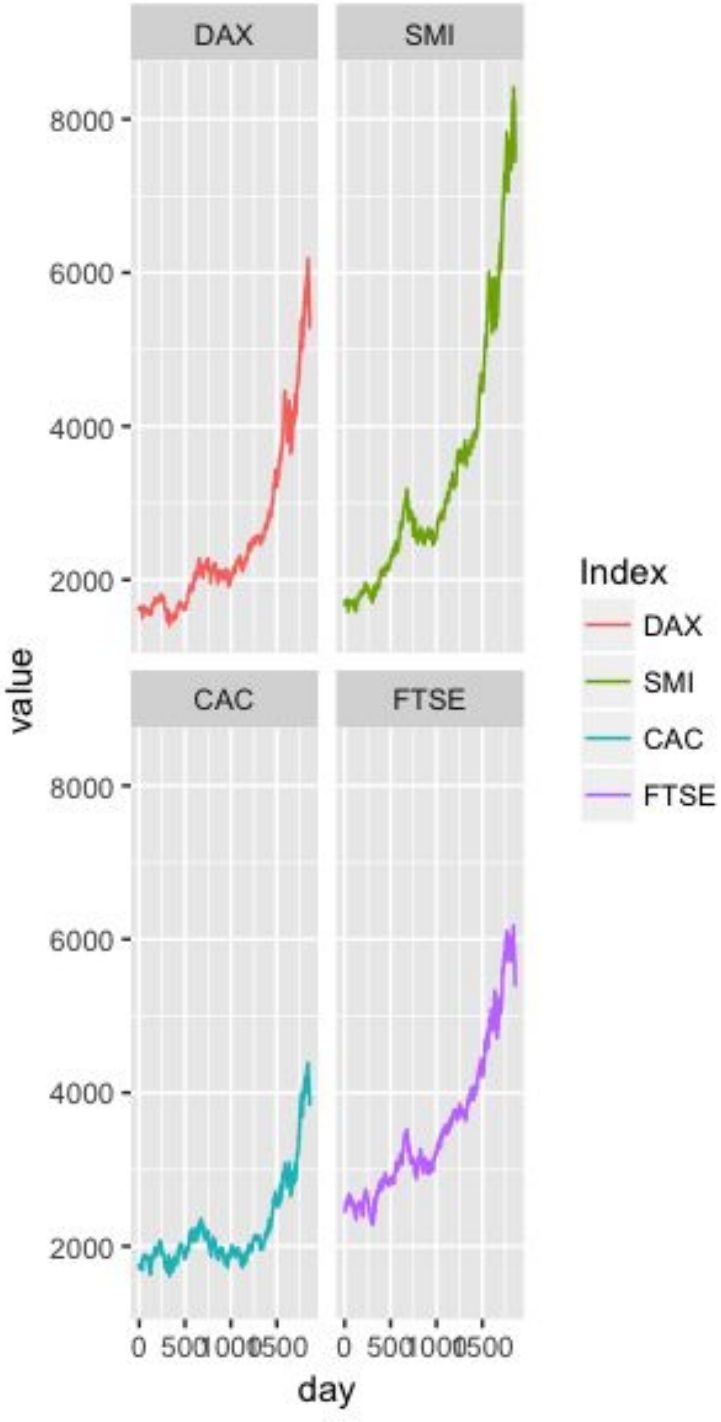


#smoothing

```
ggplot(EuStock, aes(day,value)) +
geom_line(aes(colour=Index))
```

```
ggplot(EuStock, aes(day,value)) +
geom_line(aes(colour=Index)) +
                    facet_wrap(~ Index)
```

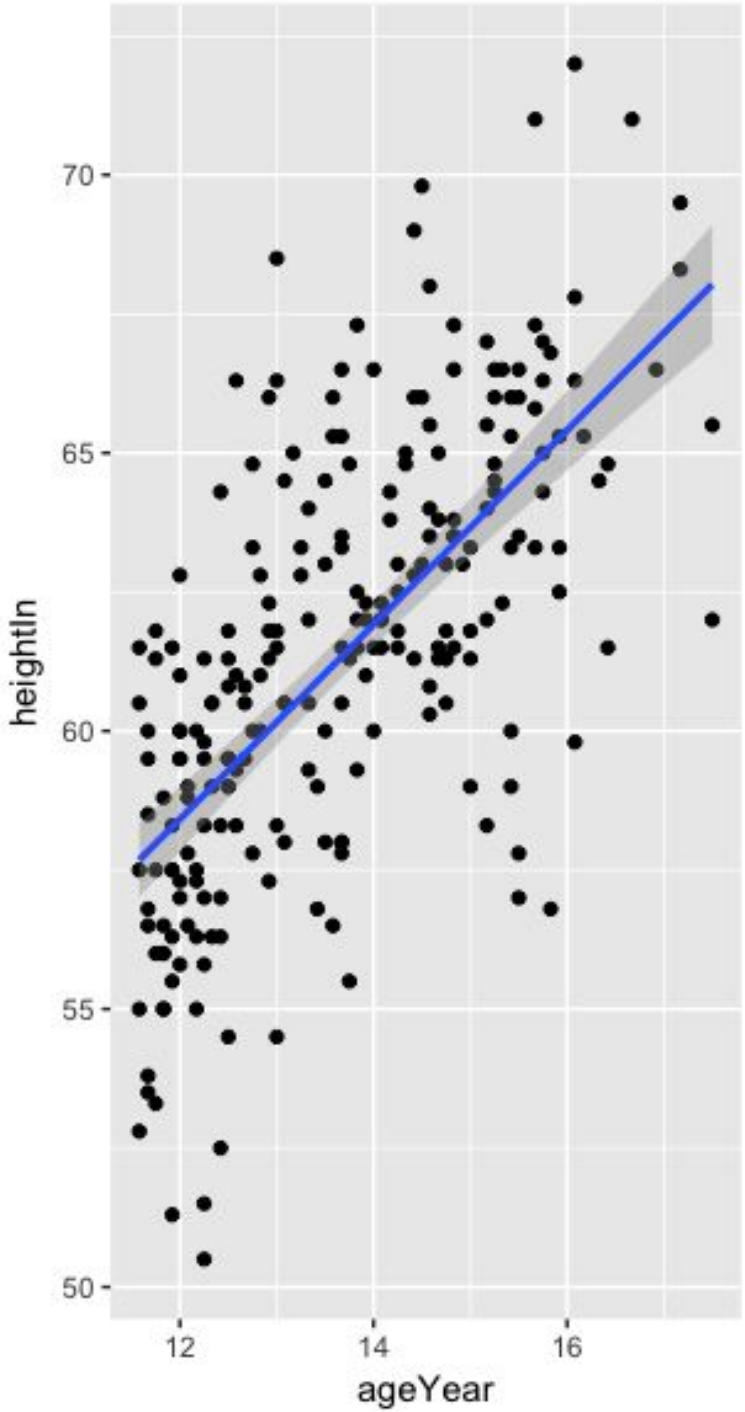




```
library(gcookbook) # for heightweight dataset

head(heightweight)

sp <- ggplot(heightweight, aes(x=ageYear, y=heightIn))
```

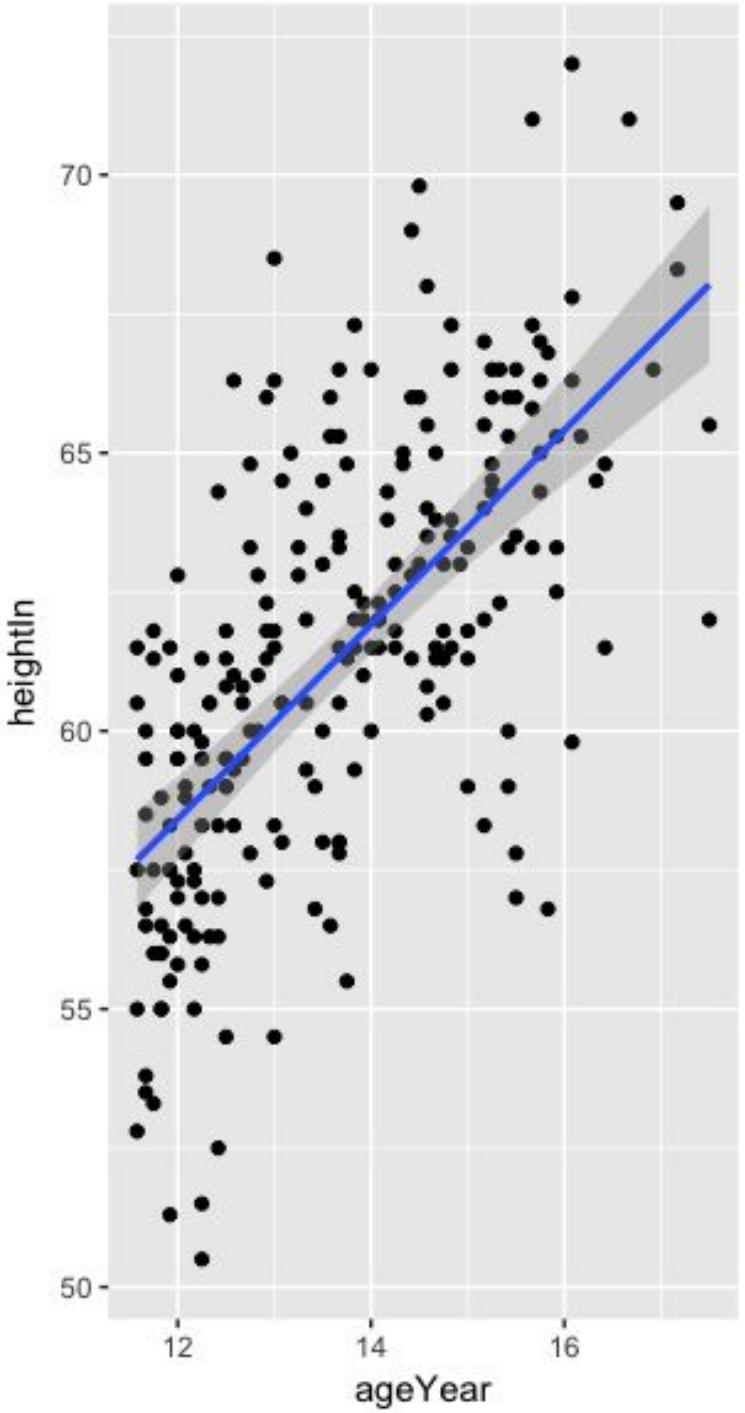
with a linear model (lm) smooth, 95% conf interval shaded

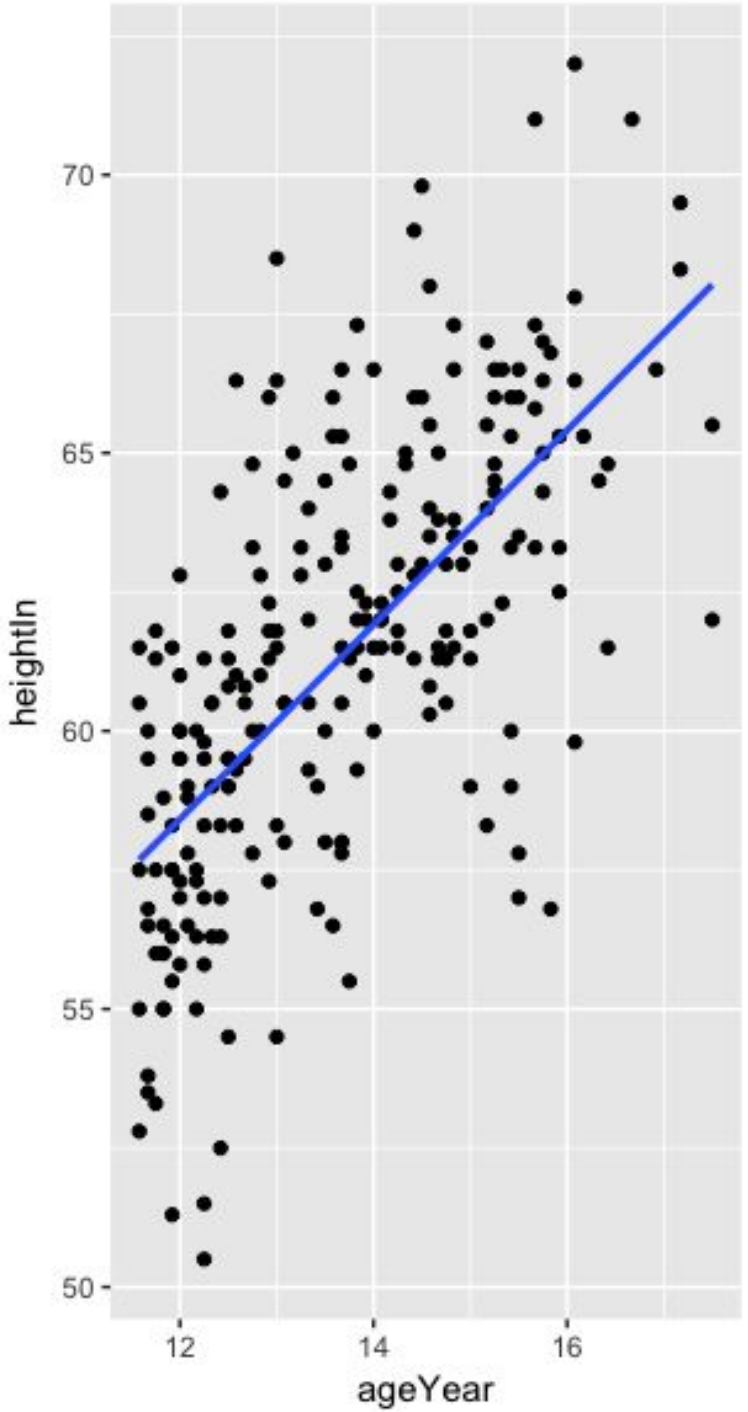
sp + geom_point() +
stat_smooth(method=lm)



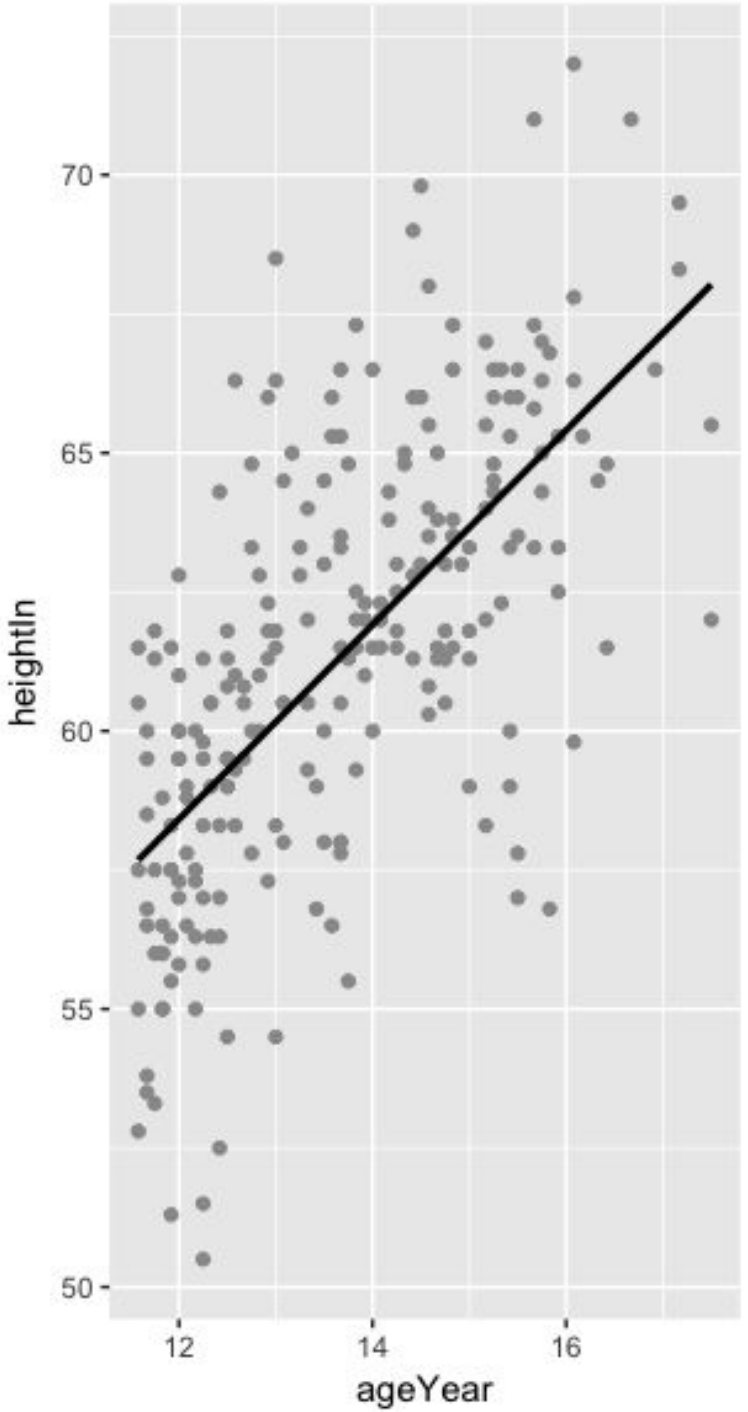
control the confident interval

```
sp + geom_point() +
stat_smooth(method=lm, level=0.99)

sp + geom_point() +
stat_smooth(method=lm, se=FALSE) # nc
```





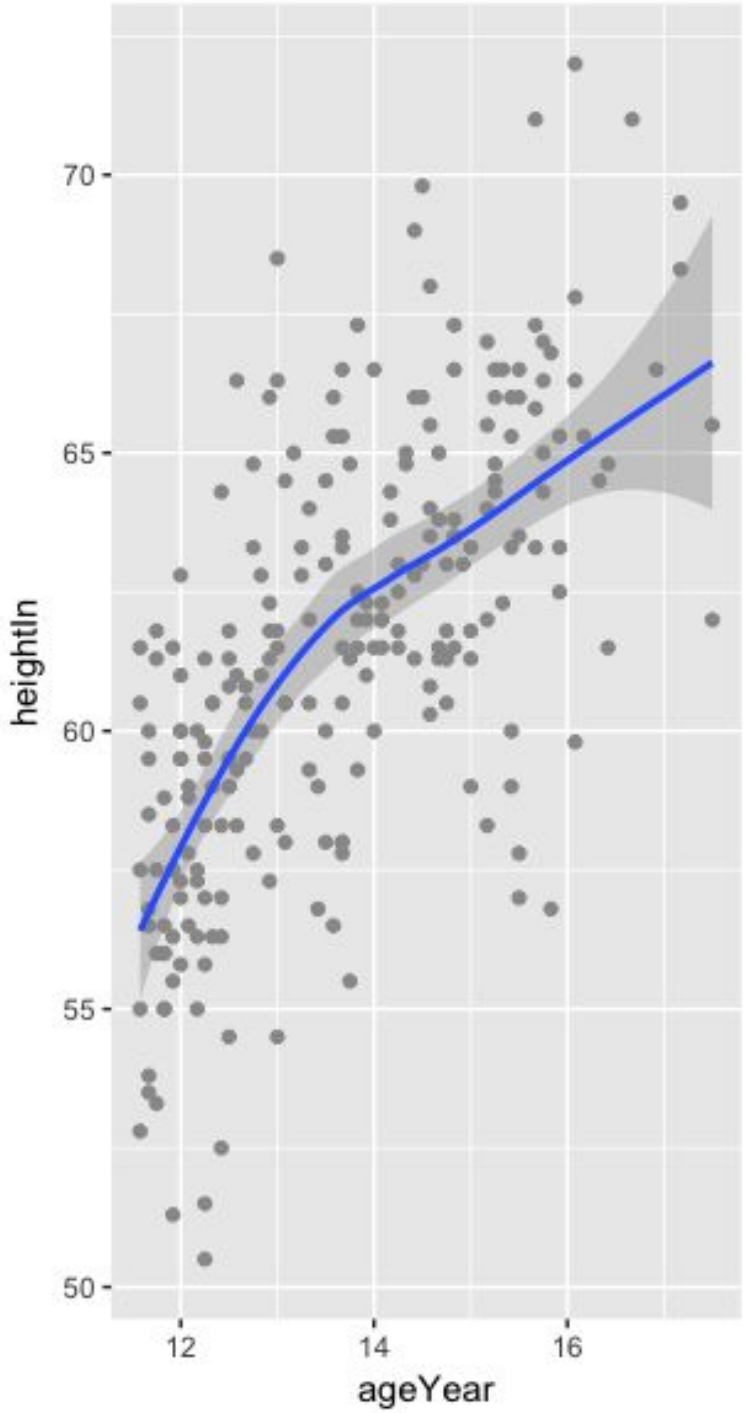
```
# emphasize the fit line with color
sp + geom_point(colour="grey60") +
    stat_smooth(method=lm, se=FALSE,
colour="black")
```



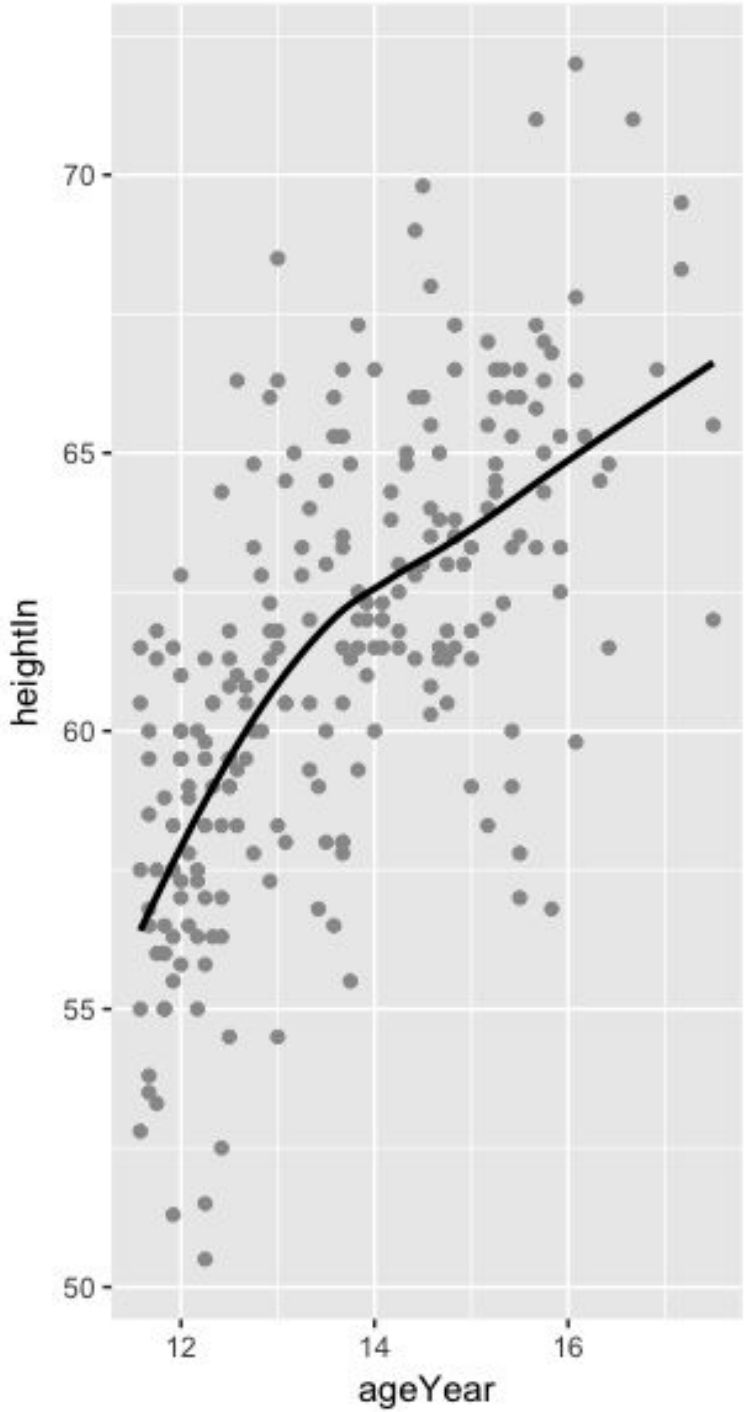
non-linear smoothing (LOESS)

sp + geom_point(colour="grey60") +
stat_smooth() # same as below

sp + geom_point(colour="grey60") +
stat_smooth(method=loess)



```
# no confidence region, black line on grey
sp + geom_point(colour="grey60") +
    stat_smooth(colour="black", se=FALSE) # same as below
```



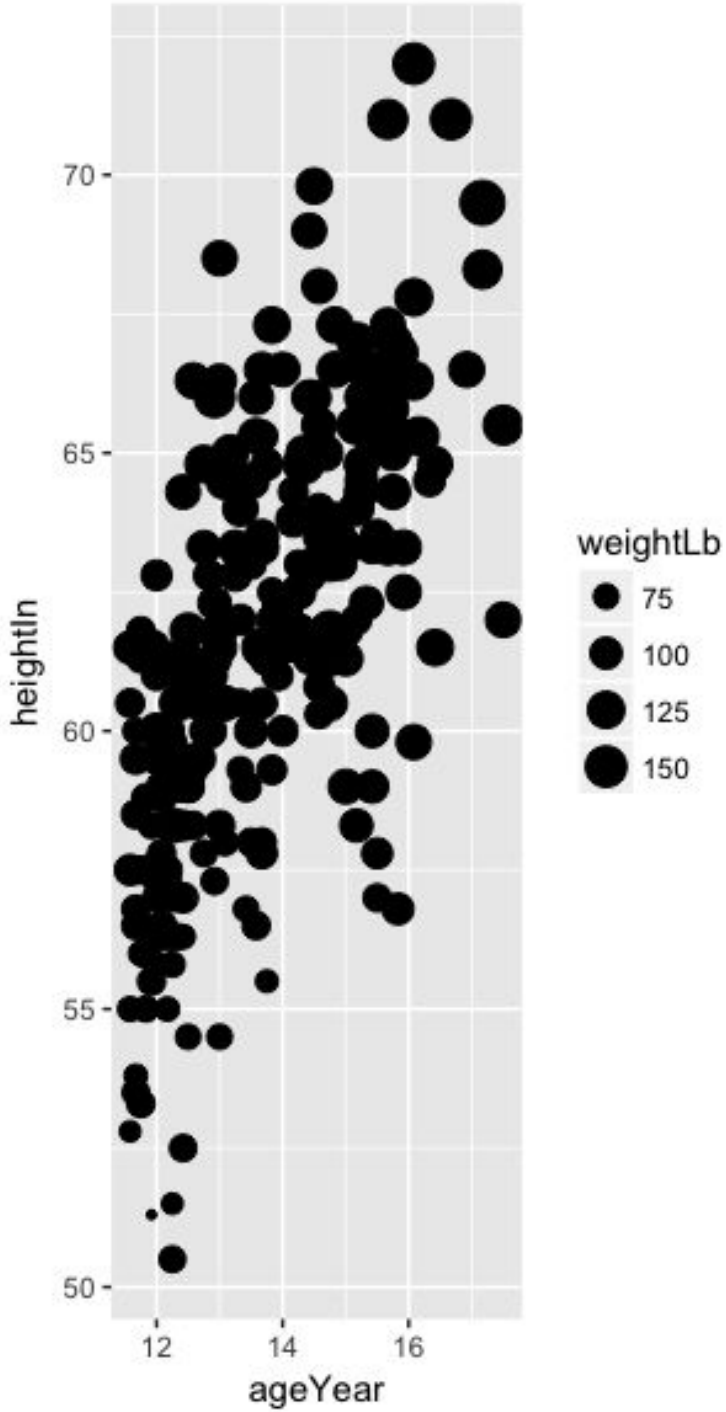
map size to points on scatterplot

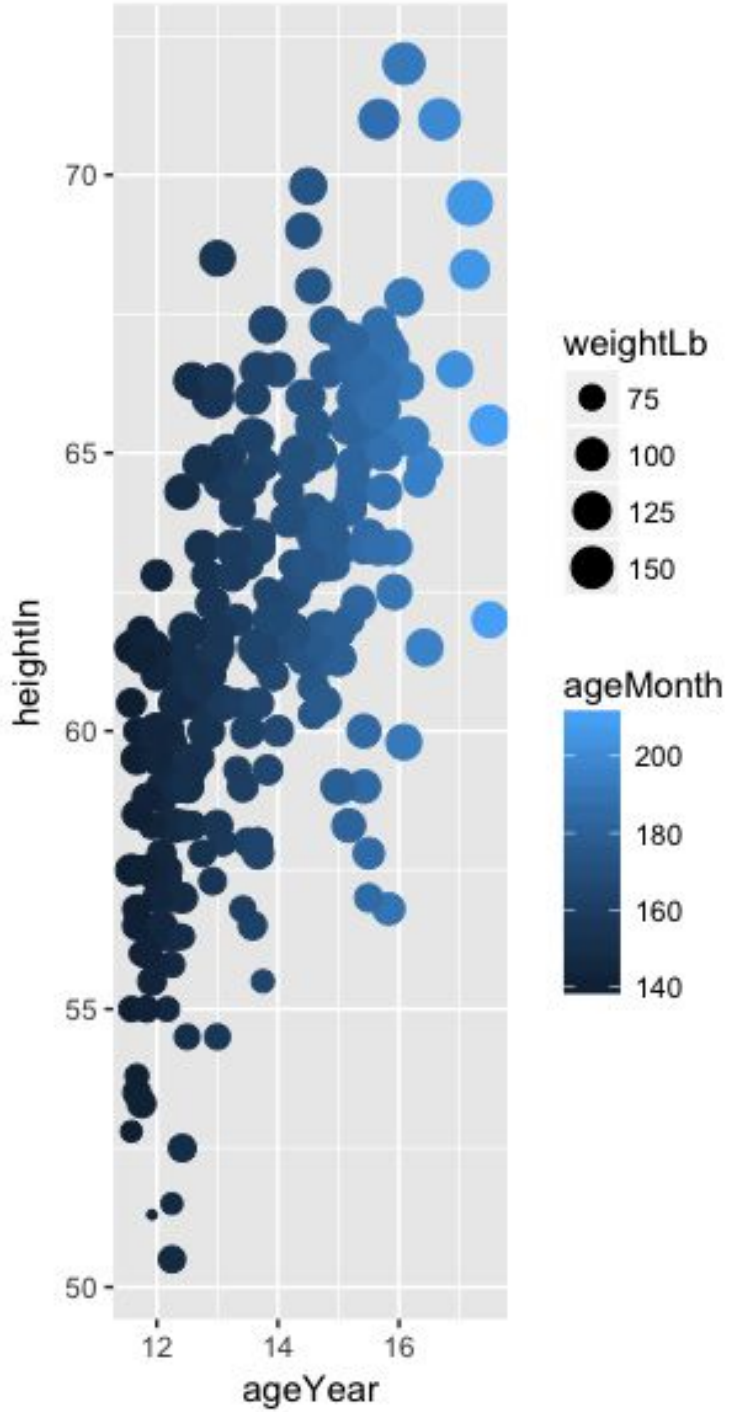
we have age and height, so let's do wei
to size of point

sp + geom_point(aes(size=weightLb))

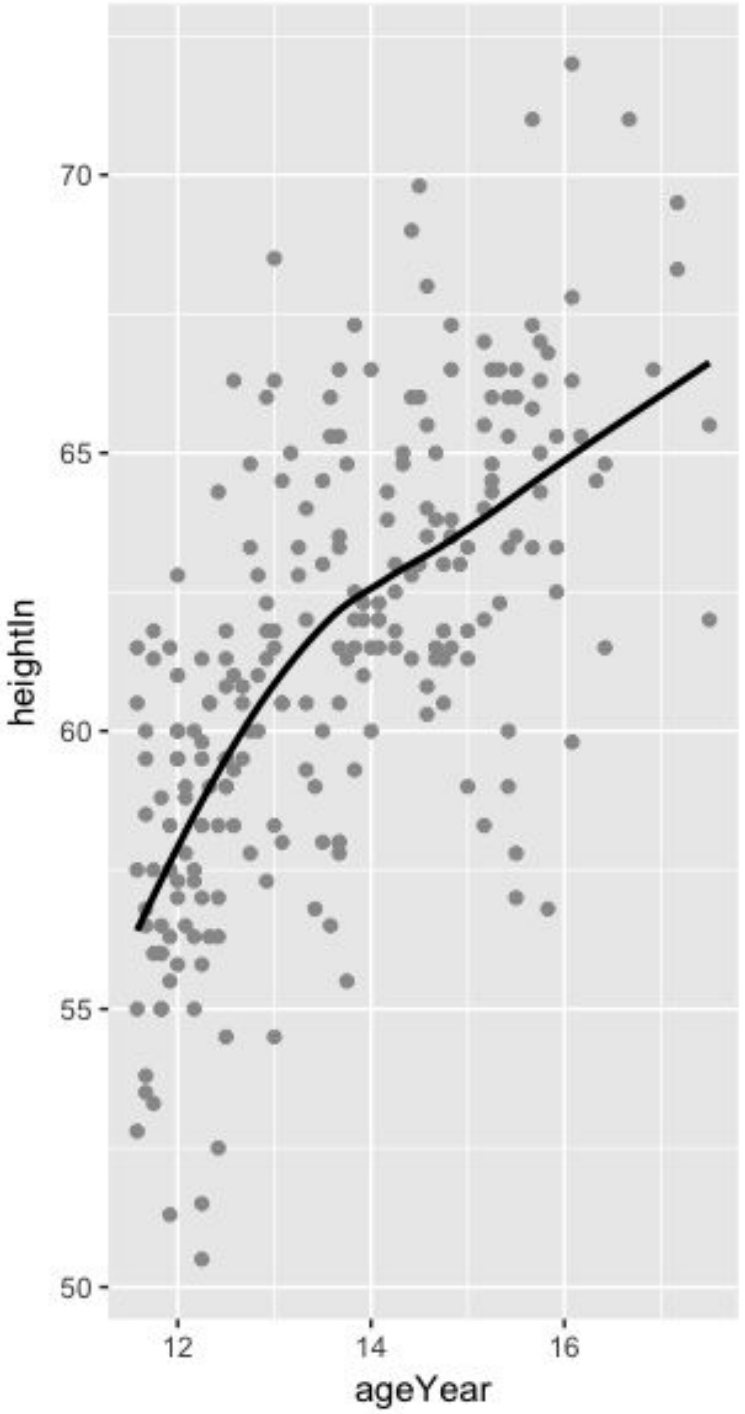
or color to age (not much point)

sp + geom_point(aes(size=weightLb,
colour=ageMonth))



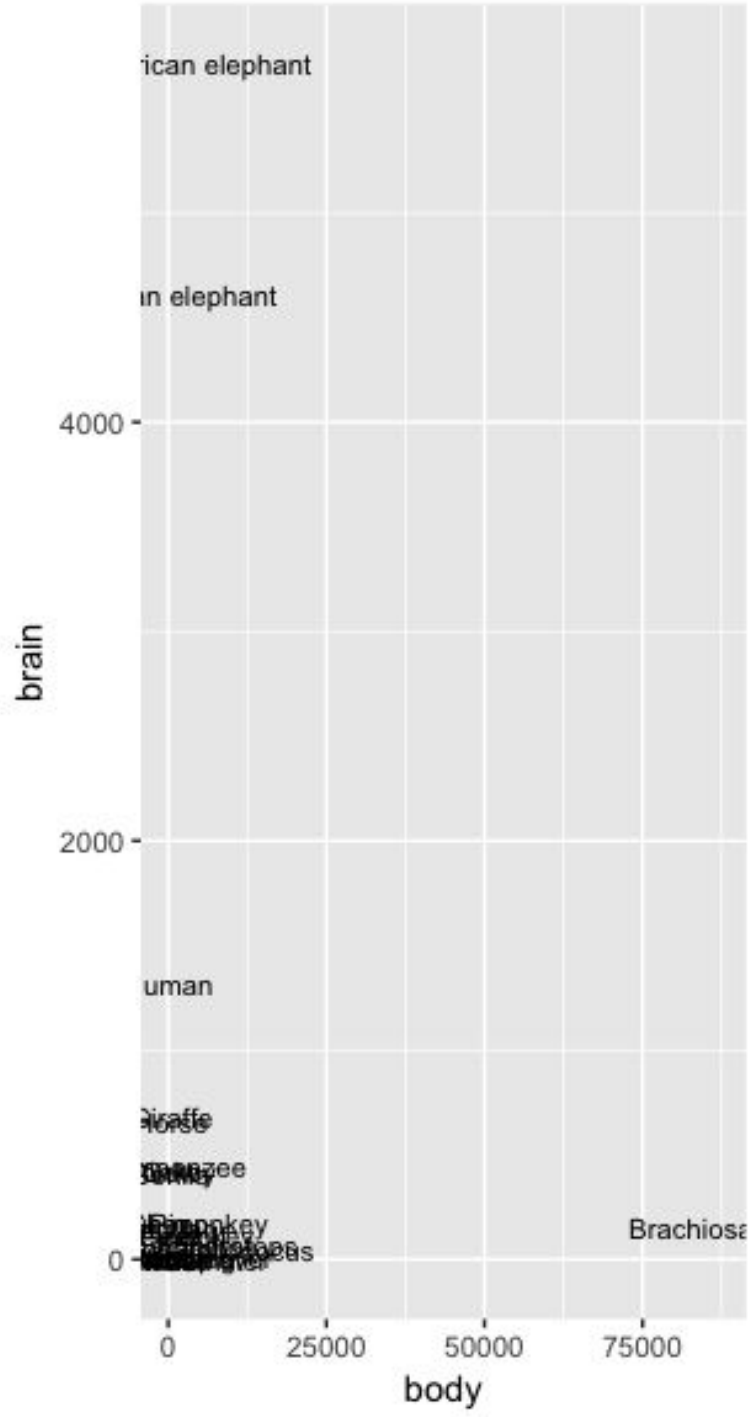


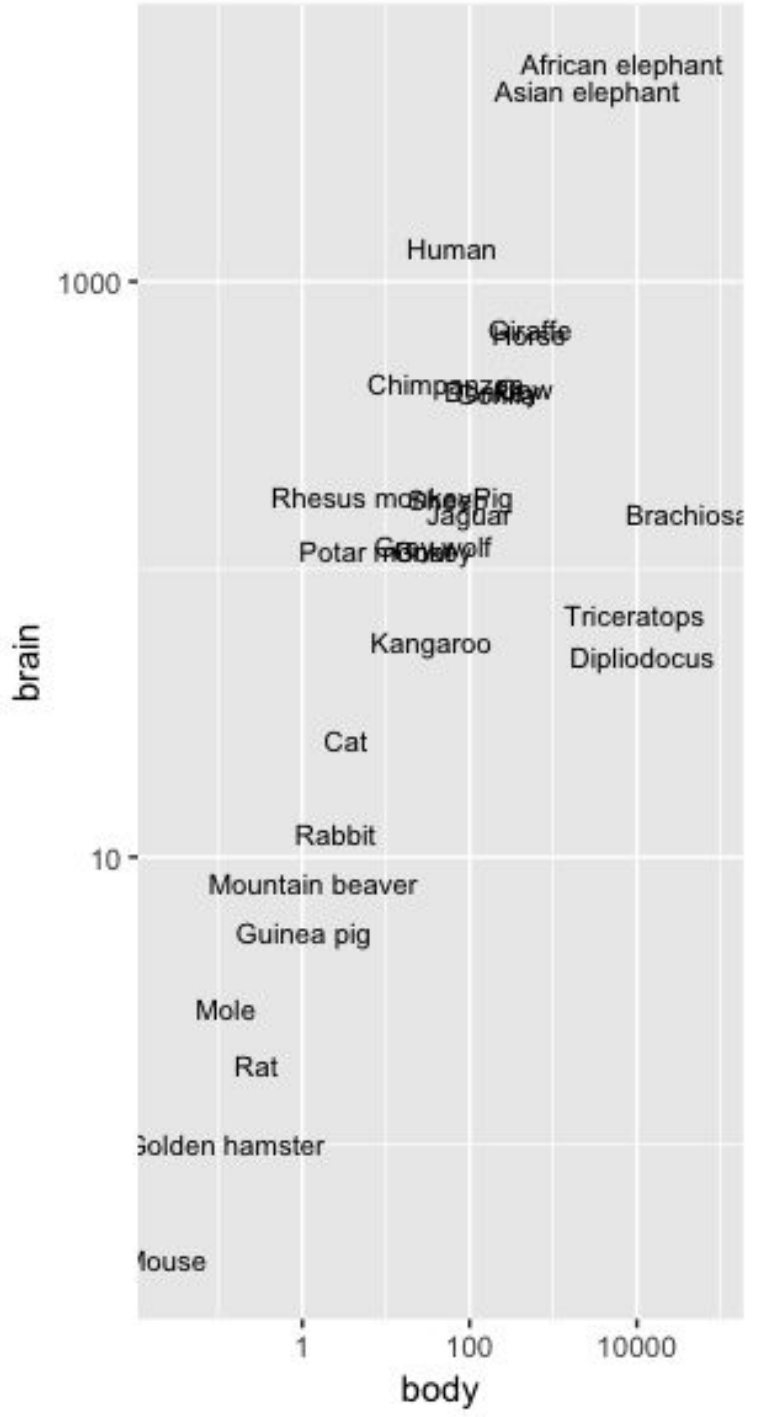
```
# get the subset of data where the subje
female

femHW <- heightweight[heightweight$s
"f" ,]

spfem <- ggplot(heightweight,
aes(x=ageYear, y=heightIn))

spfem + geom_point(colour="grey60") +

      stat_smooth(colour="black", se=FAL
# same as below
```



logscale (look in R Graphics Cookbook for more detail)

```
library(MASS)

p <- ggplot(Animals, aes(x=body, y=brain,
label=rownames(Animals))) +

  geom_text(size=3)

p

p + scale_x_log10() + scale_y_log10()
```

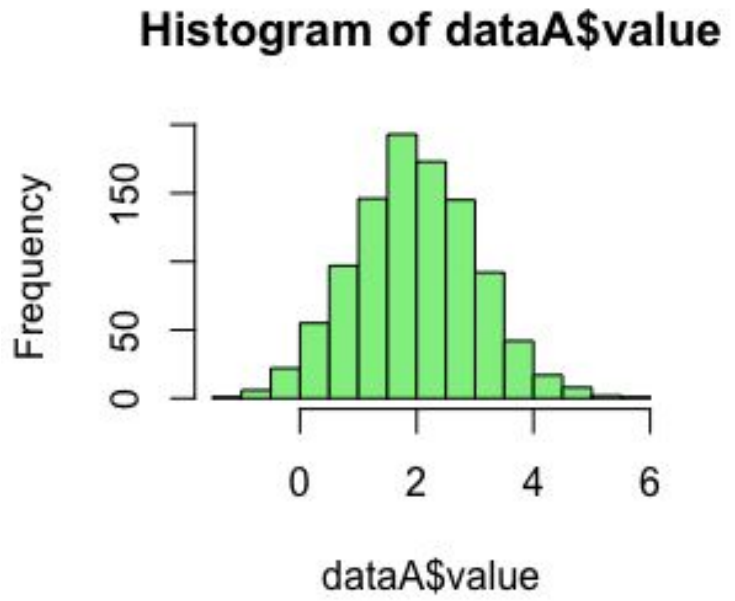


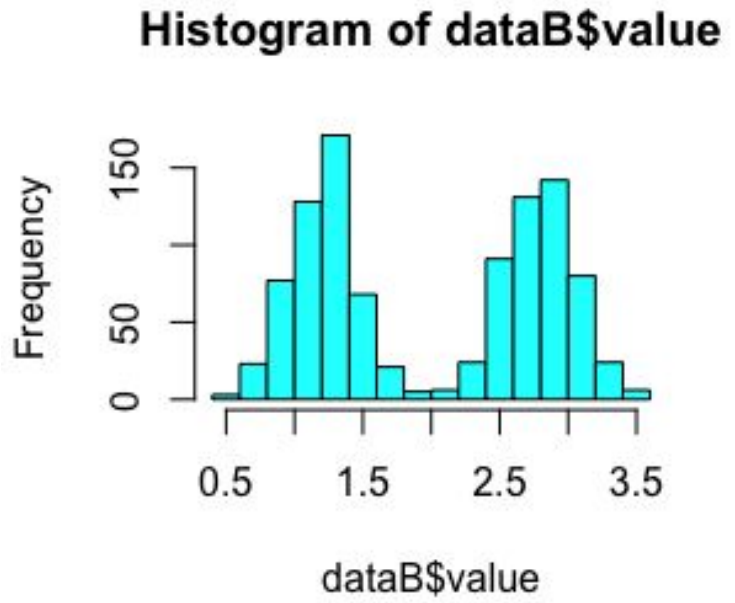


Gaussian Jitter

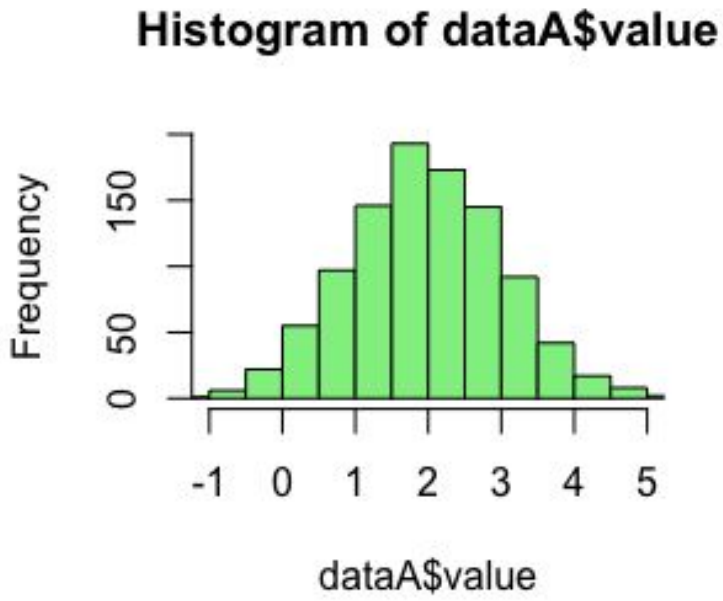
```
library(ggplot2)

# Comparing a normally distrbuted set with a bimodal one
n.each = 1000
ID = 1:1000
dataA = rnorm(n.each, 2, 1)
selector = round(runif(n.each, 0, 1))
dataB = (1.0 - selector) * rnorm(n.each, 1.2, .25) + selector * rnorm(n.each, 2.8, .25)
data = c(dataA, dataB)
values = data
dataGroup = rep(c("A", "B"), each=n.each)
numericGroup = rep(c(1, 2), each=n.each)
ds = data.frame(value=data, group=dataGroup)

head(ds)

par(bty="n")
par(mfrow=c(2, 1))
```

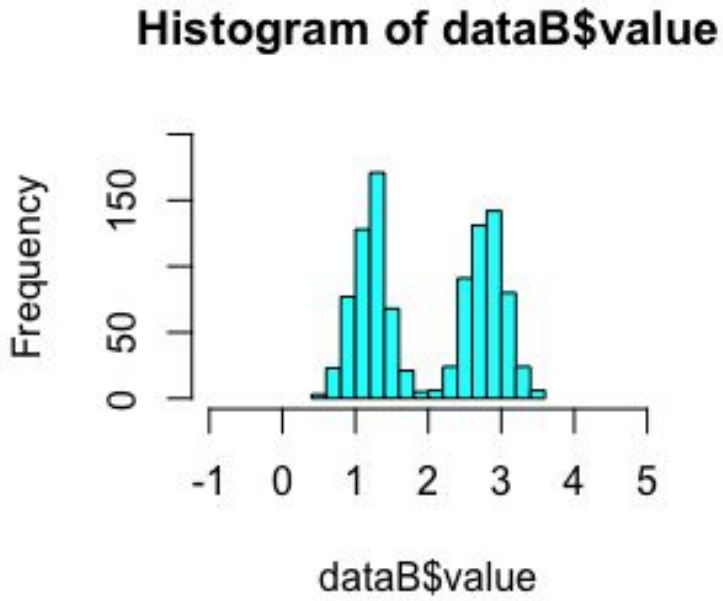
```
dataA = subset(ds, group=="A")
dataB = subset(ds, group=="B")
hist(dataA$value, col="lightgreen")
hist(dataB$value, col="cyan")
```



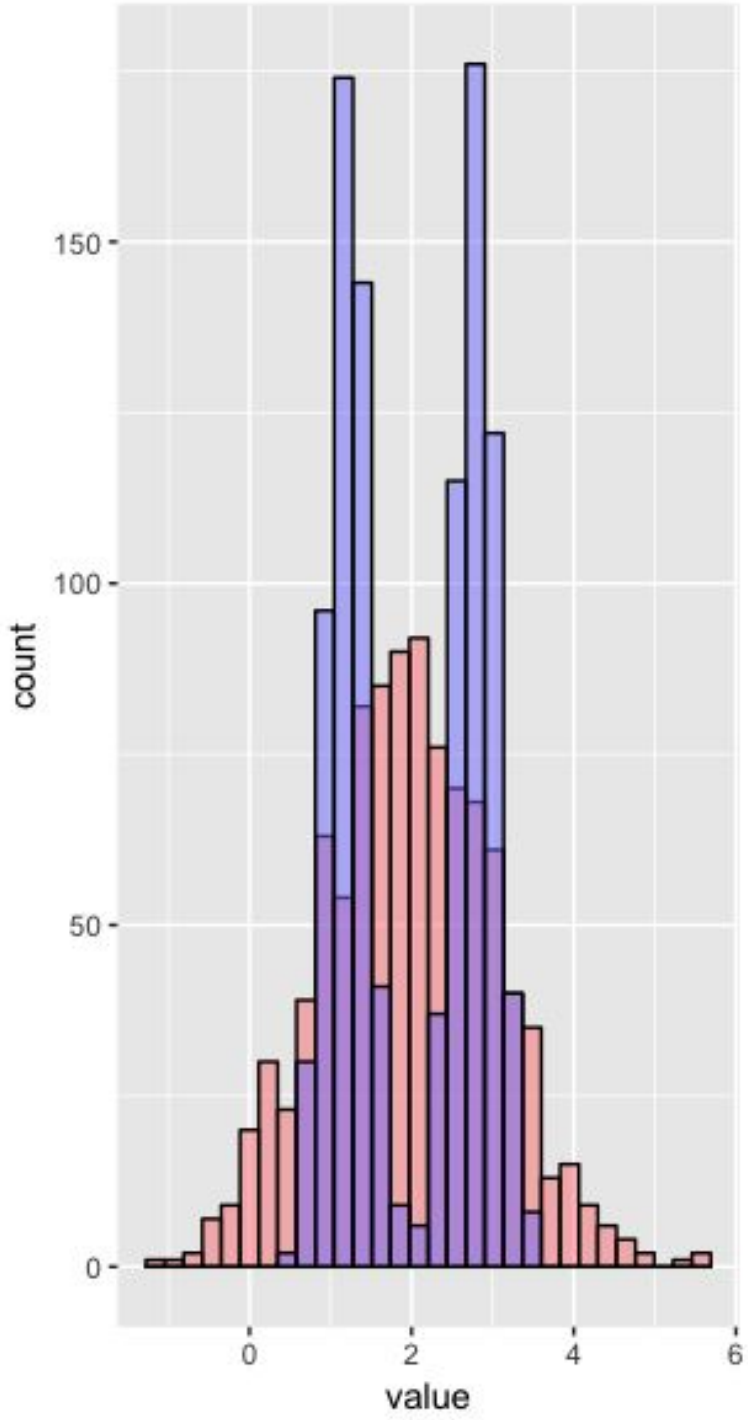


```
hist(dataA$value, col="lightgreen", xlim=c(-1,
5), ylim=c(0, 200))

hist(dataB$value, col="cyan", xlim=c(-1, 5),
ylim=c(0, 200))
```

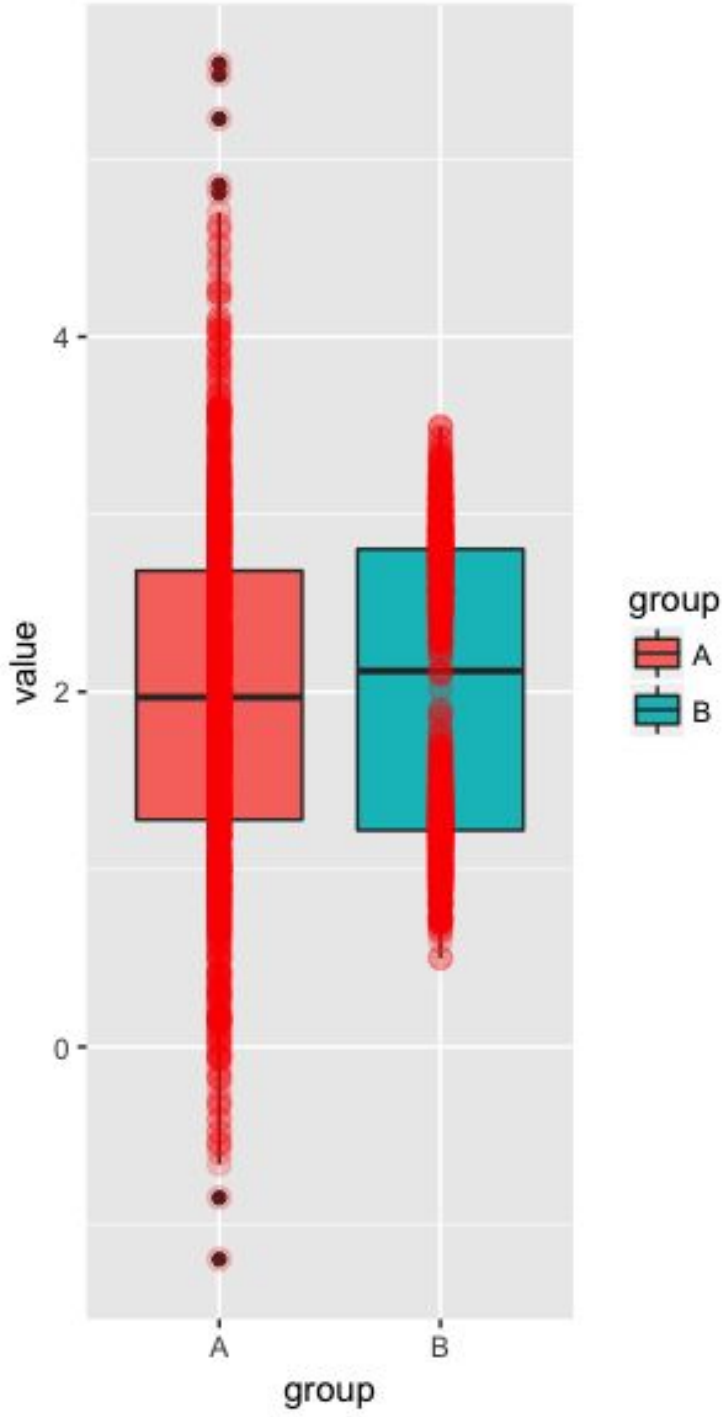




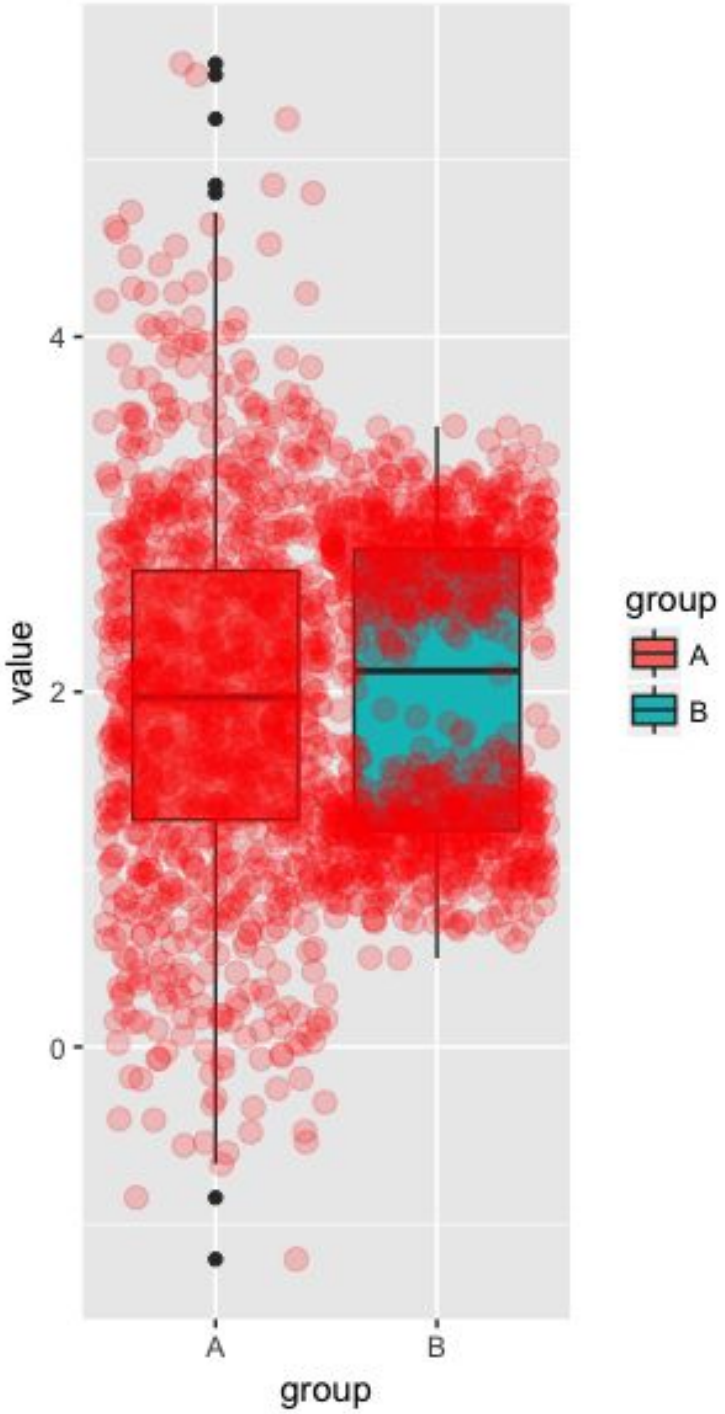
```
ggplot(ds, aes(x=value)) +
geom_histogram(data=subset(ds,
group=="A"), color="black", fill=rgb(1, .5, .5,
.5)) +

geom_histogram(data=subset(ds,
group=="B"), color="black", fill=rgb(.5, .5, 1,
.5))
```



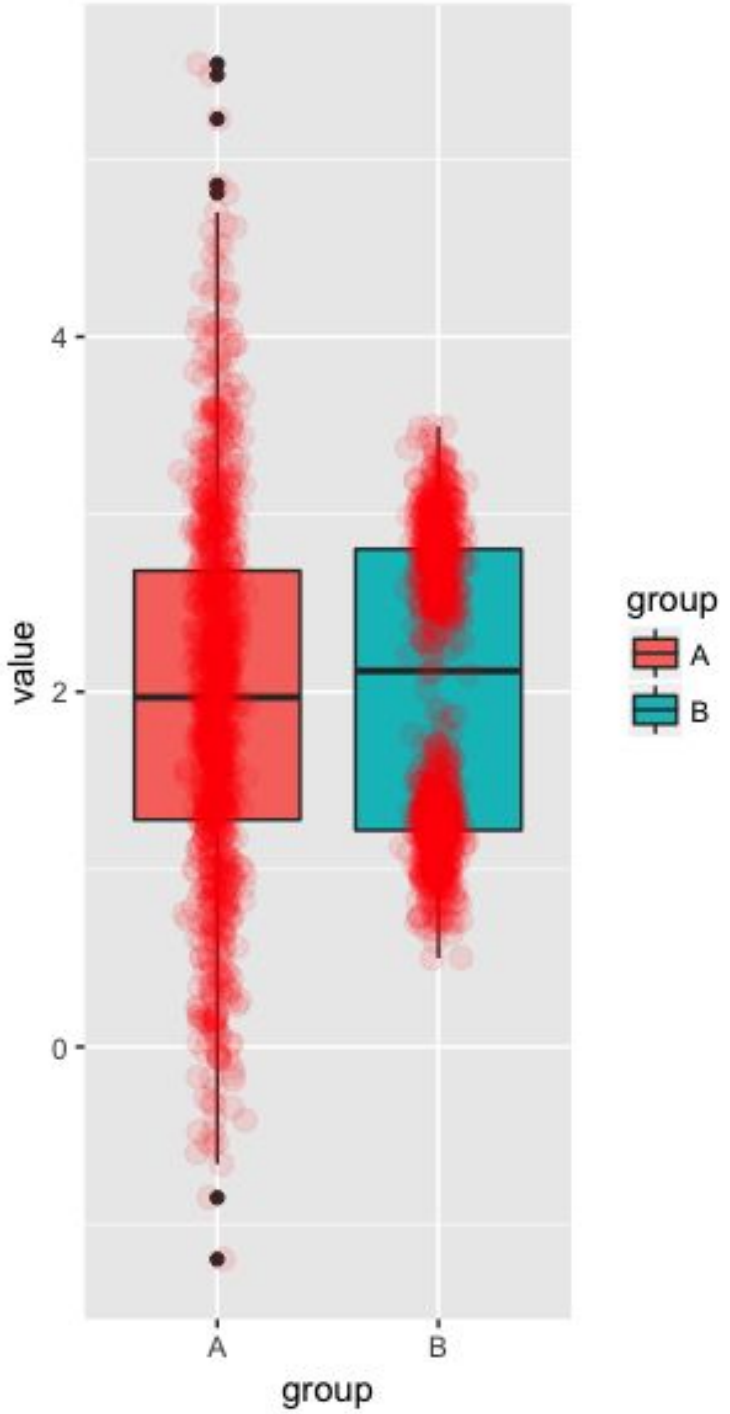
```
ggplot(ds, aes(x=group, y=value)) +
geom_boxplot(aes(fill=group)) +
                    geom_point(color="red",
alpha=.2, size=3)
```



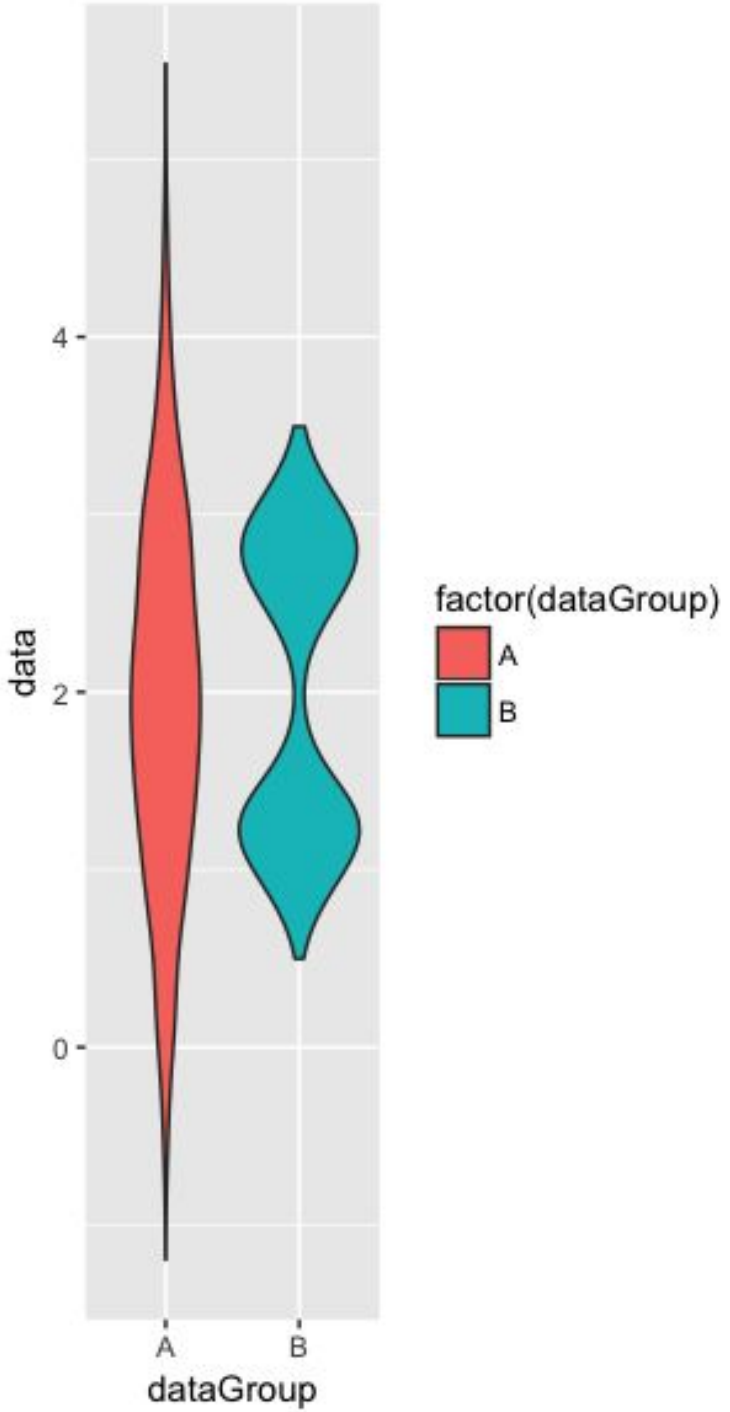
```
ggplot(ds, aes(x=group, y=value)) +
geom_boxplot(aes(fill=group)) +

  geom_jitter(color="red", alpha=.2, size=3,
width=.5)
```



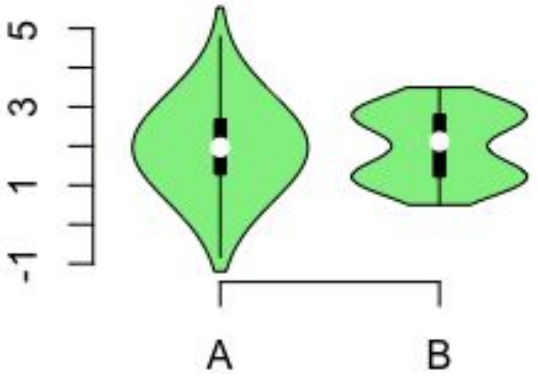
```
# Manually jittering with a normal
distribution

ggplot(ds, aes(x=group, y=value)) +
geom_boxplot(aes(fill=group)) +

  geom_point(aes(x=as.numeric(group) +
rnorm(n.each, 0, .05), y=value), color="red",
alpha=.1, size=3)
```



qplot(dataGroup, data, geom="violin",
fill=factor(dataGroup))



```
library(vioplot)

dataA = subset(ds, group=="A")

dataB = subset(ds, group=="B")

vioplot(dataA$value, dataB$value,
names=c("A", "B"), col=c("lightgreen",
"cyan"))
```



```
#Import data

df=read.csv("PC.csv", header=T)



#Check the first six rows

head(df)


#Install package

library(plotly)


#See max and min value of average BMI

max(df$Avg..BMI)

min(df$Avg..BMI)


#Create Dummy vairables for Gender

df$GenderDummy=ifelse(df$Gender=="Female",1,0)

df$GenderDummy=ifelse(df$Gender=="Male",1,0)

head(df)

df$GenderDummy


#Transform Data type



#Create Parallel Coordinate

p<-df%>%

 plot_ly(type='parcoords', line=list(color=~Year),

        dimensions=list(

          list(range=c(0), label="Gender", values=~df$GenderDummy),

          list(range=c(1975,2014), label="Year", values=~df$Year),

          list(range=c(4,23), label="Average BMI", values=~df$Avg..BMI)

        )

        )


Sys.setenv("plotly_username"="MewTwoZhuang")

Sys.setenv("plotly_api_key"="fYBdfCz5NiURwirK6uZM")


chart_link=api_create(p, filename = "MewTwoZhuang")

chart_link
```