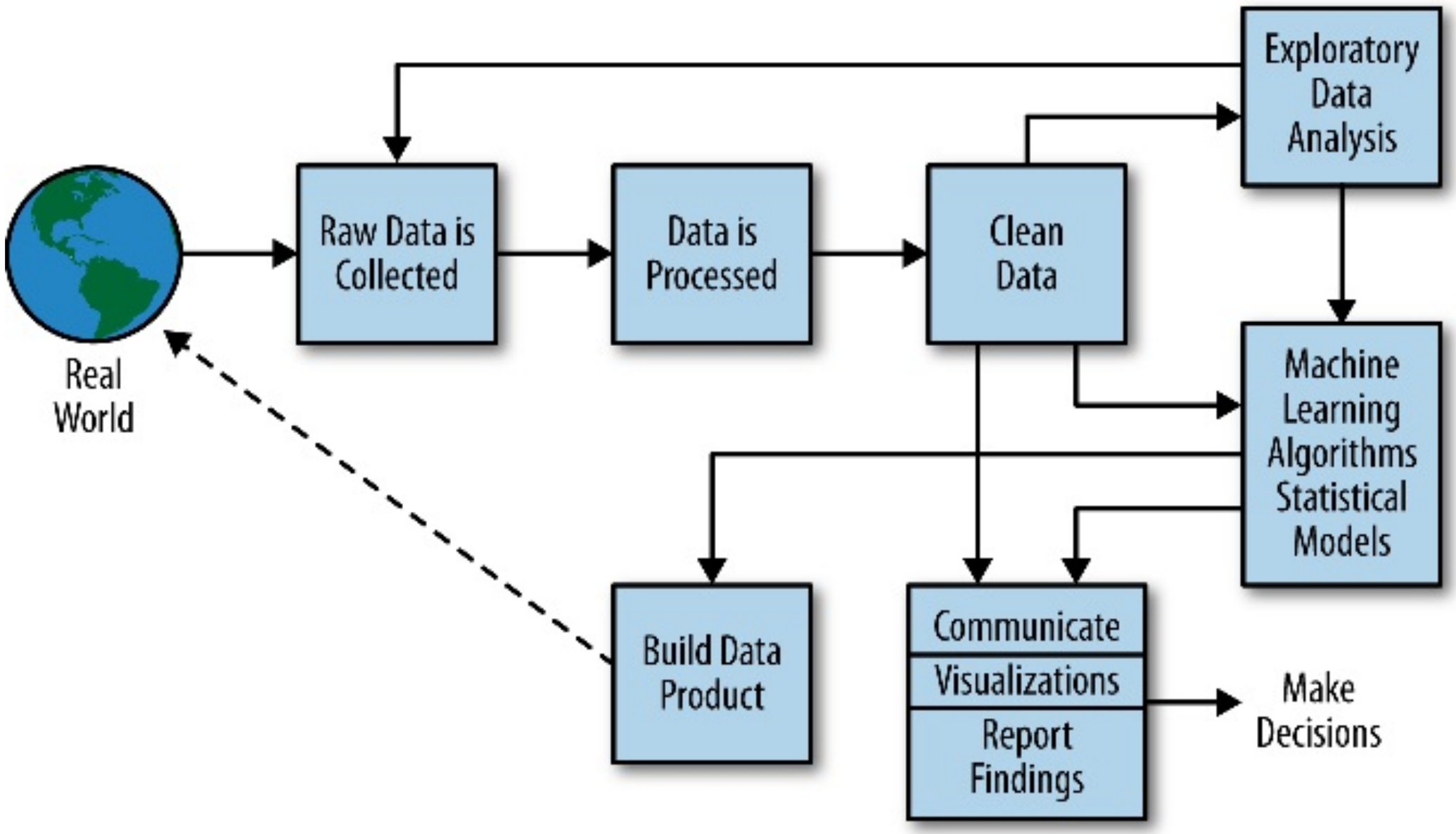
Data Science Certification Module 2

Veerasak Kritsanapraphan

Data Science Life Cycle

Data Science Process

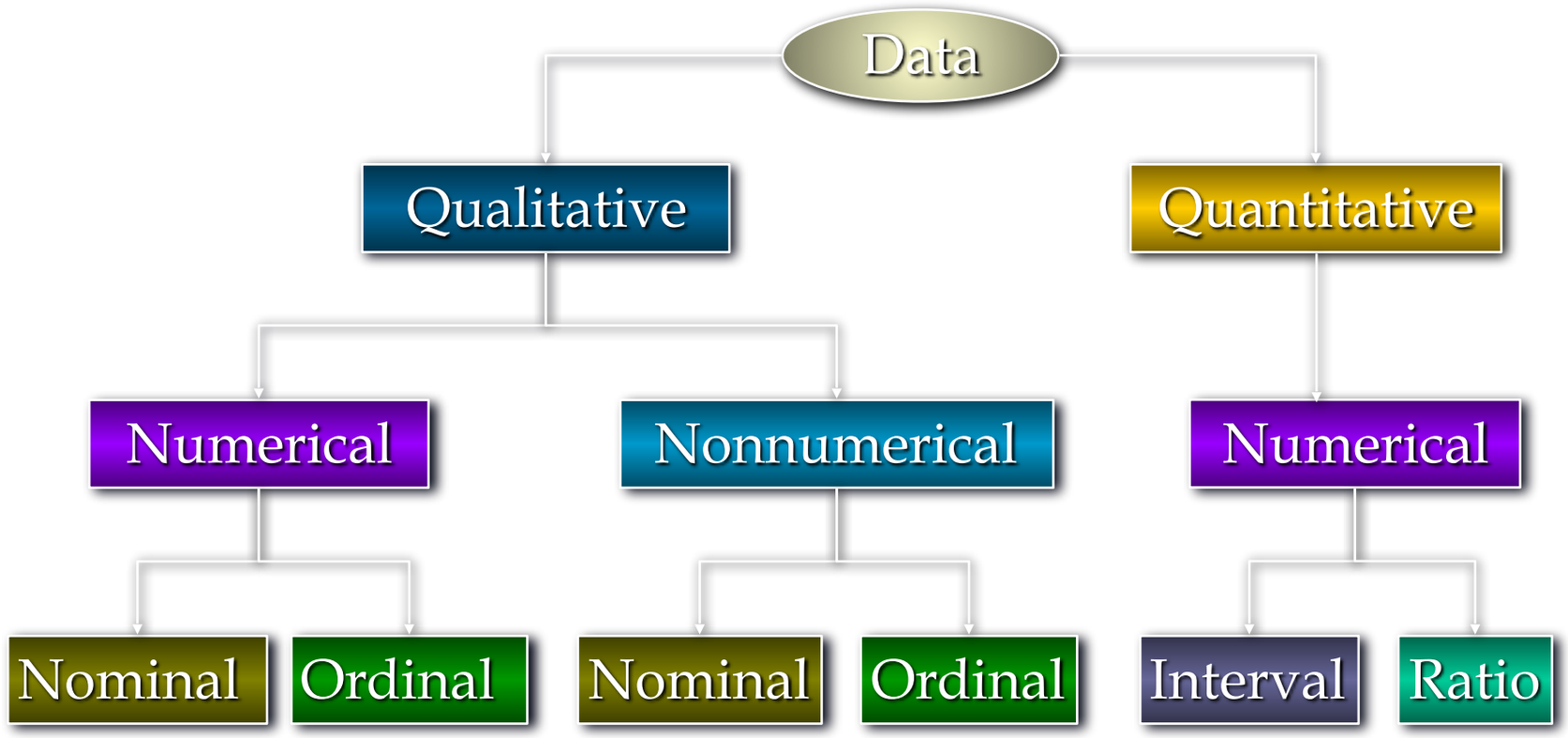


Data and Getting Data

Data

- Before one can present and interpret information, there has to be a process of gathering and sorting data.
- Definition of data is **"facts or figures from which conclusions can be drawn”**.
- Usually we collect many measurement on a person or object. Each measurement we call “**Variable**” and each person or object we call “**Observation**”.

Scale of Measurement



- **Nominal**

Data are labels or names used to identify an attribute of the element.

A nonnumeric label or numeric code may be used.

- Ordinal

The data have the properties of nominal data and the order or rank of the data is meaningful.

A nonnumeric label or numeric code may be used.

Interval

The data have the properties of ordinal data, and the interval between observations is expressed in terms of a fixed unit of measure.

Interval data are always numeric.

- **Ratio**

The data have all the properties of interval data and the ratio of two values is meaningful.

Variables such as distance, height, weight, and time use the ratio scale.

This scale must contain a zero value that indicates that nothing exists for the variable at the zero point.

categorical

numerical

Scale has levels that are:	Nominal	Ordinal	Interval	Ratio
Distinctive	X	X	X	X
Ordered		X	X	X
Equally spaced			X	X
Has an absolute zero				X

Example

Nominal	County where you live Race/ethnicity Favorite flavor of ice cream
Ordinal	Favorite size of coffee you order from Starbucks Birth order
Interval	IQ Score on Depression
Ratio	Number of computers in a household Temperature in Kelvin

A. Nominal B. Ordinal C. Interval D. Ratio

A. Nominal B. Ordinal C. Interval D. Ratio

A. Nominal B. Ordinal C. Interval D. Ratio

Finishing order of horse.

A. Nominal B. Ordinal C. Interval D. Ratio

A. Nominal B. Ordinal C. Interval D. Ratio

Number of football player’s jersey.

A. Nominal B. Ordinal C. Interval D. Ratio

Age

C. Garvan
3540 NW 29th Place
Gainesville, FL
32605

Mrs. Susan Plastaras
3 Brookdale Drive
Doylestown, PA
18901

Zip Code

A. Nominal B. Ordinal C. Interval D. Ratio

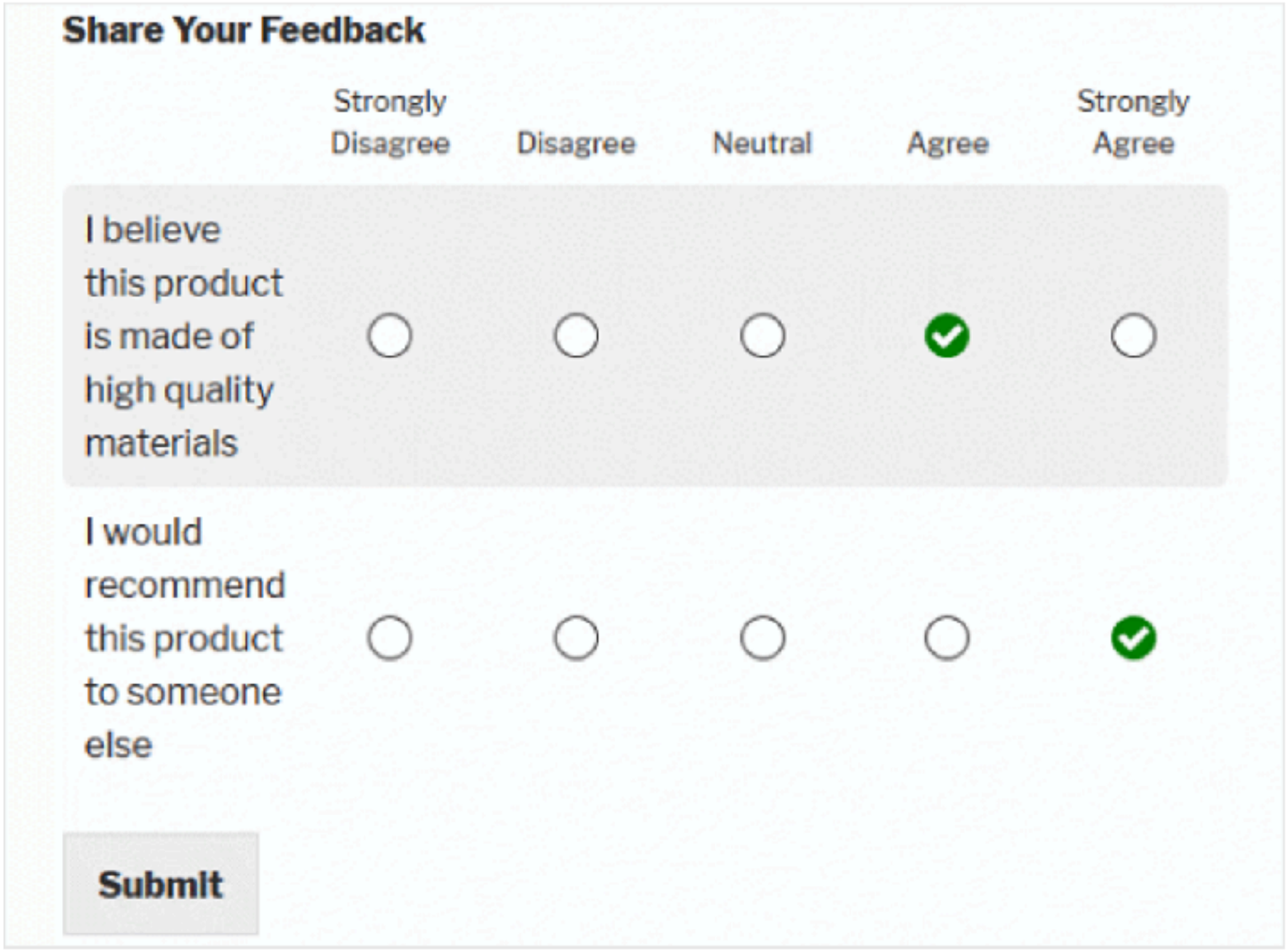
A. Nominal B. Ordinal C. Interval D. Ratio

Romatic Quiz

Question: Does love at first sight exist in your mind?

A) Definitely.
B) Maybe.
C) For some people.
D) Not at all.

A. Nominal B. Ordinal C. Interval D. Ratio



Getting data in R

R can import data from practically everywhere

- csv
- excel
- SPSS
- SAS
- Stata
- SQL
- XML
- JSON

Recap import data in R

```
# first row contains variable names, comma is separator
# assign the variable id to row names
# note the / instead of \ on mswindows systems

mydata <- read.table("c:/mydata.csv", header=TRUE,
   sep=",", row.names="id")
```

Data Frame

Function

Keep first line as a header

Field Separation

Filename to import

Getting Data

Iris Data Set from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris)

Attribute Information:

1. Sepal Length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal length in cm
5. Classes:
   - Iris Setosa
   - Iris Versicolour
   - Iris Virginica

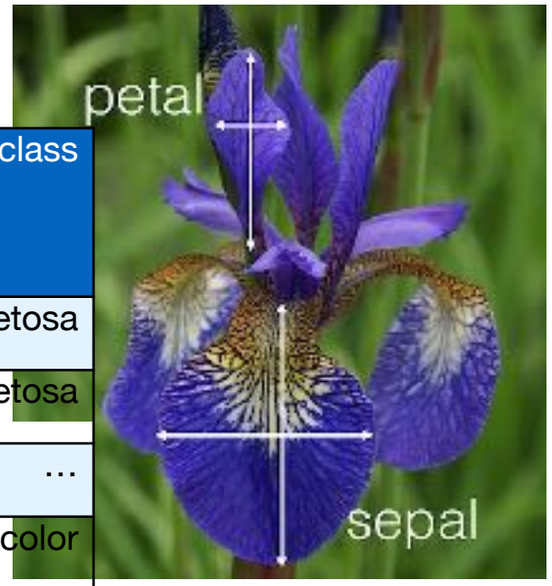
Nomenclature

IRIS

https://archive.ics.uci.edu/ml/datasets/Iris

Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
…	…	…	…	…	…
50	6.4	3.2	4.5	1.5	veriscolor
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	virginica



Features (attributes, dimensions, variables)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

Iris setosa

Iris versicolor

Iris virginica

Getting Data

> iris <- read.csv(“iris.data.csv”, header=TRUE)

> library(datasets)

> iris

> colnames(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> nrow(iris)
[1] 150
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
>
```

Workshop 2.1 Getting Data

- Choose data from dataset below.
  - Sales Win or Loss Dataset
  - HR Employee Attrition Dataset
  - Telco Customer Churn Dataset
  - Titanic Survival Dataset
  - Or data you bring from your work
- Import data and explore your data using str to see what types of data you selected. Is it in the correct data format?

Visualization (Exploratory Data Analysis)

Exploratory Data Analysis (EDA)

- The goal during EDA is to develop an understanding of your data.
- The easiest way to do this is to use questions as tools to guide your investigation.
- When you ask a question, the question focuses your attention on a specific part of your dataset and helps you decide which graphs, models, or transformations to make.
- Two types of questions will always be useful for making discoveries within your data.
    1. What type of variation occurs within my variables?
    2. What type of covariation occurs between my variables?

Variation

- The tendency of the values of a variable to change from measurement to measurement. You can see variation easily in real life.
- If you measure any continuous variable twice, you will get two different results. This is true even if you measure quantities that are constant, like the speed of light. Categorical variables can also vary if you measure across different subjects or different times.
- Every variable has its own pattern of variation, which can reveal interesting information. The best way to understand that pattern is to visualise the distribution of the variable’s values.

Visualizing distributions

How you visualize the distribution of a variable will depend on whether the variable is categorical or continuous.

- A variable is **categorical** if it can only take one of a small set of values. In R, **categorical variables** are usually saved as factors or character vectors. To examine the distribution of a categorical variable, **use a bar chart.**
- A variable is **continuous** if it can take any of an infinite set of ordered values. Numbers and date-times are two examples of continuous variables. To examine the distribution of a **continuous** variable, **use a histogram**:

Covariation

If variation describes the behavior within a variable, **covariation describes the behavior between variables**. **Covariation is the tendency for the values of two or more variables to vary together in a related way**. The best way to spot covariation is to visualize the relationship between two or more variables. How you do that should again depend on the type of variables involved.

1) **A categorical and continuous variable** : We want to explore the distribution of a continuous variable broken down by a categorical variable. Boxplot can help us

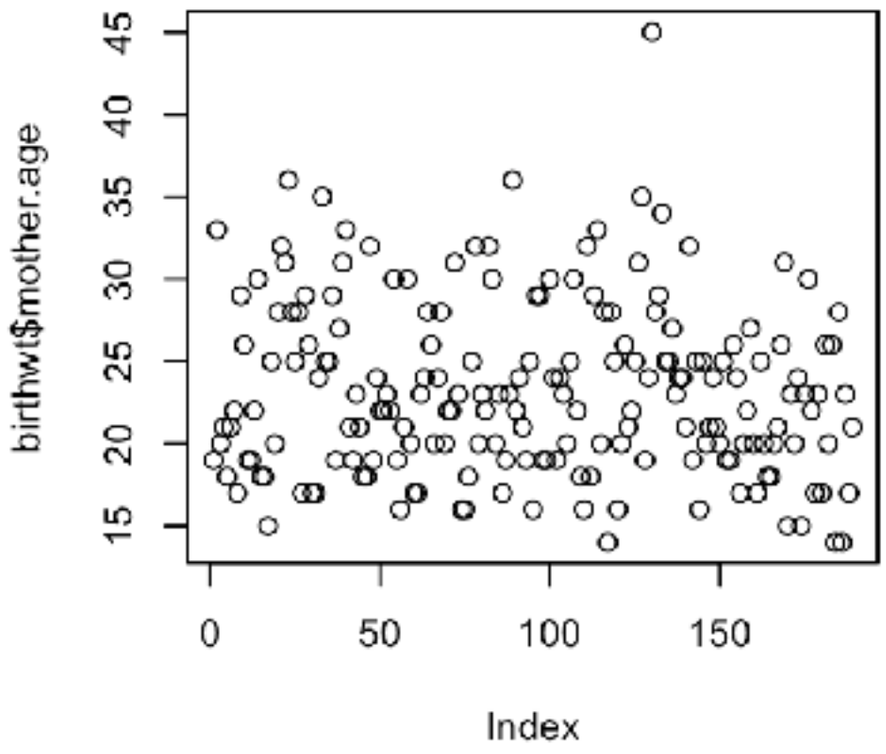
2) Two continuous variables : One great way to visualize the covariation between two continuous variables: draw a **scatterplot**.

Basic single plot

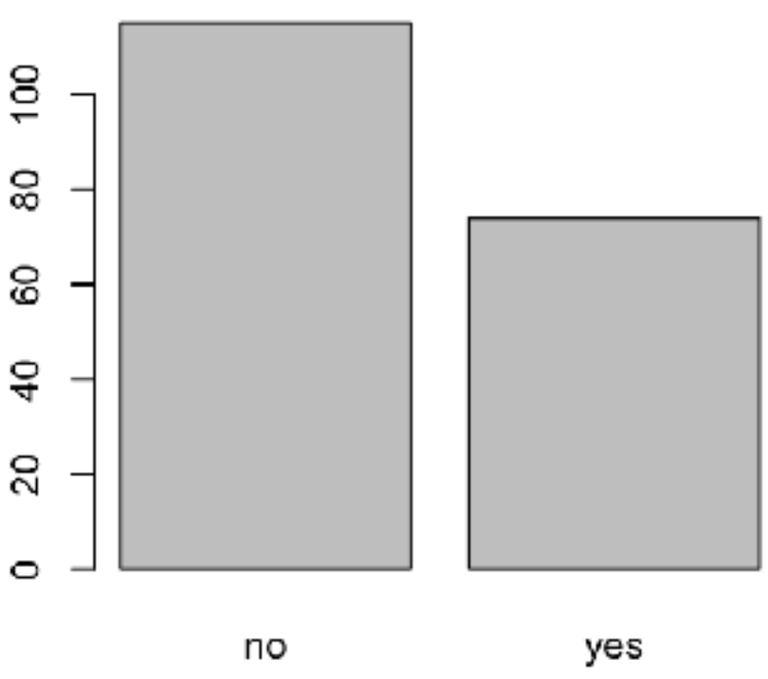
Let's continue with the `birthwt` data from the `MASS` library.

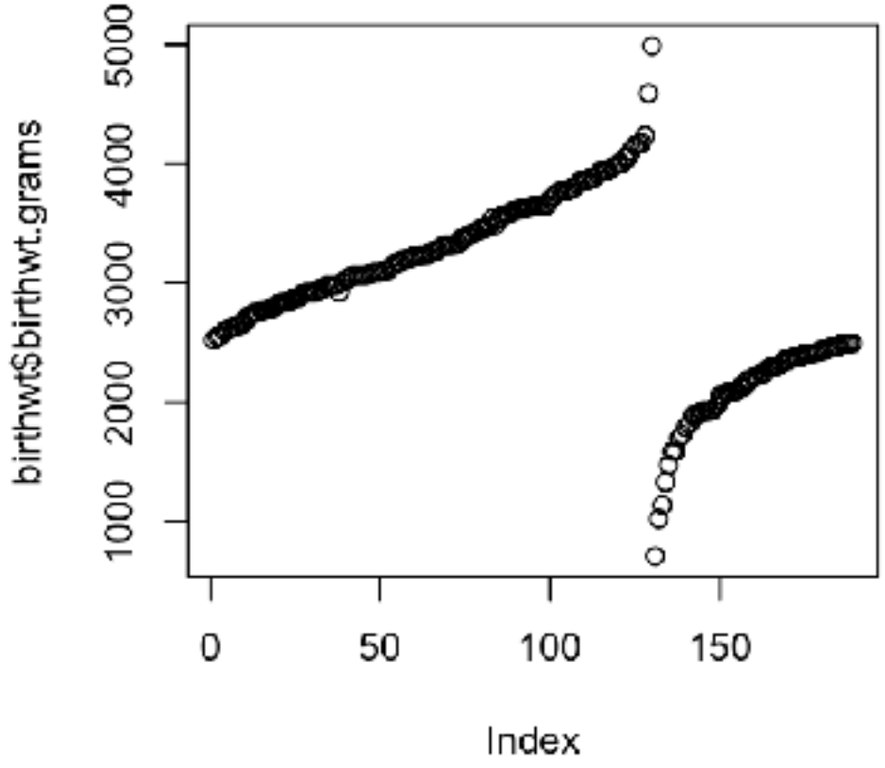
Here are some basic single-variable plots.

```
par(mfrow = c(2,2)) # Display plots in a single 2 x 2 figure
plot(birthwt$mother.age)
with(birthwt, hist(mother.age))
plot(birthwt$mother.smokes)
plot(birthwt$birthwt.grams)
```


birthwt$mother.age
15
20
25
30
35
40
45
0
50
100
150
Index


Histogram of mother.age
Frequency
0
10
20
30
40
50
60
10
15
20
25
30
35
40
45
mother.age

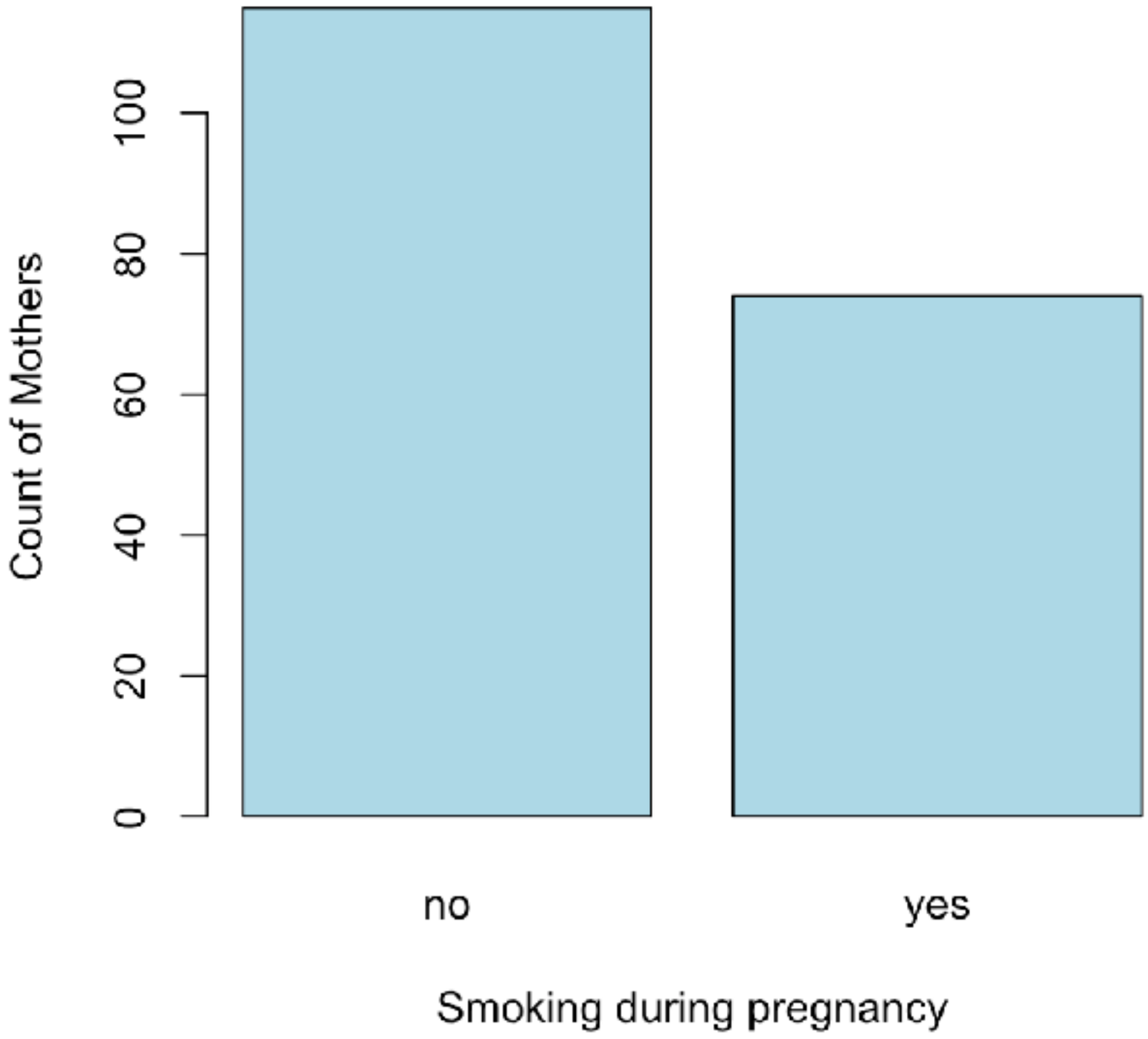

0
20
40
60
80
100
no
yes


birthwt$birthwt.grams
1000
2000
3000
4000
5000
0
50
100
150
Index

Another example

Let's add more information to the smoking bar plot, and also change the color by setting the `col` option.

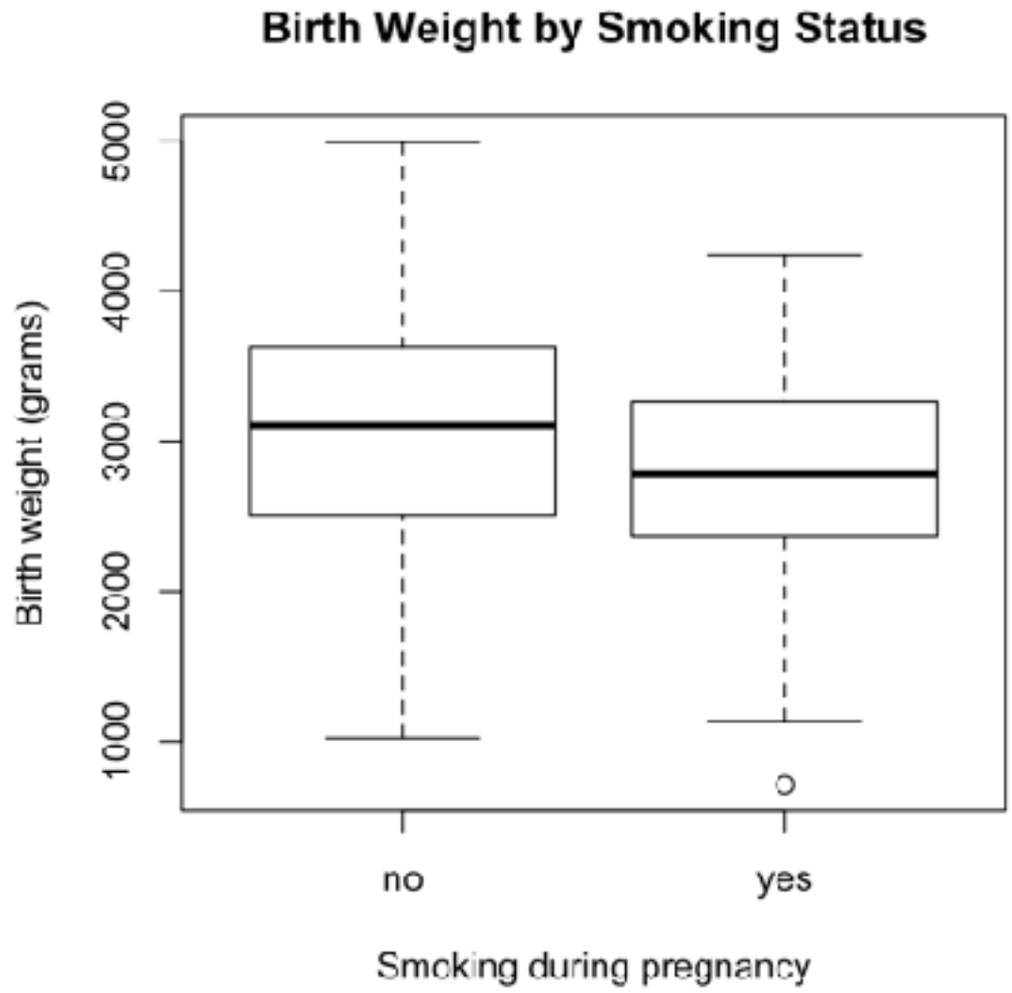
```
plot(birthwt$mother.smokes,
     main = "Mothers Who Smoked In Pregnancy",
     xlab = "Smoking during pregnancy",
     ylab = "Count of Mothers",
     col = 'lightblue')
```


Mothers Who Smoked In Pregnancy
Count of Mothers
0
20
40
60
80
100
no
yes
Smoking during pregnancy

Plots with several variables

If we call `plot(x,y,...)` with `x` a factor and `y` numeric, R will produce boxplots of `y` at every level of `x`.

```
with(birthwt, plot(mother.smokes, birthwt.grams,
                   main = "Birth Weight by Smoking Status",
                   xlab = "Smoking during pregnancy",
                   ylab = "Birth weight (grams)"))
```

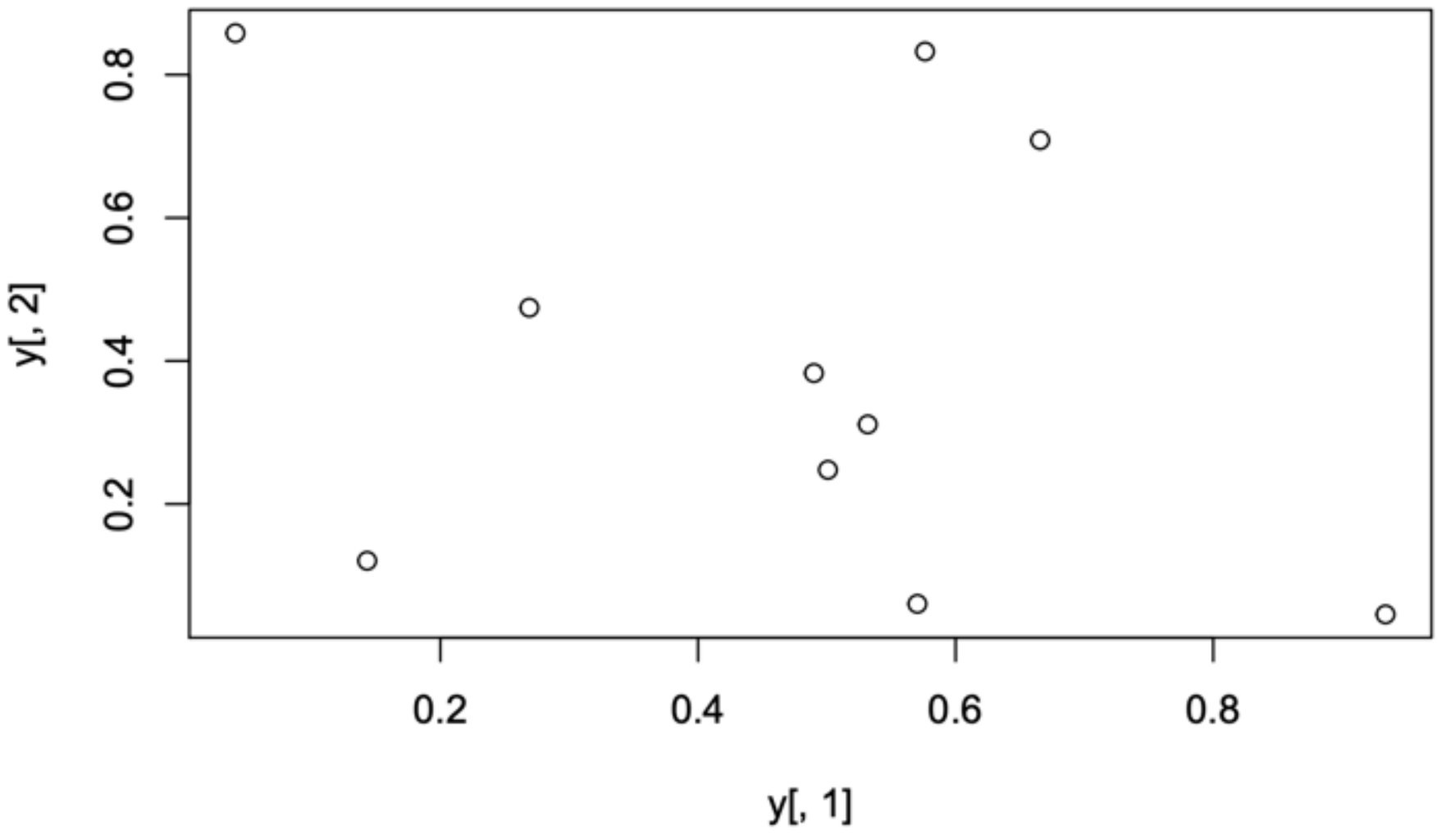


Basic Scatter Plots

Basic scatter plots

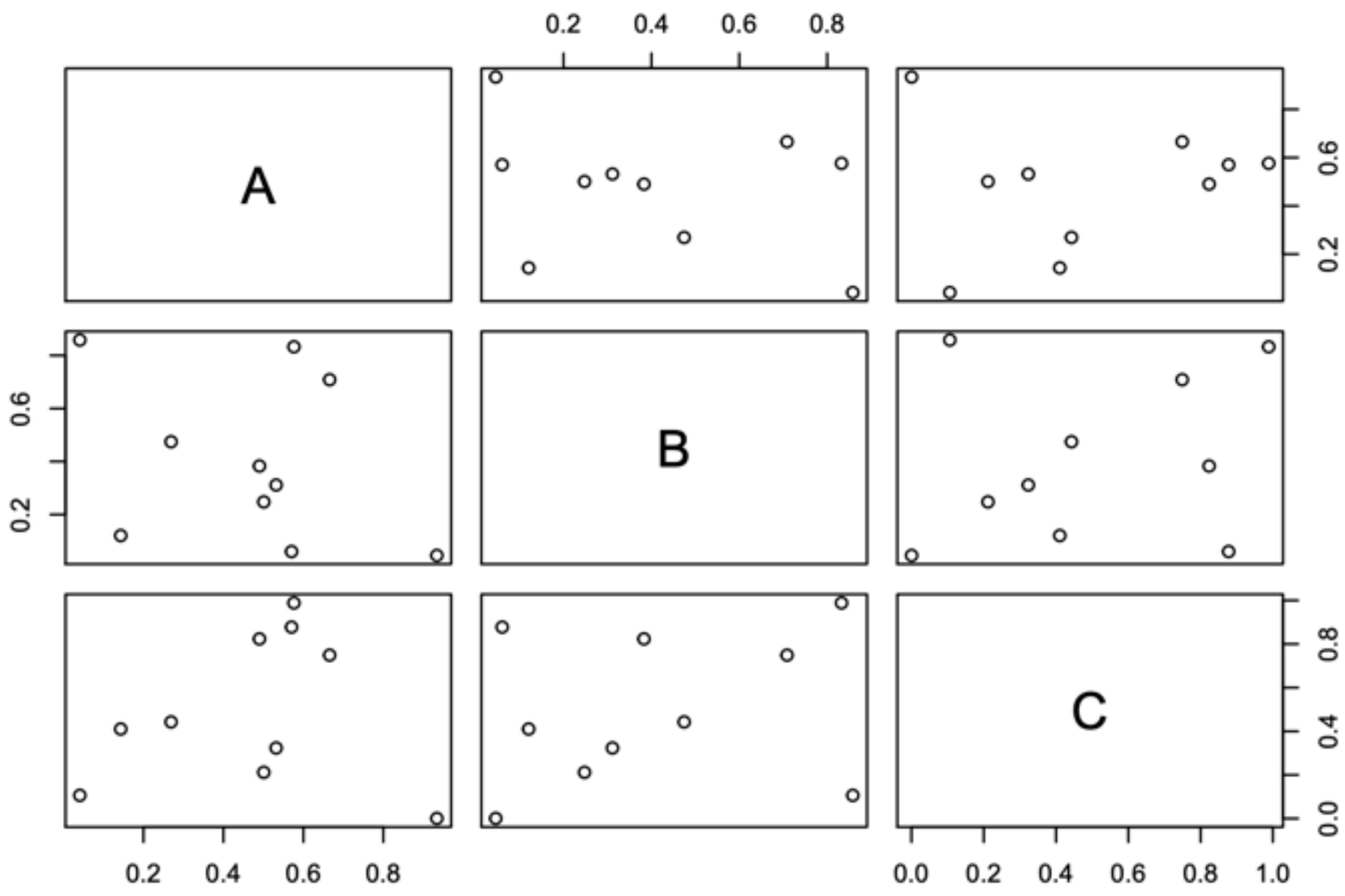
Sample data set for subsequent plots

```
set.seed(1410)
y <- matrix(runif(30), ncol=3, dimnames=list(letters[1:10], LETTERS[1:3]))
plot(y[,1], y[,2])
```



Pairs

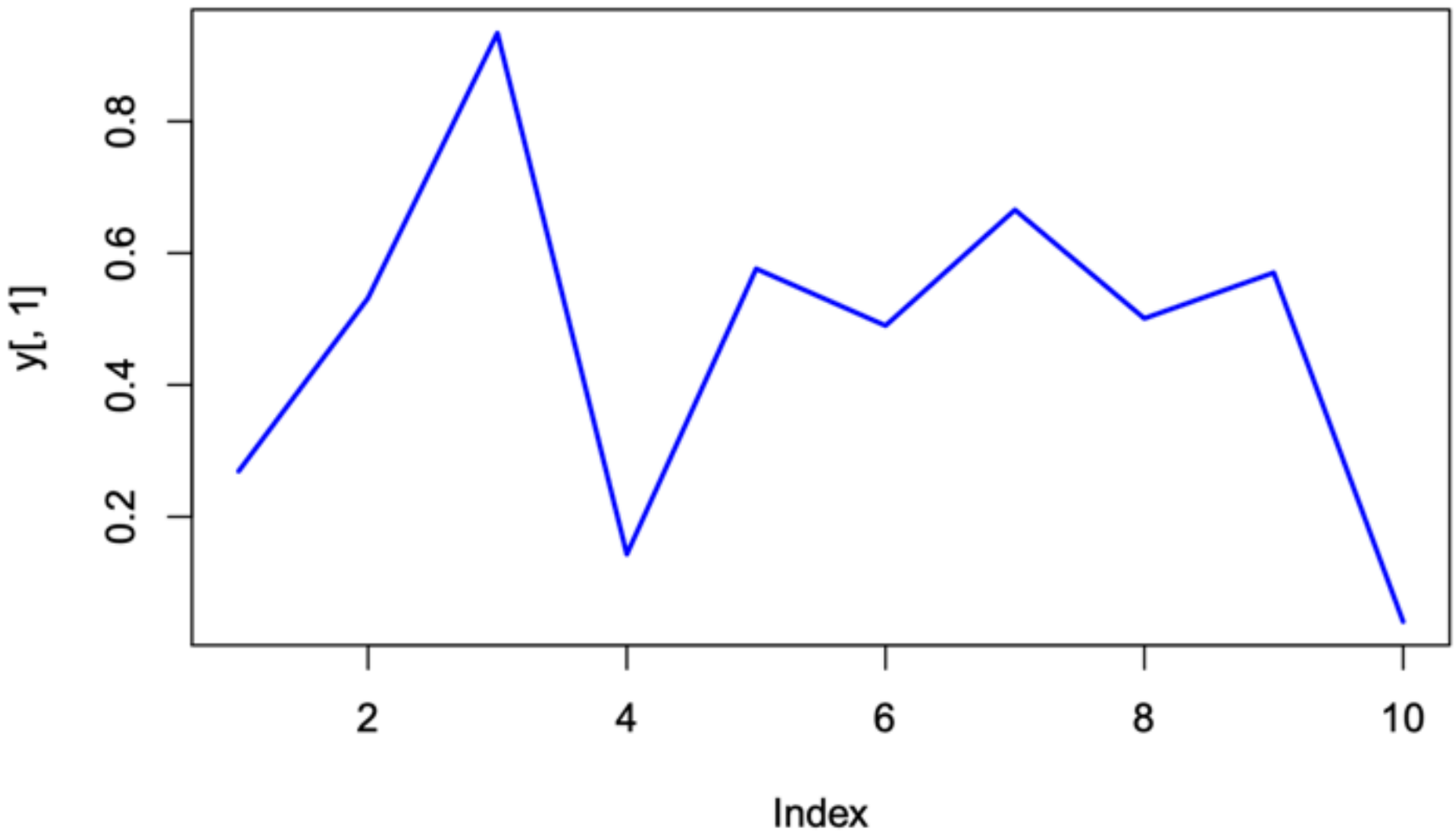
```
pairs(y)
```



Line Plots

Single Data Set

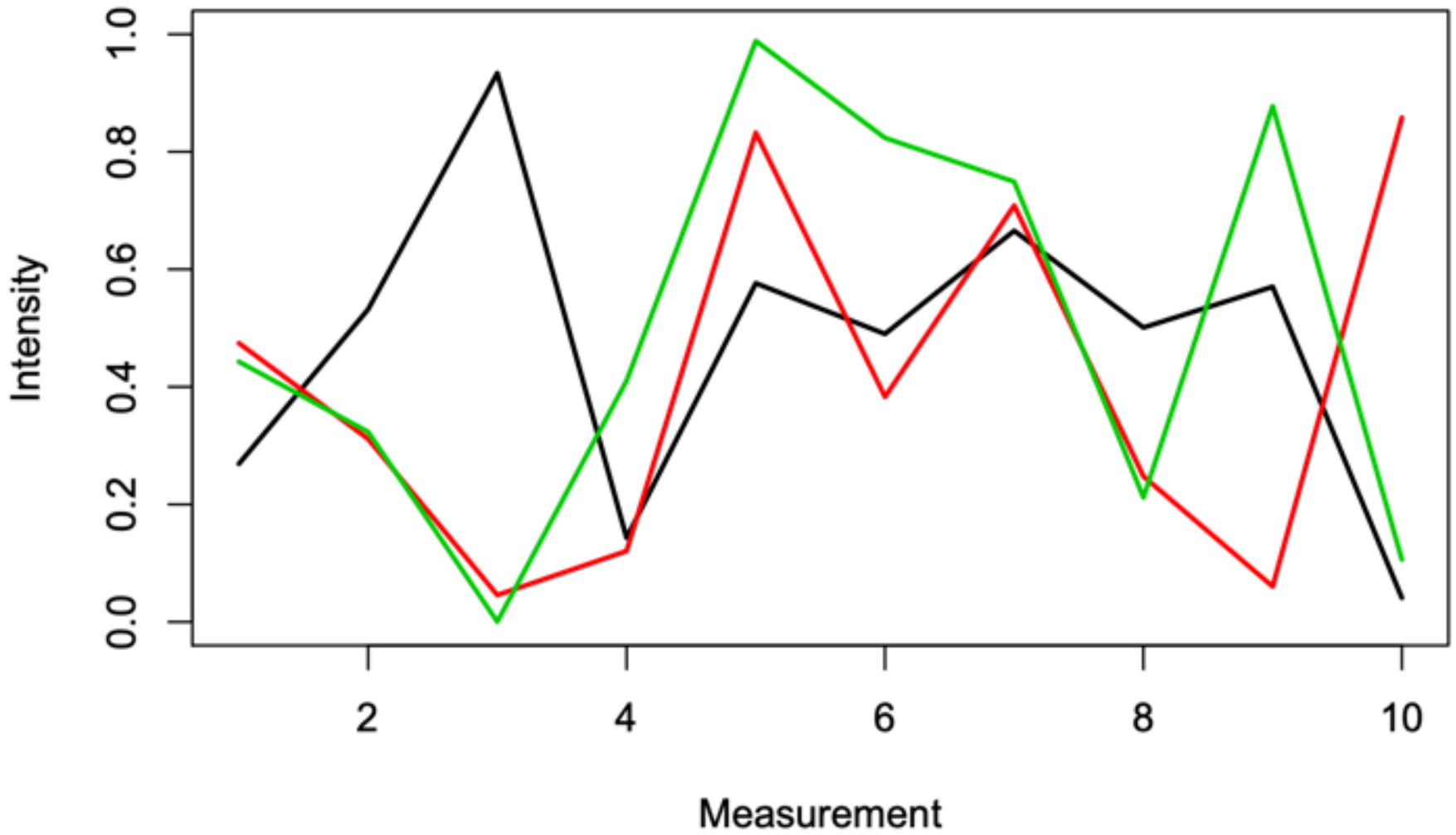
```
plot(y[,1], type="l", lwd=2, col="blue")
```



Line Plot Many Data Set

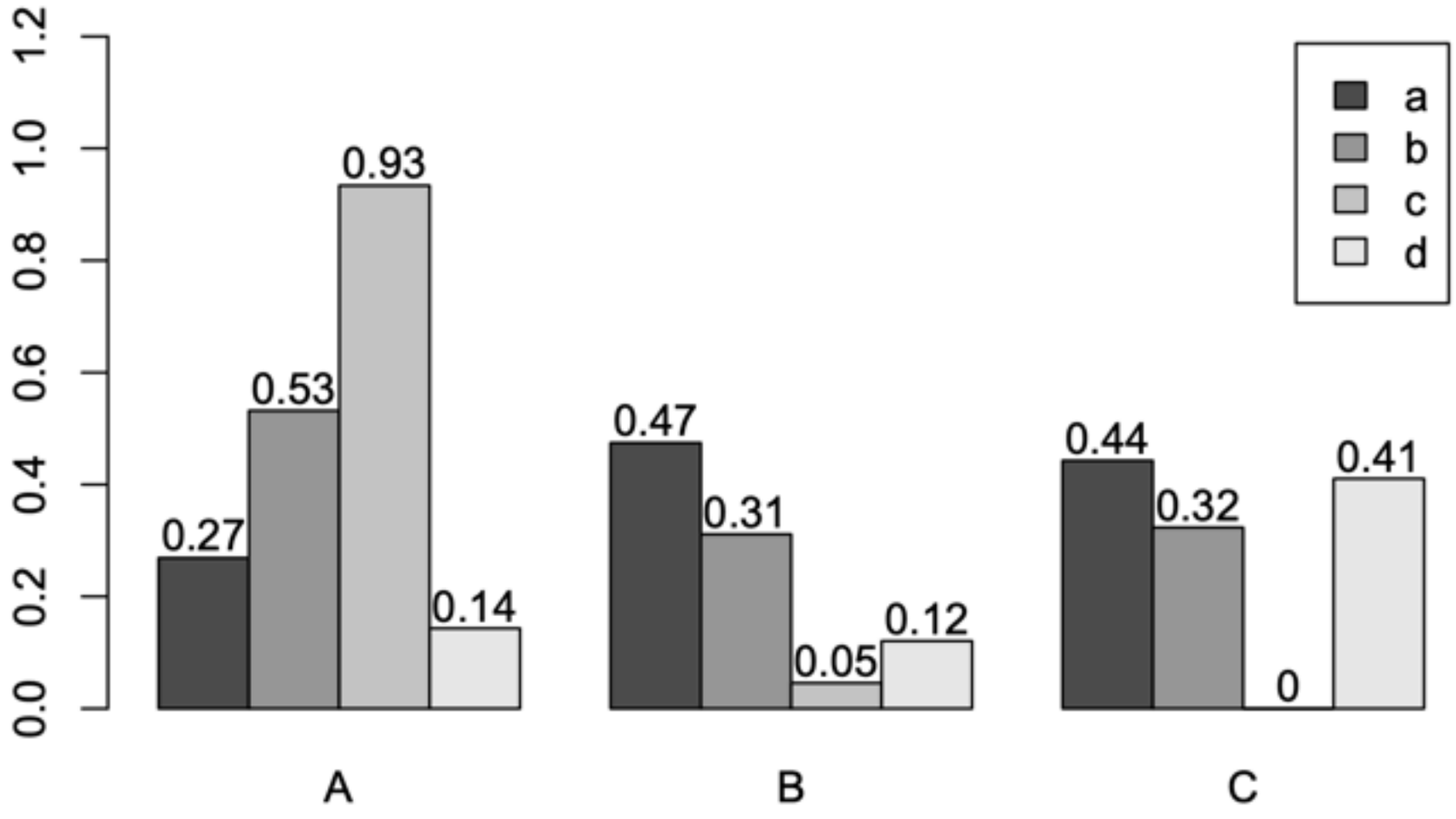
```
## [1] 1
```

```
plot(y[,1], ylim=c(0,1), xlab="Measurement", ylab="Intensity", type="l", lwd=2, col=1)
for(i in 2:length(y[1,])) {
    screen(1, new=FALSE)
    plot(y[,i], ylim=c(0,1), type="l", lwd=2, col=i, xaxt="n", yaxt="n", ylab="",
             xlab="", main="", bty="n")
}
```



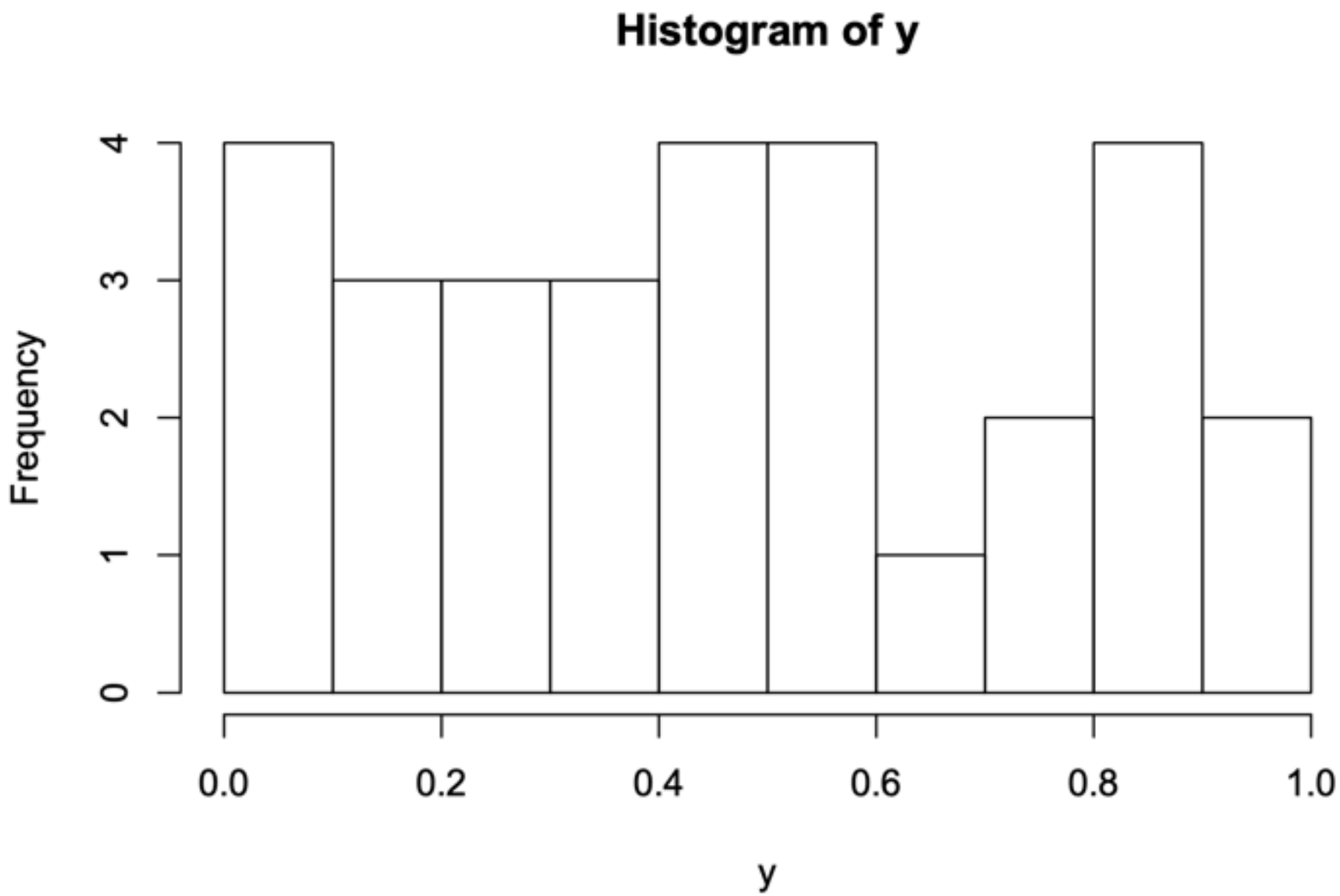
Bar Plot

```
barplot(y[1:4,], ylim=c(0, max(y[1:4,])+0.3), beside=TRUE,
        legend=letters[1:4])
text(labels=round(as.vector(as.matrix(y[1:4,])),2), x=seq(1.5, 13, by=1)
     +sort(rep(c(0,1,2), 4)), y=as.vector(as.matrix(y[1:4,]))+0.04)
```



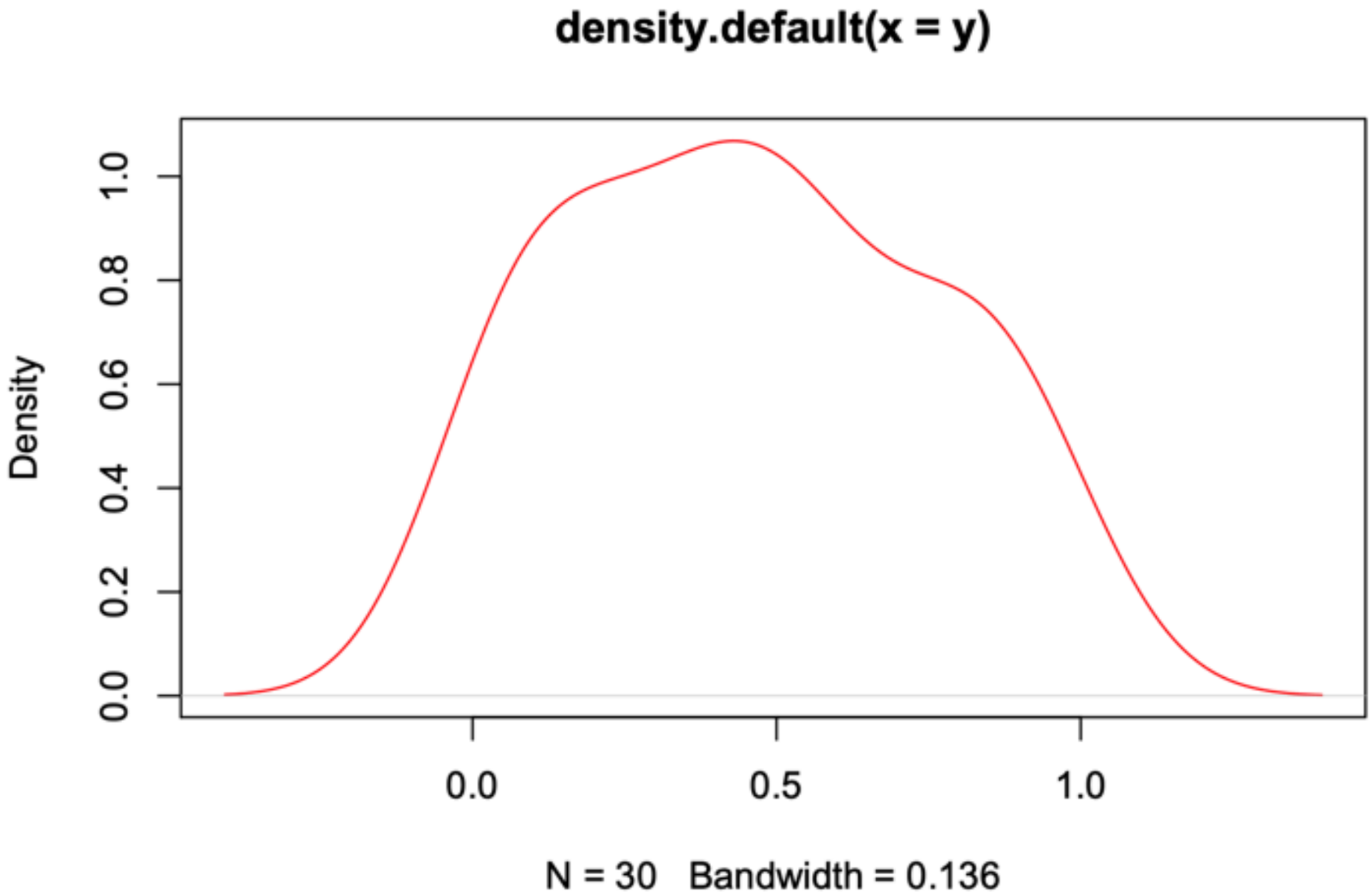
Histogram

```
hist(y, freq=TRUE, breaks=10)
```



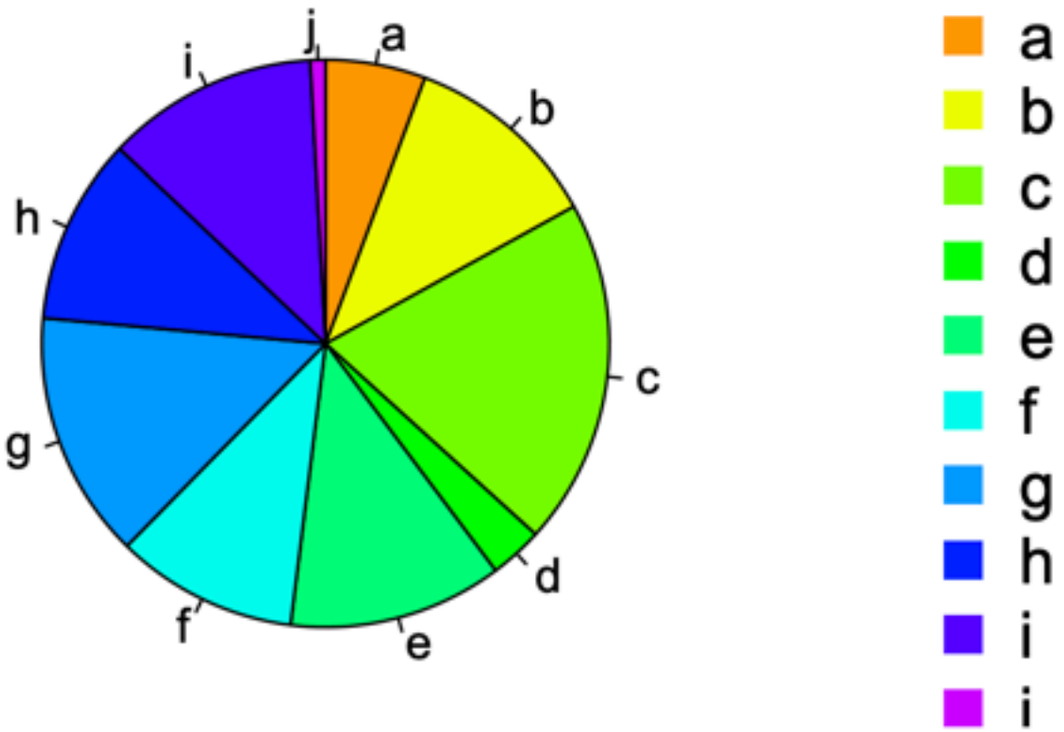
Density Plot

```
plot(density(y), col="red")
```



Pie Chart

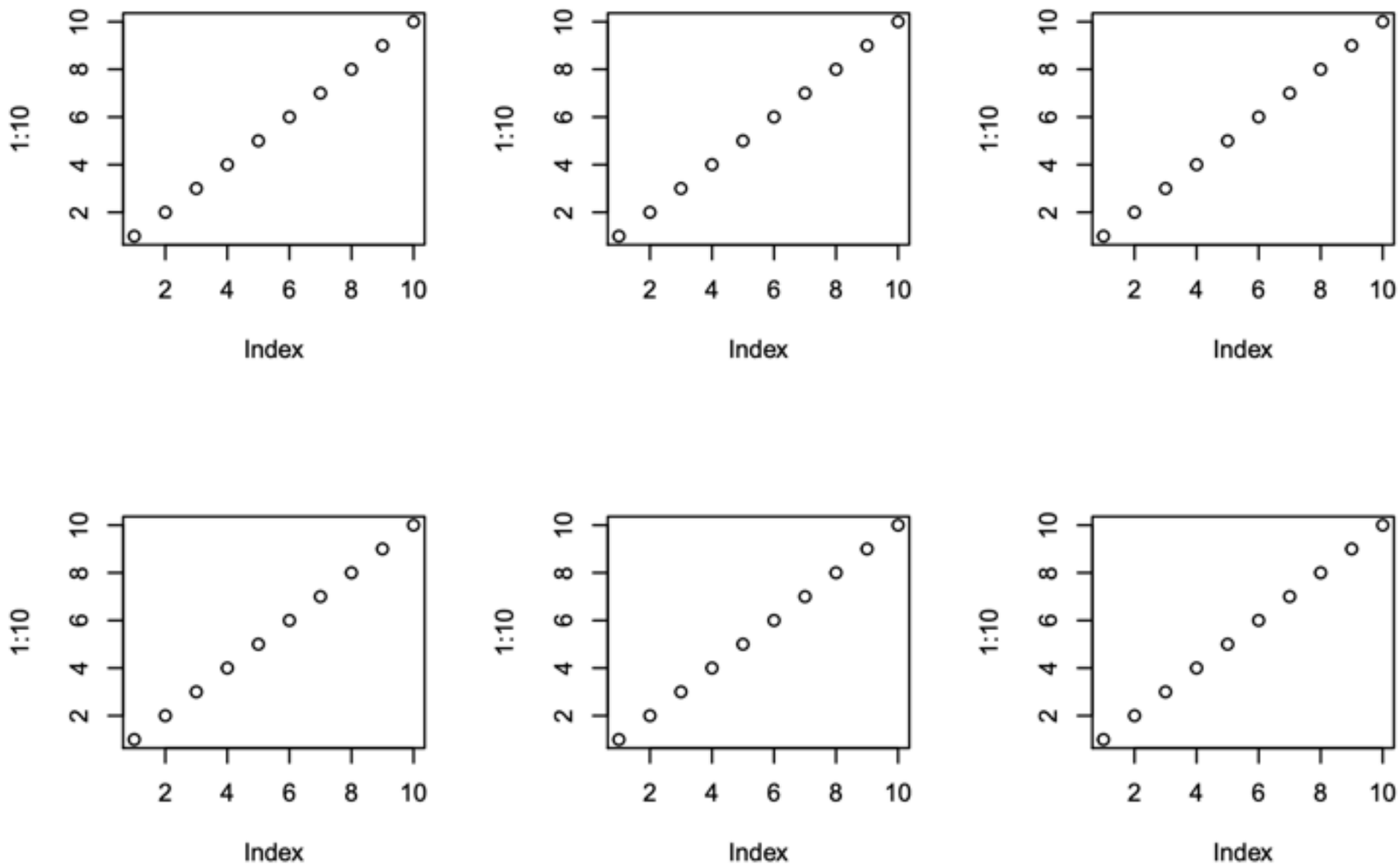
```
pie(y[,1], col=rainbow(length(y[,1]), start=0.1, end=0.8), clockwise=TRUE)
legend("topright", legend=row.names(y), cex=1.3, bty="n", pch=15, pt.cex=1.8,
col=rainbow(length(y[,1]), start=0.1, end=0.8), ncol=1)
```



Several Plot on Single Pane

With `par(mfrow=c(nrow,ncol))` one can define how several plots are arranged next to each other.

```
par(mfrow=c(2,3)); for(i in 1:6) { plot(1:10) }
```



Plotting using ggplot2

What is ggplot2?

- An implementation of the Grammar of Graphics by Leland Wilkinson
- Written by Hadley Wickham (while he was a graduate student at Iowa State)
- A “third” graphics system for R (along with base and lattice)
- Available from CRAN via install.packages()
- Web site: http://ggplot2.org (better documentation)

What is ggplot2?

- Grammar of graphics represents and abstraction of graphics ideas/ objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects
- “Shorten the distance from mind to page”

Grammar of Graphics

- “In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system”
- from ggplot2 book

The Basics: qplot()

- Works much like the plot function in base graphics system
- Looks for data in a data frame, similar to lattice, or in the parent environment
- Plots are made up of aesthetics (size, shape, color) and geoms (points, lines)

The Basics: qplot()

- Factors are important for indicating subsets of the data (if they are to have different properties); they should be labeled
- The qplot() hides what goes on underneath, which is okay for most operations
- ggplot() is the core function and very flexible for doing things qplot() cannot do

qplot

The syntax of qplot is similar as R's basic plot function

Arguments

– x: x-coordinates (e.g. col1)

– y: y-coordinates (e.g. col2)

– data: data.frame or tibble with corresponding column names

– xlim, ylim: e.g. xlim=c(0,10)

– log: e.g. log="x" or log="xy"

– main: main title; see ?plotmath for mathematical formula

– xlab, ylab: labels for the x- and y-axes

– color, shape, size

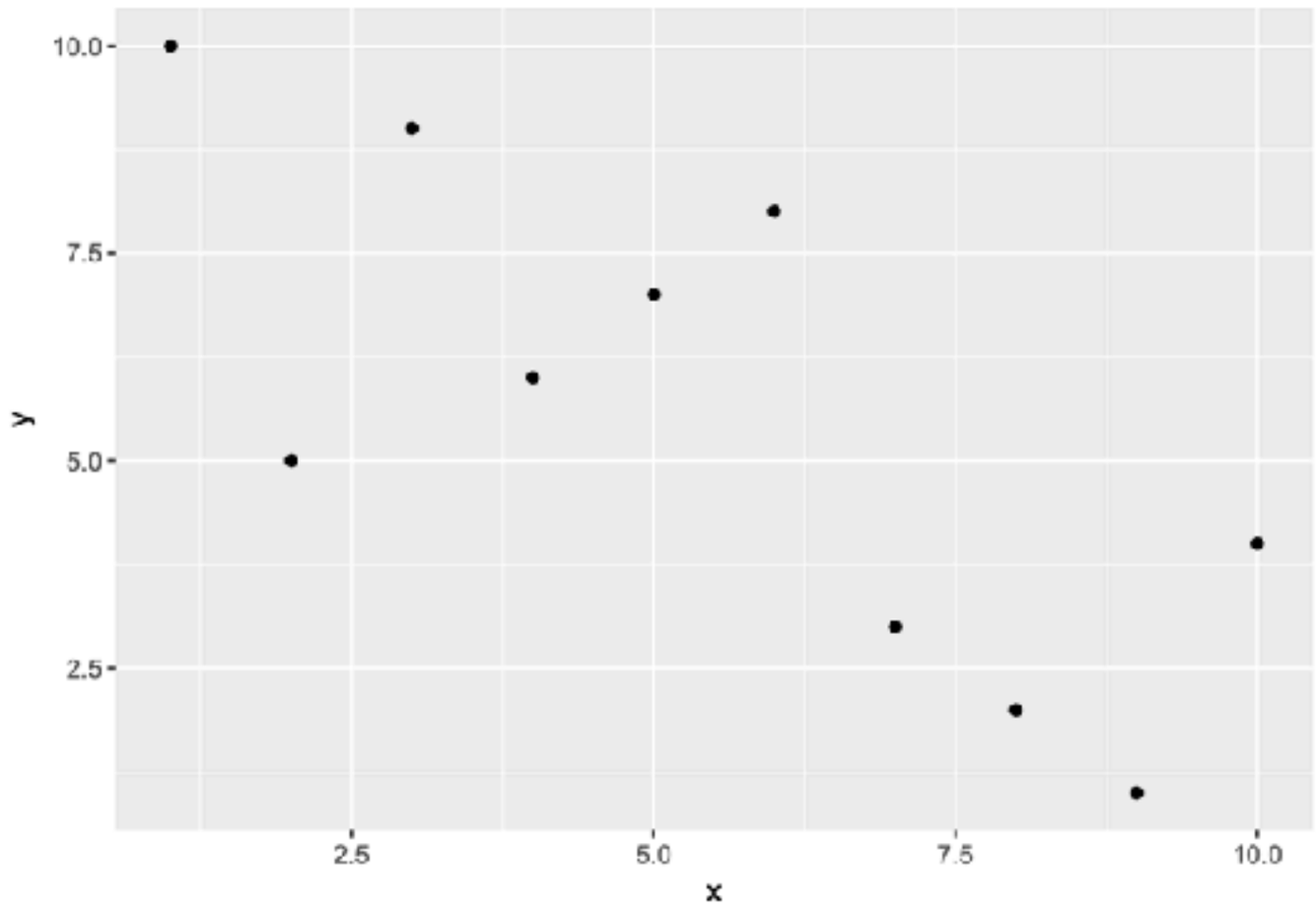
– ...: many arguments accepted by plot function

Create sample data

```
library(ggplot2)
x <- sample(1:10, 10); y <- sample(1:10, 10); cat <- rep(c("A", "B"), 5)
```

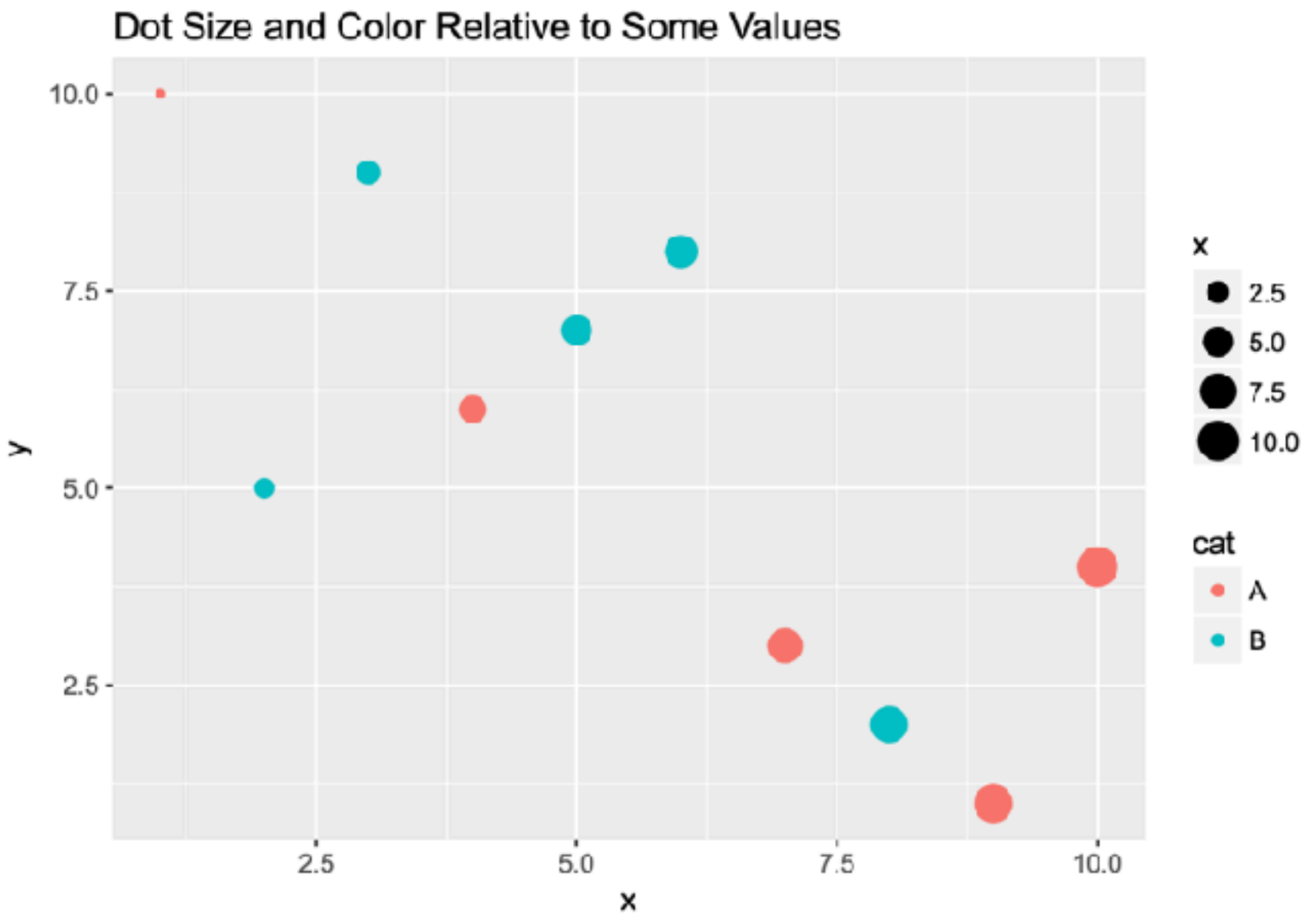
Simple scatter plot

```
qplot(x, y, geom="point")
```



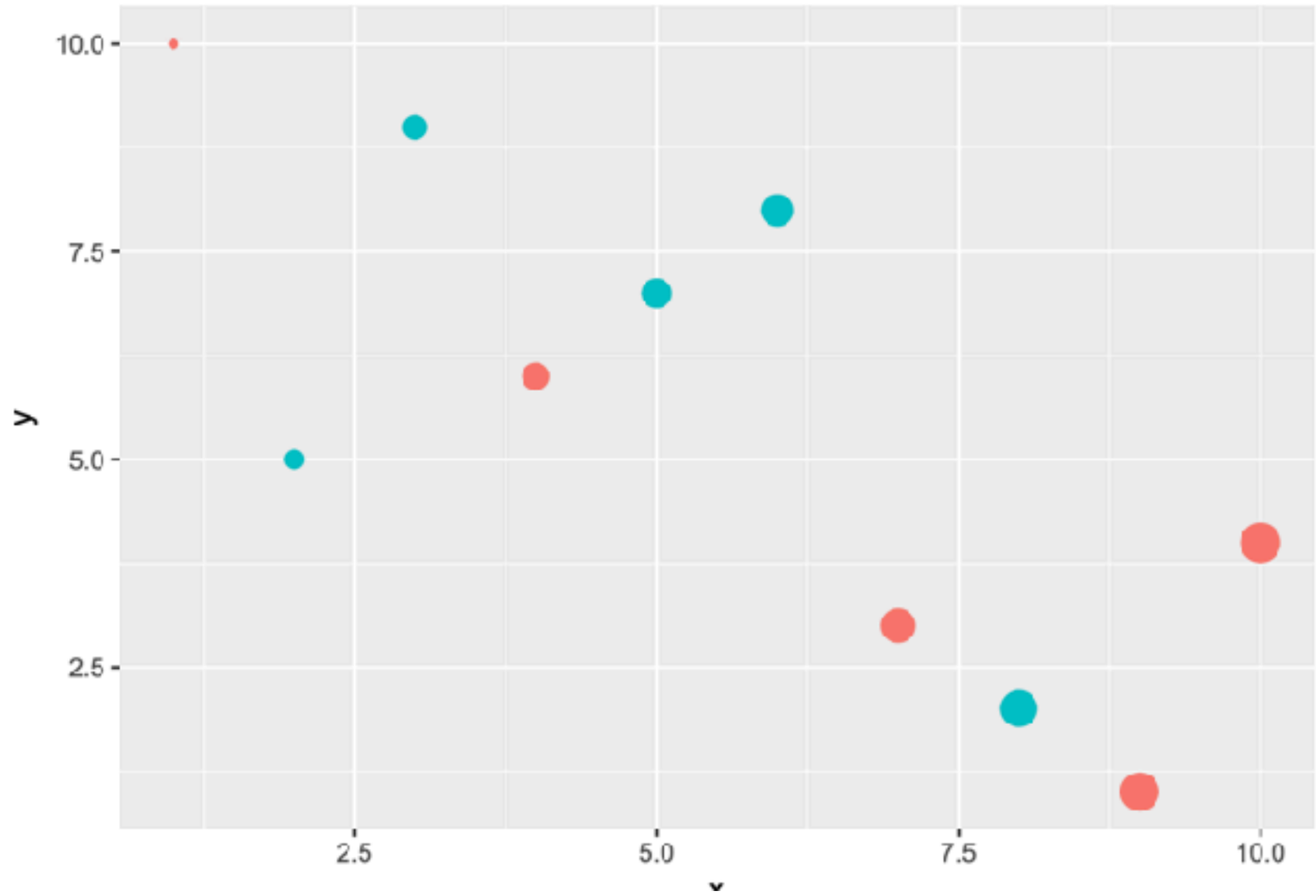
Prints dots with different sizes and colors

```
qplot(x, y, geom="point", size=x, color=cat,
      main="Dot Size and Color Relative to Some Values")
```



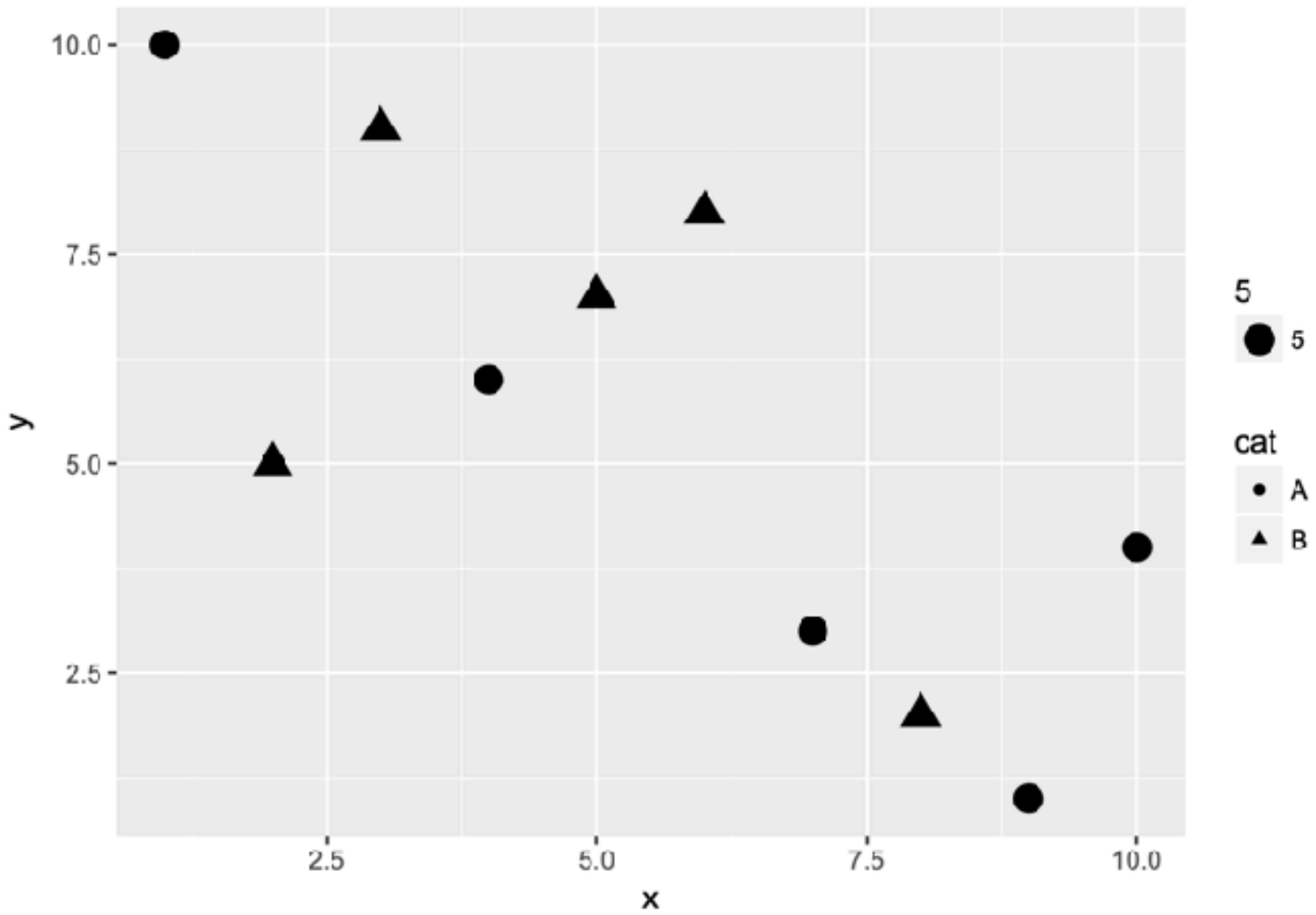
Drops legend

```
qplot(x, y, geom="point", size=x, color=cat) +
      theme(legend.position = "none")
```



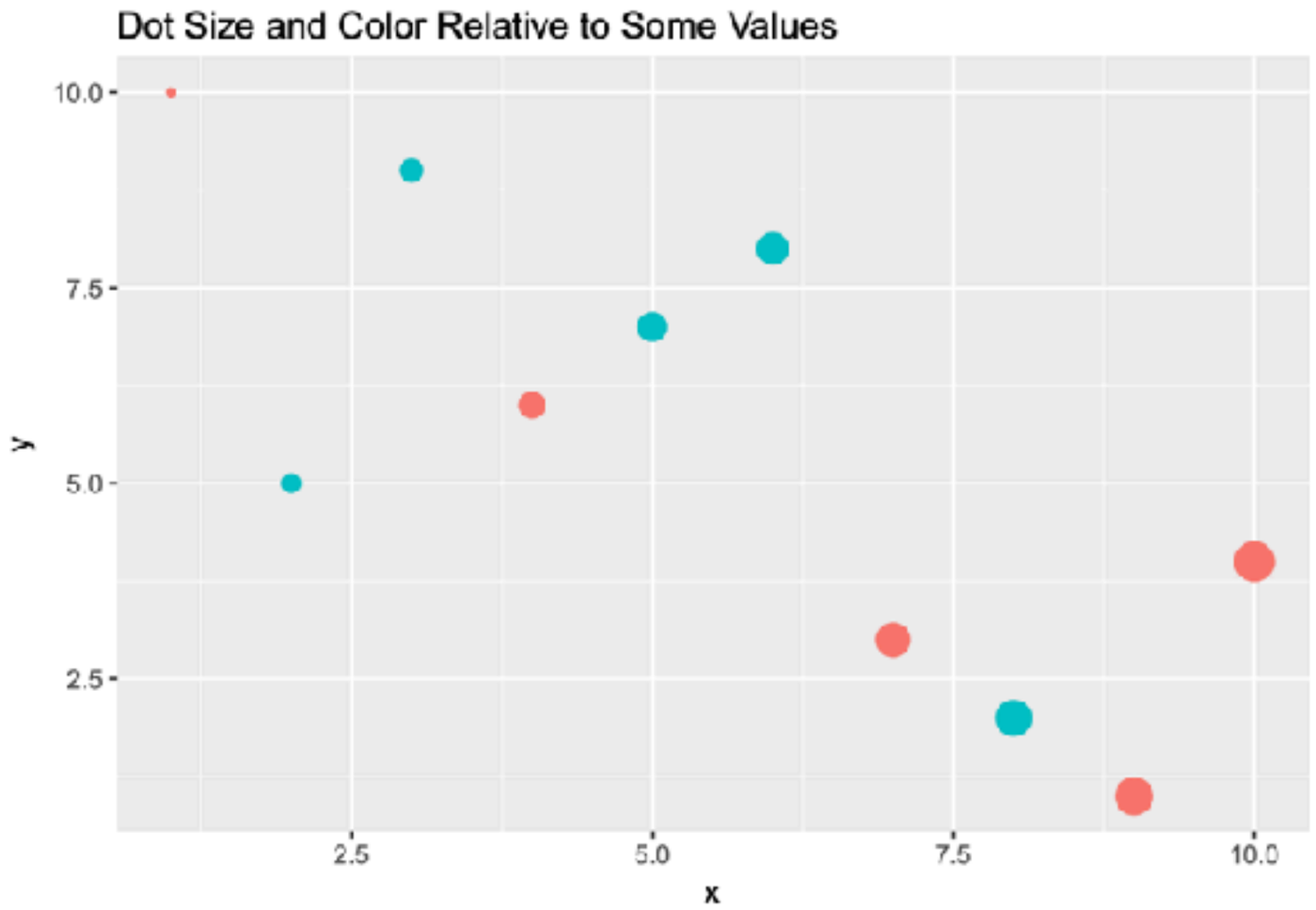
Plot different shapes

```
qplot(x, y, geom="point", size=5, shape=cat)
```



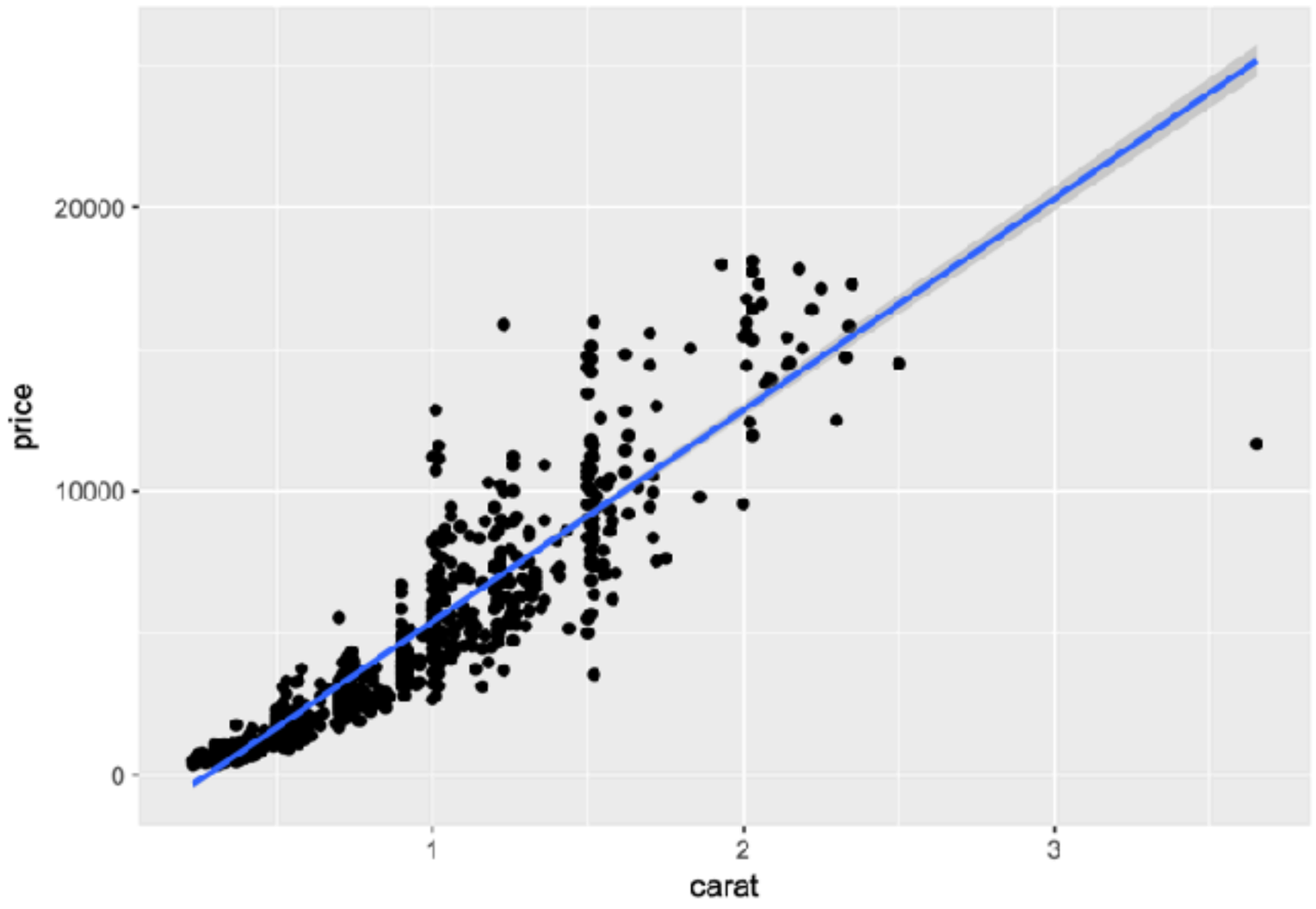
Colored groups

```
p <- qplot(x, y, geom="point", size=x, color=cat,
             main="Dot Size and Color Relative to Some Values") +
     theme(legend.position = "none")
print(p)
```



Regression Line

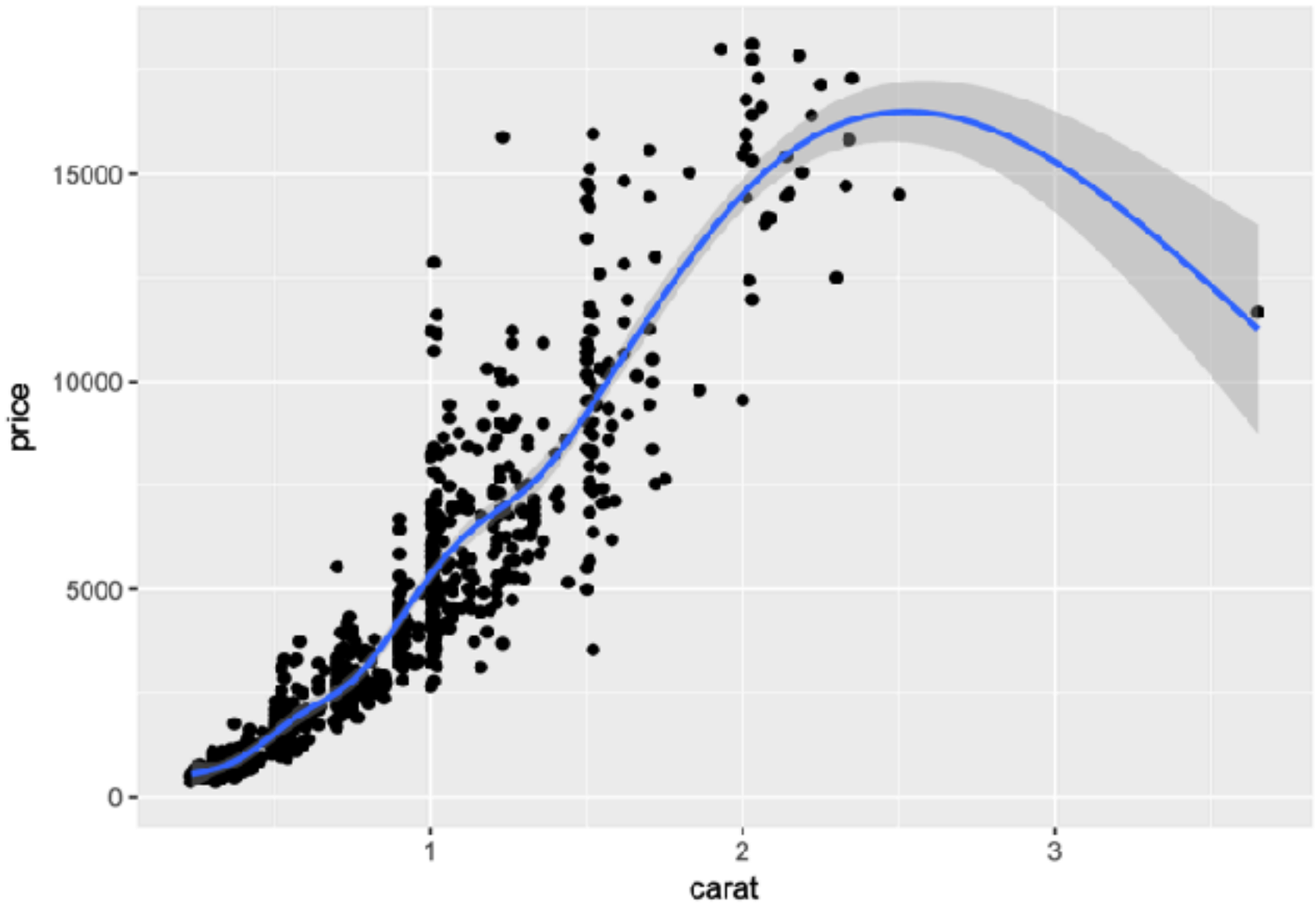
```
set.seed(1410)
dsmall <- diamonds[sample(nrow(diamonds), 1000), ]
p <- qplot(carat, price, data = dsmall) +
           geom_smooth(method="lm")
print(p)
```



Local regression curve (loess)

```
p <- qplot(carat, price, data=dsmall, geom=c("point", "smooth"))
print(p) # Setting se=FALSE removes error shade
```

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



ggplot Function

- More important than `qplot` to access full functionality of `ggplot2`
- Main arguments
  - data set, usually a `data.frame` or `tibble`
  - aesthetic mappings provided by `aes` function
- General `ggplot` syntax
  - `ggplot(data, aes(...)) + geom() + ... + stat() + ...`
- Layer specifications
  - `geom(mapping, data, ..., geom, position)`
  - `stat(mapping, data, ..., stat, position)`
- Additional components
  - `scales`
  - `coordinates`
  - `facet`
- `aes()` mappings can be passed on to all components (`ggplot`, `geom`, etc.). Effects are global when passed on to `ggplot()` and local for other components.
  - `x, y`
  - `color`: grouping vector (factor)
  - `group`: grouping vector (factor)

Changing Plotting Themes in ggplot

- Theme settings can be accessed with `theme_get()`
- Their settings can be changed with `theme()`

Example how to change background color to white

```
... + theme(panel.background=element_rect(fill = "white", colour = "black"))
```

Storing ggplot Specifications

Plots and layers can be stored in variables

```
p <- ggplot(dsmall, aes(carat, price)) + geom_point()
p # or print(p)
```

Returns information about data and aesthetic mappings followed by each layer

```
summary(p)
```

Print dots with different sizes and colors

```
bestfit <- geom_smooth(method = "lm", se = F, color = alpha("steelblue", 0.5), size = 2)
p + bestfit # Plot with custom regression line
```

Syntax to pass on other data sets

```
p %+% diamonds[sample(nrow(diamonds), 100),]
```

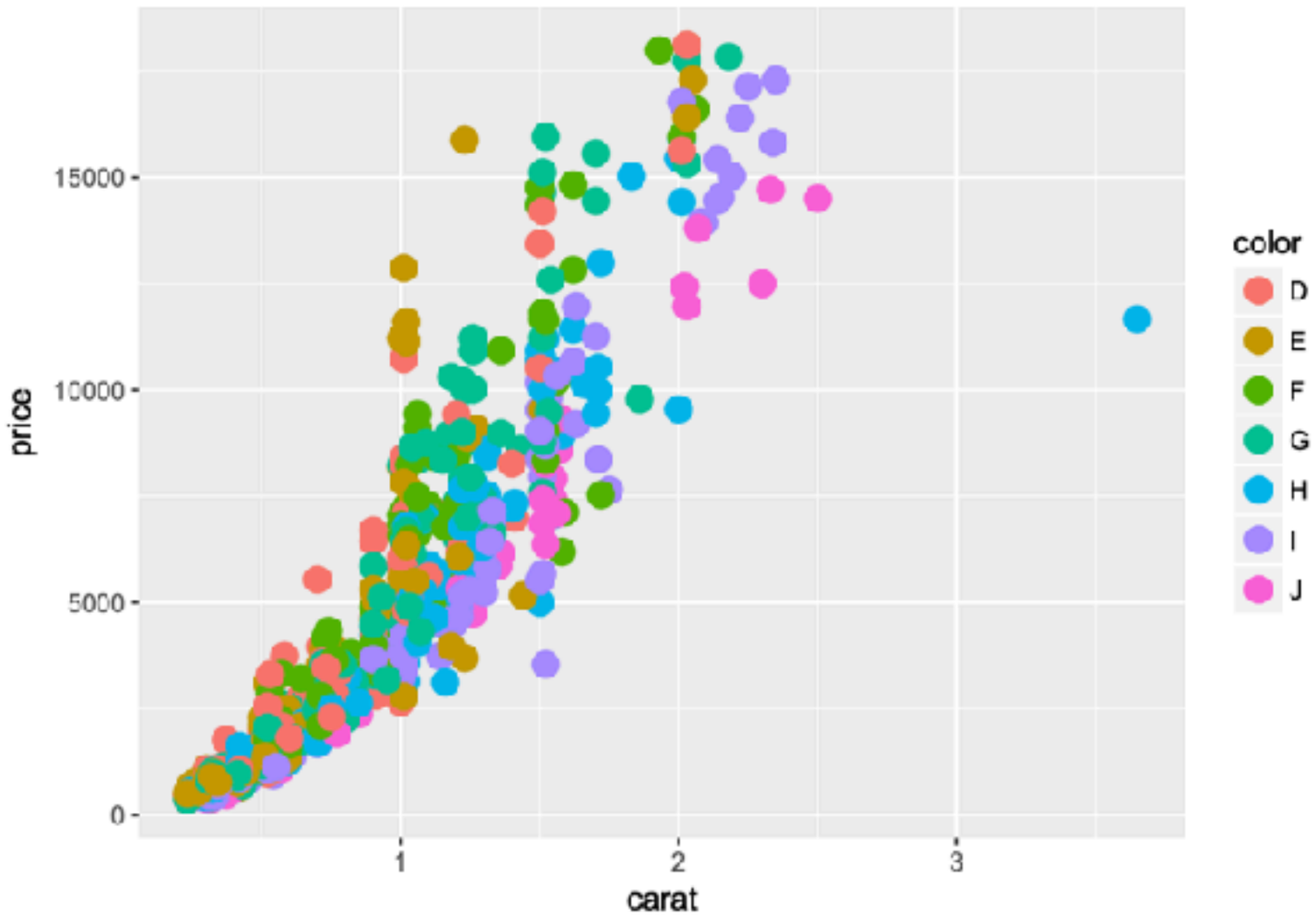
Saves plot stored in variable p to file

```
ggsave(p, file="myplot.pdf")
```

ggplot : scatter plot

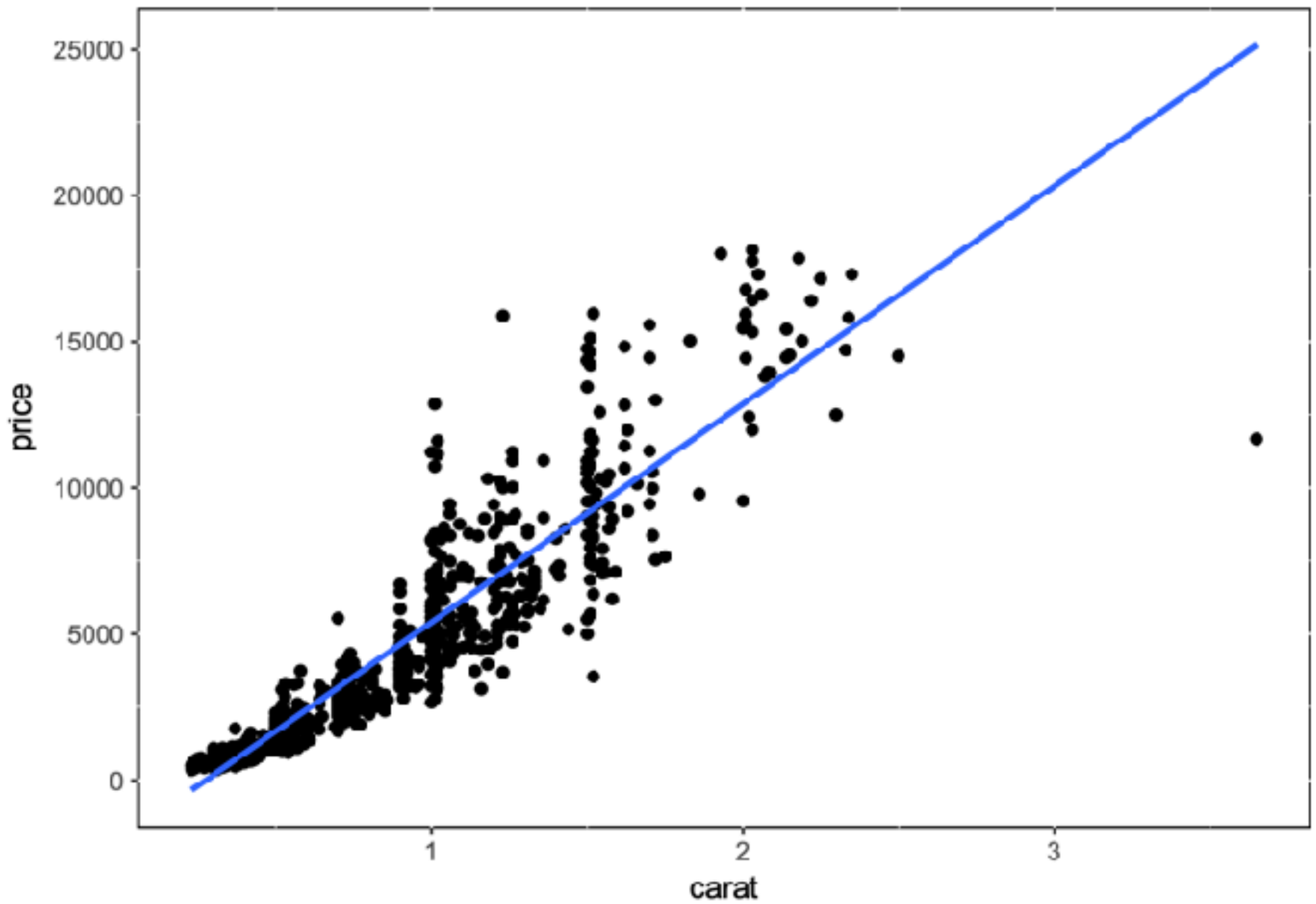
Basic example

```
set.seed(1410)
dsmall <- as.data.frame(diamonds[sample(nrow(diamonds), 1000), ])
p <- ggplot(dsmall, aes(carat, price, color=color)) +
             geom_point(size=4)
print(p)
```



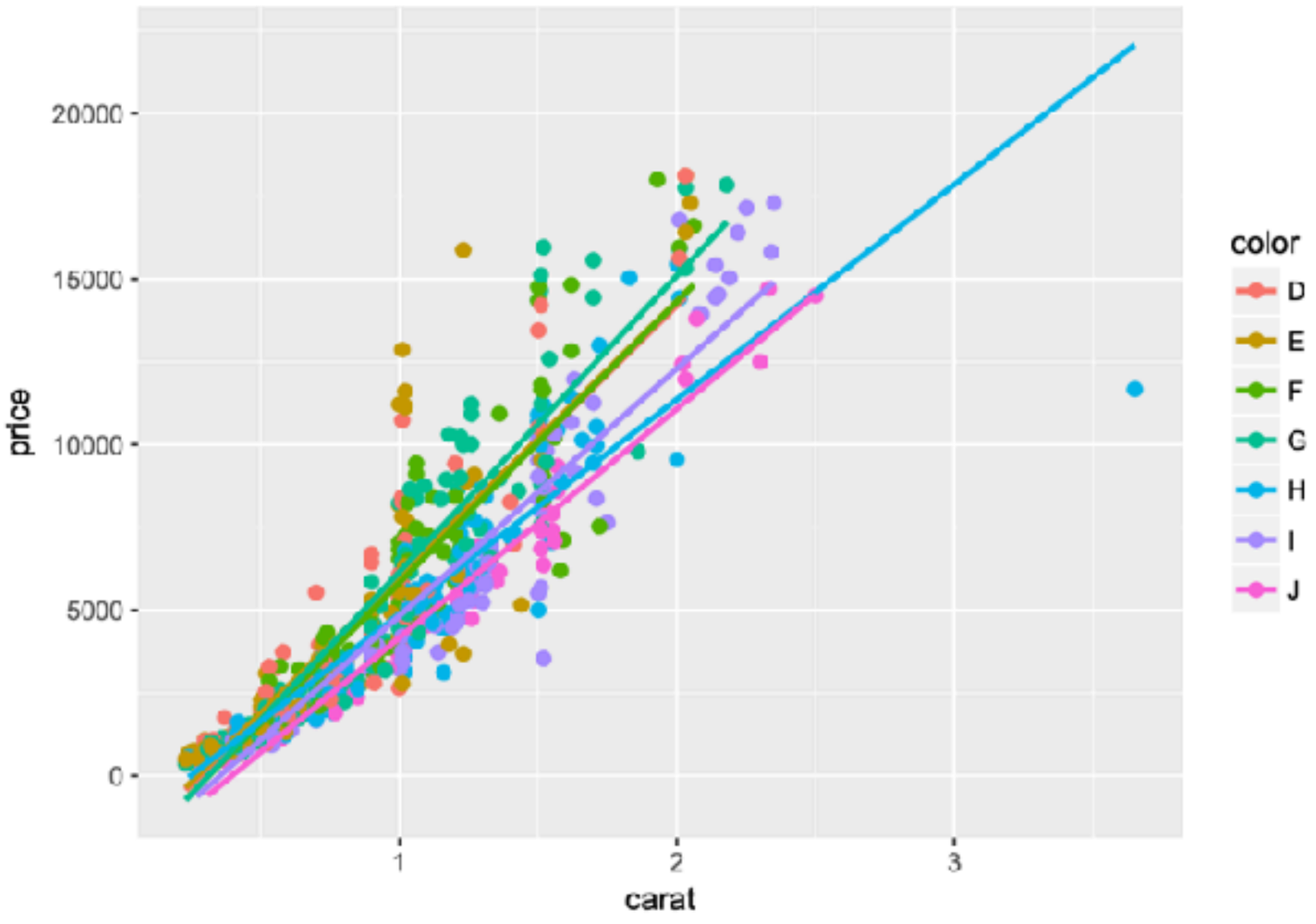
ggplot : Regression Line

```
p <- ggplot(dsmall, aes(carat, price)) + geom_point() +
            geom_smooth(method="lm", se=FALSE) +
            theme(panel.background=element_rect(fill = "white", colour = "black"))
print(p)
```



Several regression lines

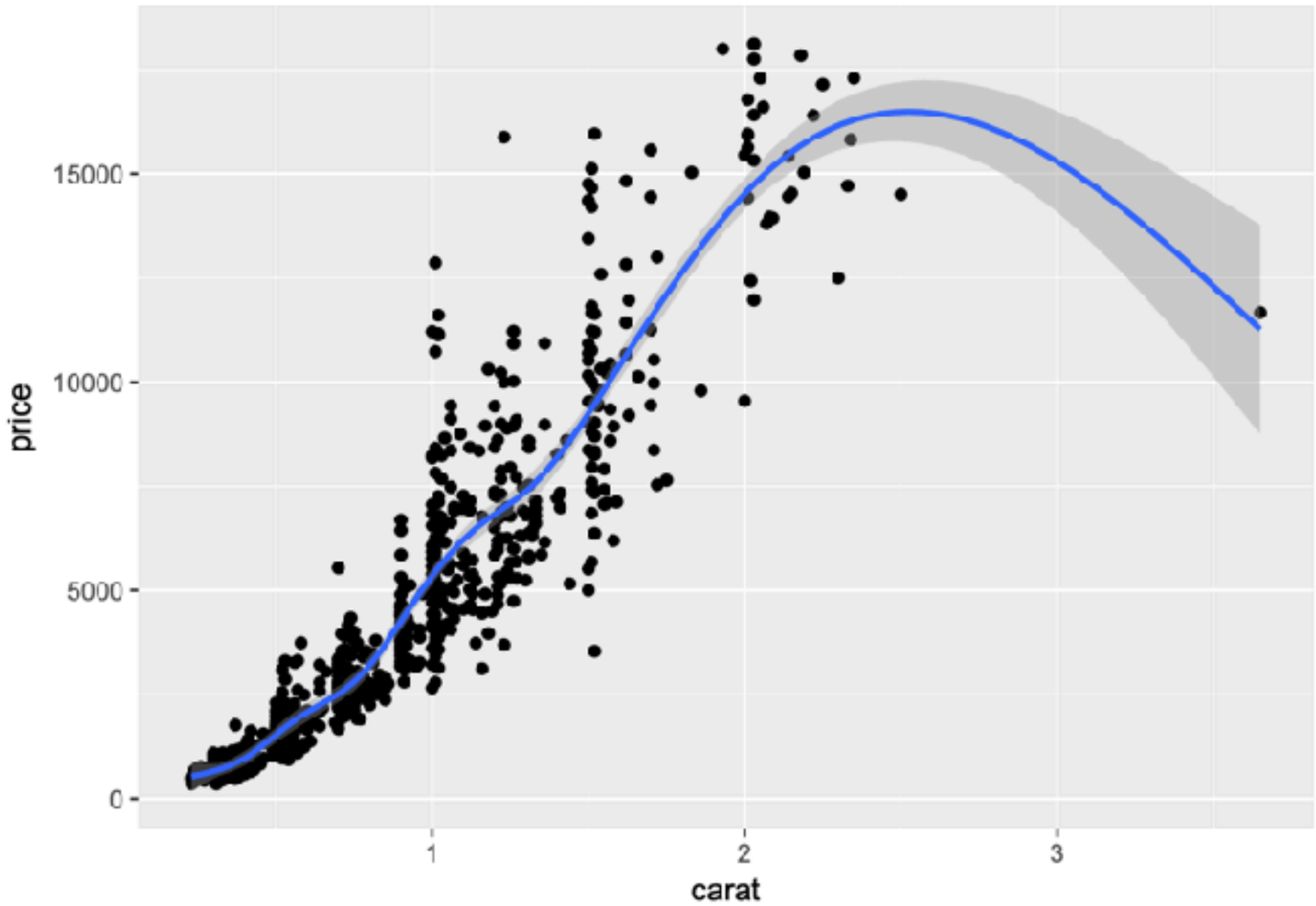
```
p <- ggplot(dsmall, aes(carat, price, group=color)) +
            geom_point(aes(color=color), size=2) +
            geom_smooth(aes(color=color), method = "lm", se=FALSE)
print(p)
```



Local regression curve (loess)

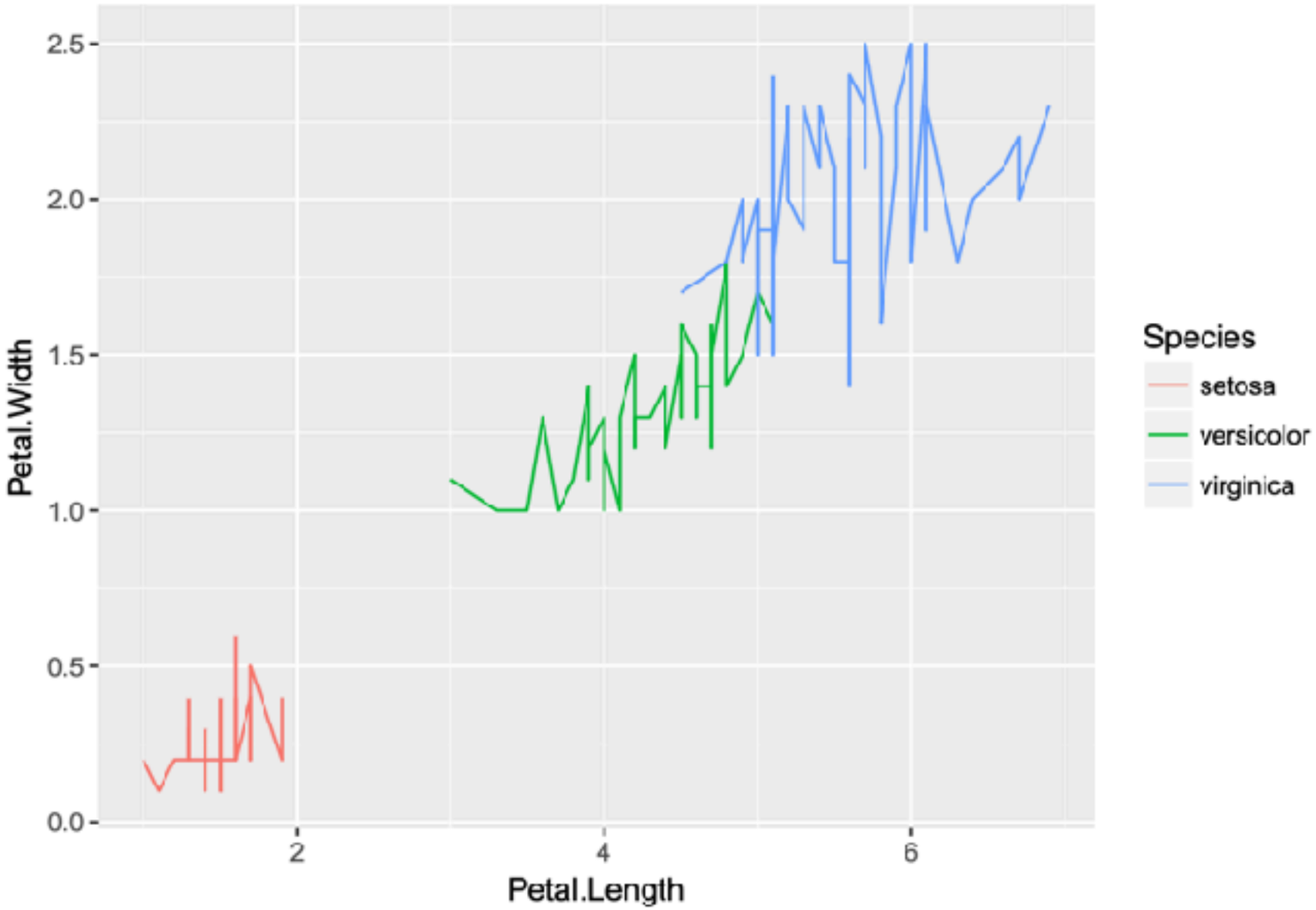
```
p <- ggplot(dsmall, aes(carat, price)) + geom_point() + geom_smooth()
print(p) # Setting se=FALSE removes error shade
```

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



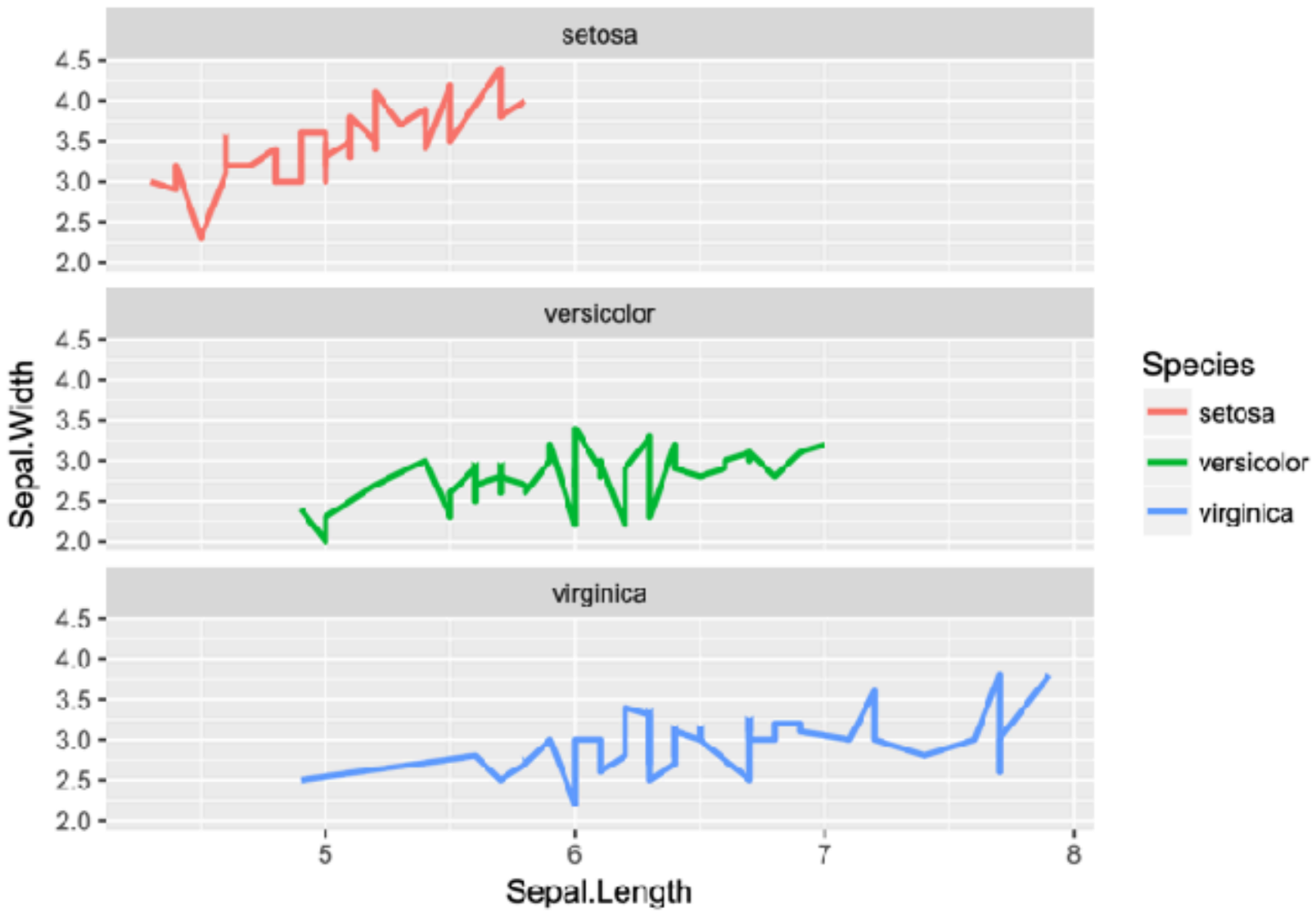
ggplot : Line Plot

```
p <- ggplot(iris, aes(Petal.Length, Petal.Width, group=Species,
                color=Species)) + geom_line()
print(p)
```



ggplot : facet

```
p <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
             geom_line(aes(color=Species), size=1) +
             facet_wrap(~Species, ncol=1)
print(p)
```



ggplot: Bar Plot

Sample Set: the following transforms the `iris` data set into a ggplot2-friendly format.

Calculate mean values for aggregates given by `Species` column in `iris` data set

```
iris_mean <- aggregate(iris[,1:4], by=list(Species=iris$Species), FUN=mean)
```

Calculate standard deviations for aggregates given by `Species` column in `iris` data set

```
iris_sd <- aggregate(iris[,1:4], by=list(Species=iris$Species), FUN=sd)
```

Reformat `iris_mean` with `melt`

```
library(reshape2) # Defines melt function
df_mean <- melt(iris_mean, id.vars=c("Species"), variable.name = "Samples", value.name="Values")
```

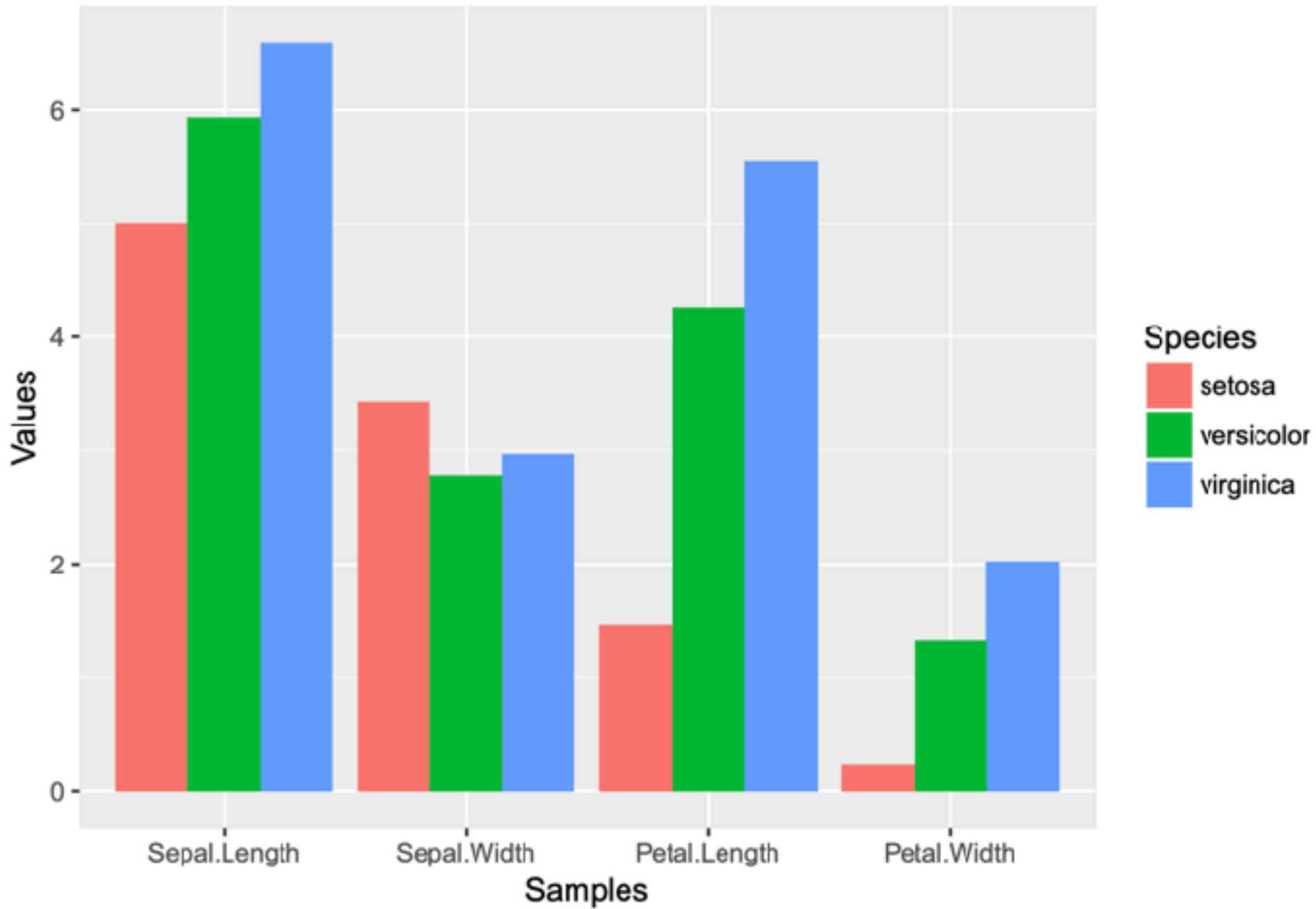
Reformat `iris_sd` with `melt`

```
df_sd <- melt(iris_sd, id.vars=c("Species"), variable.name = "Samples", value.name="Values")
```

Define standard deviation limits

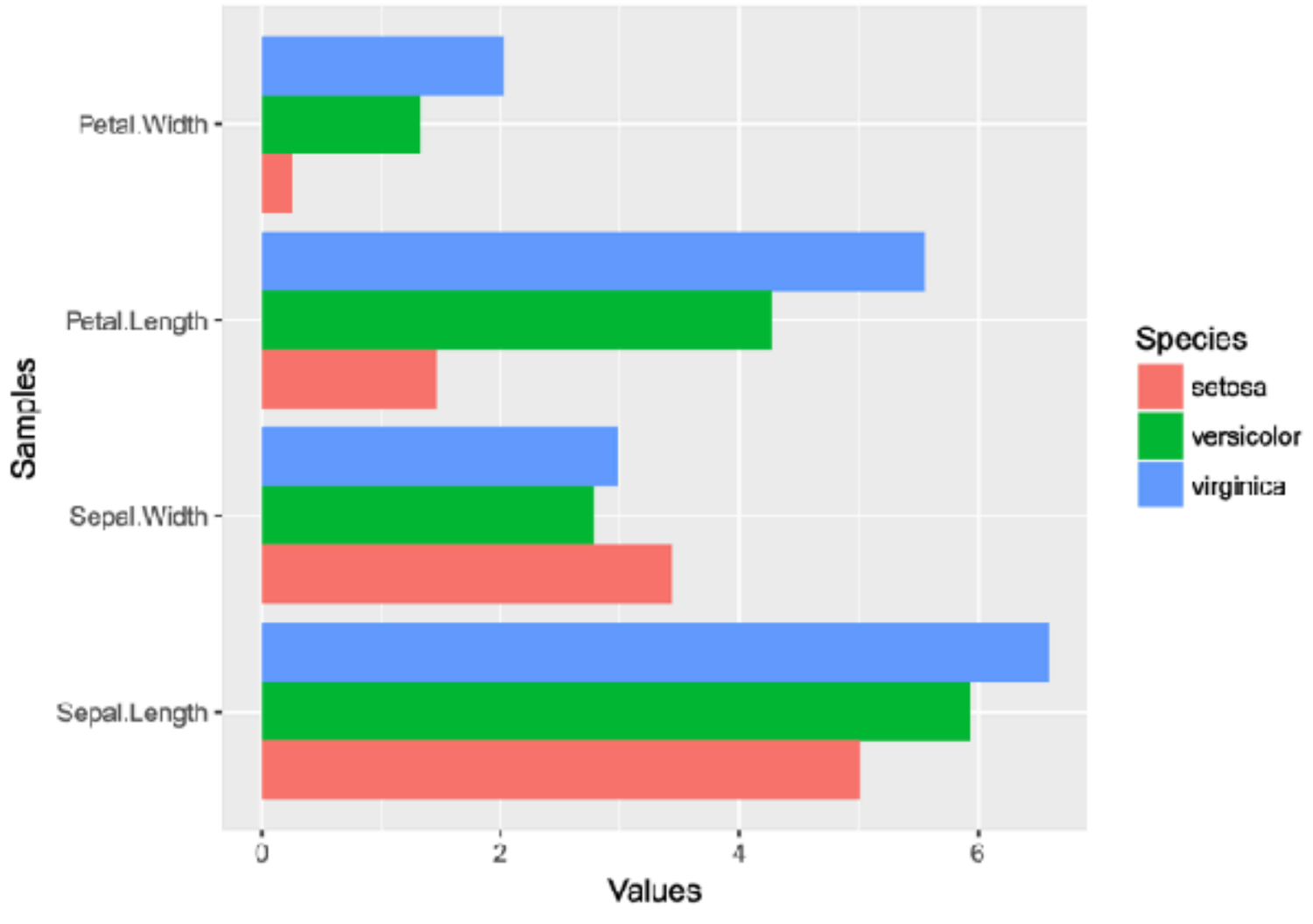
```
limits <- aes(ymax = df_mean[,"Values"] + df_sd[,"Values"], ymin=df_mean[,"Values"] - df_sd[,"Values"])
```

```r
p <- ggplot(df_mean, aes(Samples, Values, fill = Species)) +
        geom_bar(position="dodge", stat="identity")
print(p)
```



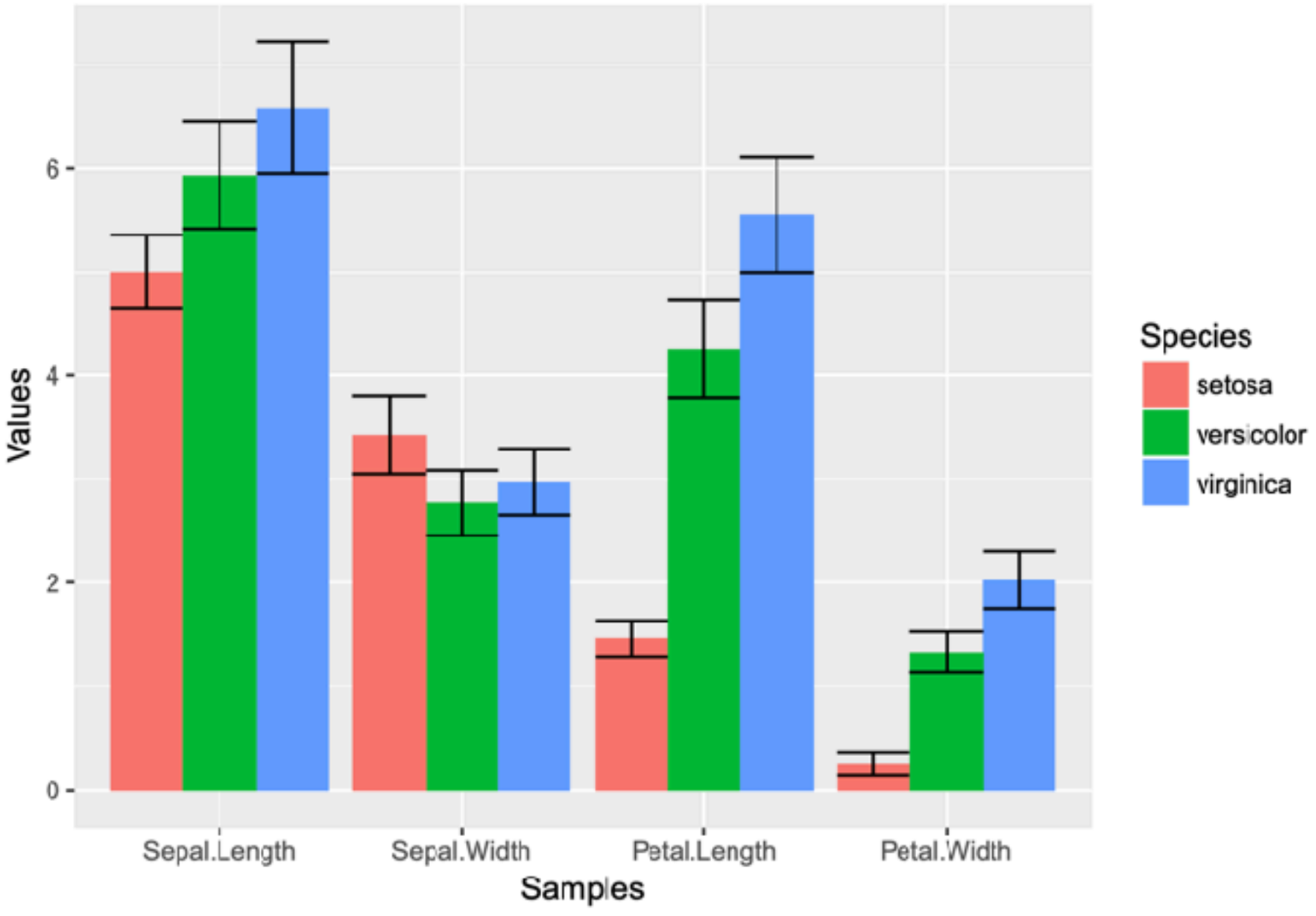
Horizontal orientation

```
p <- ggplot(df_mean, aes(Samples, Values, fill = Species)) +
            geom_bar(position="dodge", stat="identity") + coord_flip() +
            theme(axis.text.y=element_text(angle=0, hjust=1))
print(p)
```



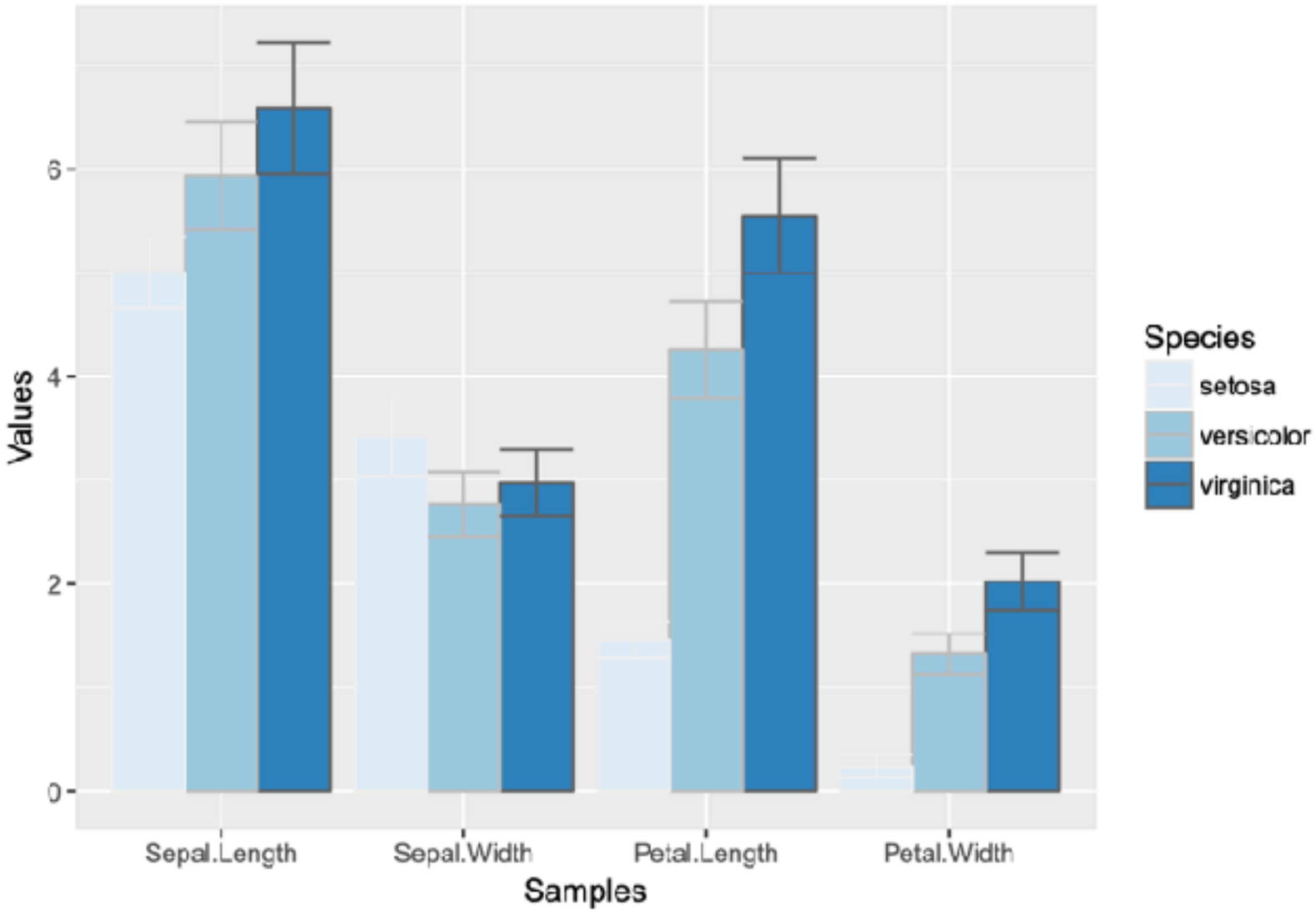
Error bars

```
p <- ggplot(df_mean, aes(Samples, Values, fill = Species)) +
        geom_bar(position="dodge", stat="identity") + geom_errorbar(limits, position="dodge")
print(p)
```



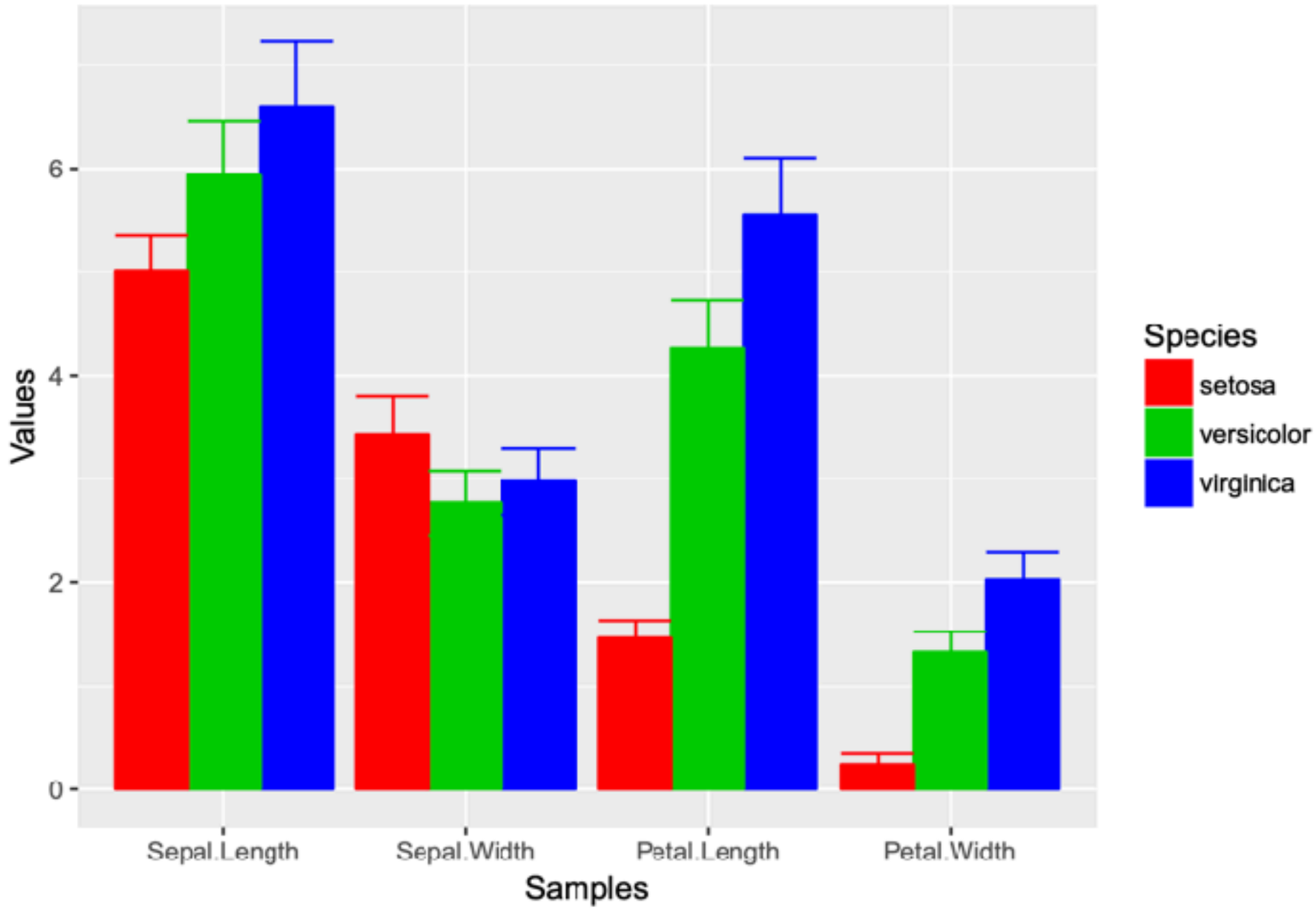
Change color

```
library(RColorBrewer)
# display.brewer.all()
p <- ggplot(df_mean, aes(Samples, Values, fill=Species, color=Species)) +
            geom_bar(position="dodge", stat="identity") + geom_errorbar(limits, position="dodge") +
            scale_fill_brewer(palette="Blues") + scale_color_brewer(palette = "Greys")
print(p)
```



Using standard color

```
p <- ggplot(df_mean, aes(Samples, Values, fill=Species, color=Species)) +
            geom_bar(position="dodge", stat="identity") + geom_errorbar(limits, position="dodge") +
            scale_fill_manual(values=c("red", "green3", "blue")) +
            scale_color_manual(values=c("red", "green3", "blue"))
print(p)
```



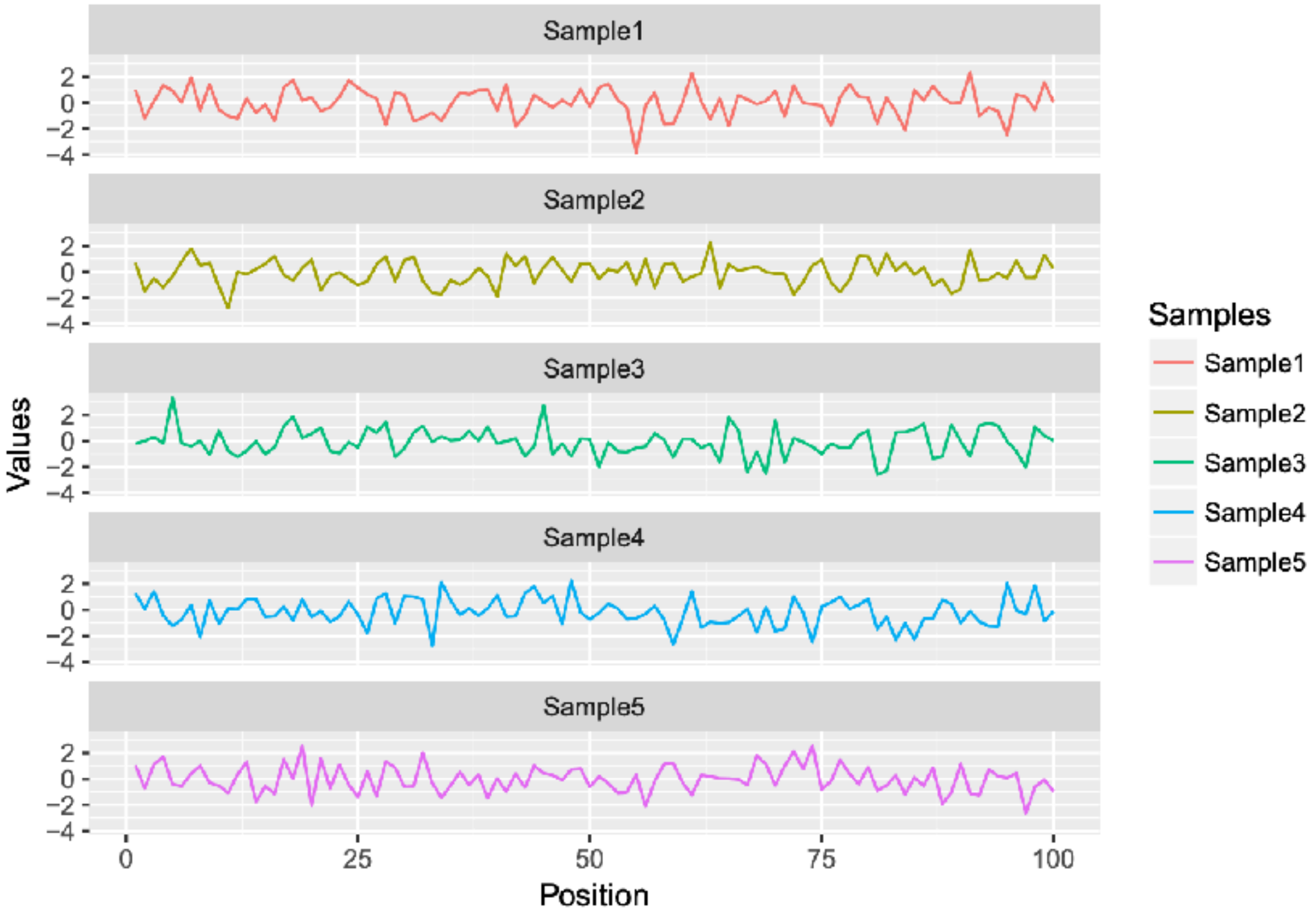
Other type of graph

Here for line plot

```
y <- matrix(rnorm(500), 100, 5, dimnames=list(paste("g", 1:100, sep=""), paste("Sample", 1:5, sep="")))
y <- data.frame(Position=1:length(y[,1]), y)
y[1:4, ] # First rows of input format expected by melt()
```

```
##    Position    Sample1    Sample2     Sample3    Sample4    Sample5
## g1        1  1.0002088  0.6850199 -0.21324932  1.27195056  1.0479301
## g2        2 -1.2024596 -1.5004962 -0.01111579  0.07584497 -0.7100662
## g3        3  0.1023678 -0.5153367  0.28564390  1.41522878  1.1084695
## g4        4  1.3294248 -1.2084007 -0.19581898 -0.42361768  1.7139697
```

```
df <- melt(y, id.vars=c("Position"), variable.name = "Samples", value.name="Values")
p <- ggplot(df, aes(Position, Values)) + geom_line(aes(color=Samples)) + facet_wrap(~Samples, ncol=1)
print(p)
```

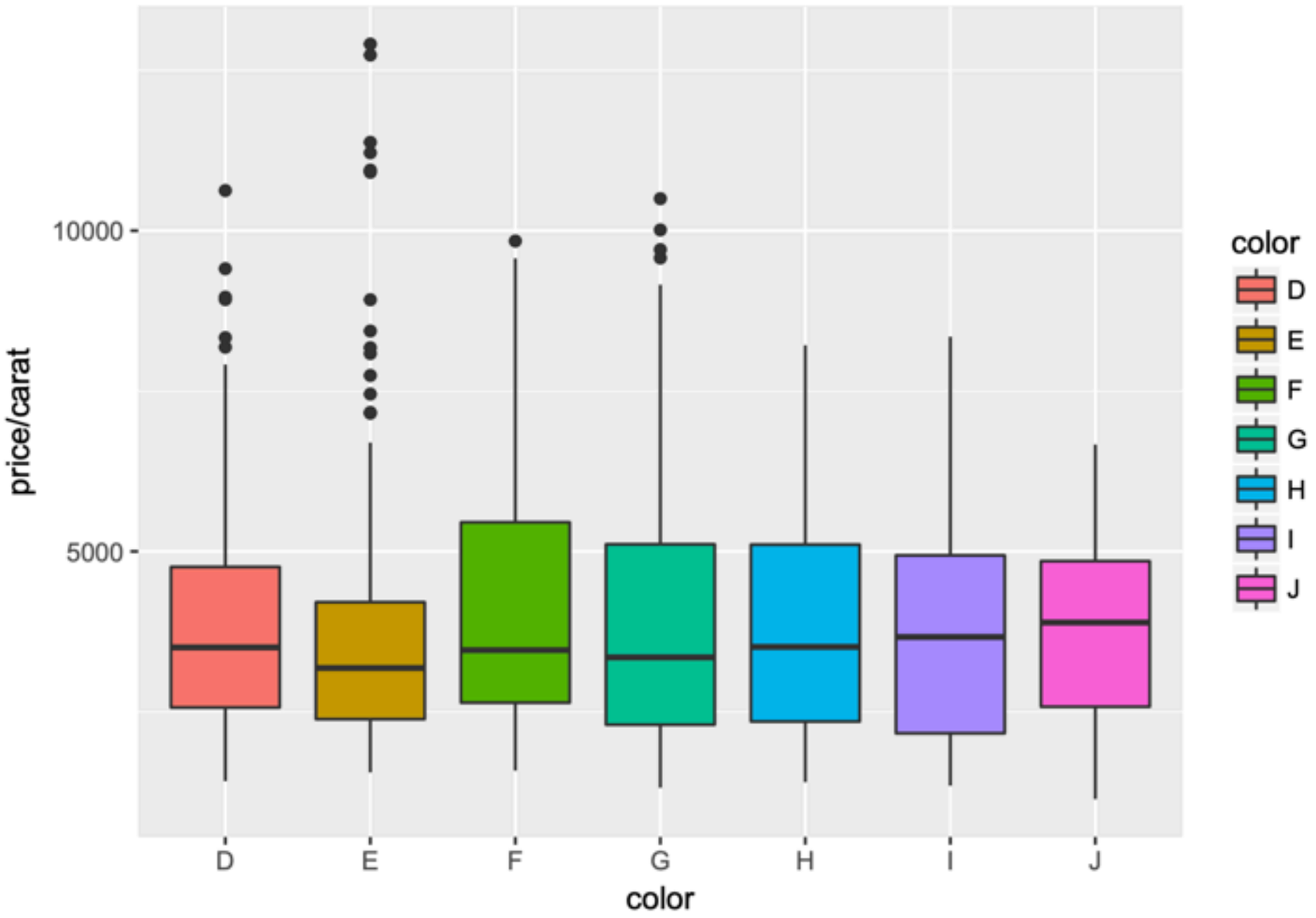

Sample1
Sample2
Sample3
Sample4
Sample5
Values
2
0
−2
−4
0
25
50
75
100
Position
Samples
Sample1
Sample2
Sample3
Sample4
Sample5

ggplot : jitter plot

```
p <- ggplot(dsmall, aes(color, price/carat)) +
           geom_jitter(alpha = I(1 / 2), aes(color=color))
print(p)
```

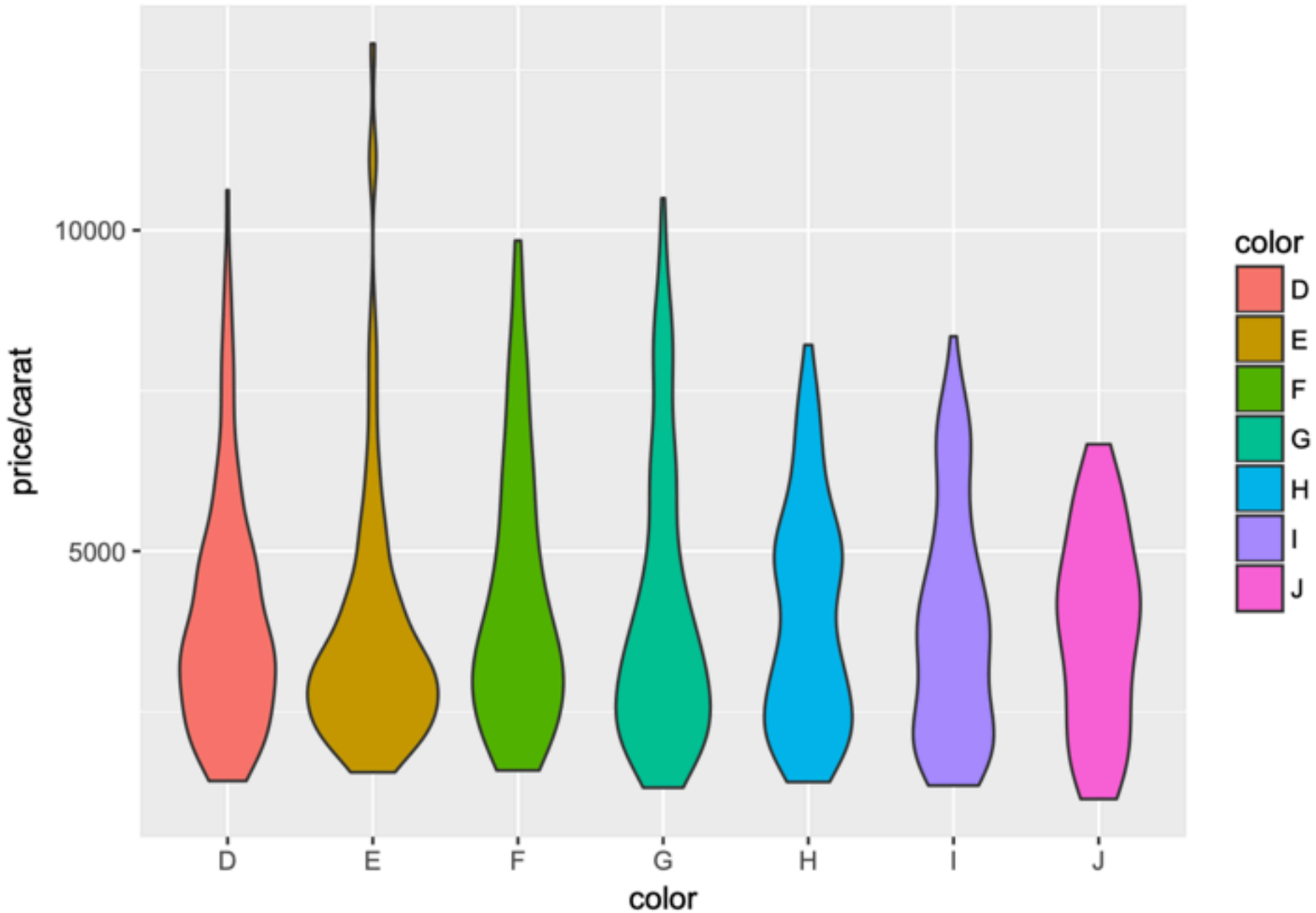
Boxplot

```
p <- ggplot(dsmall, aes(color, price/carat, fill=color)) + geom_boxplot()
print(p)
```



Violin Plot

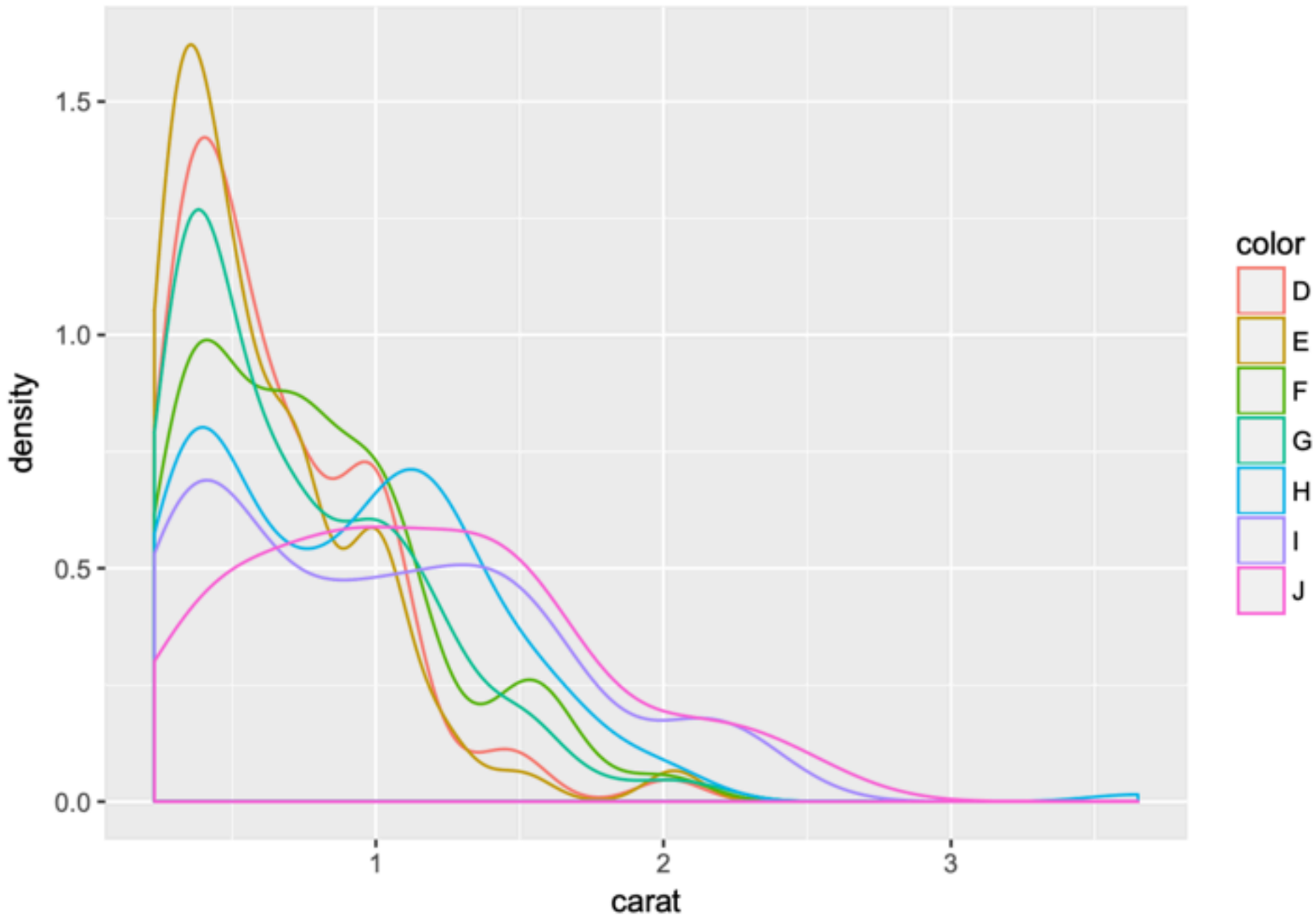
```
p <- ggplot(dsmall, aes(color, price/carat, fill=color)) + geom_violin()
print(p)
```



Density Plot

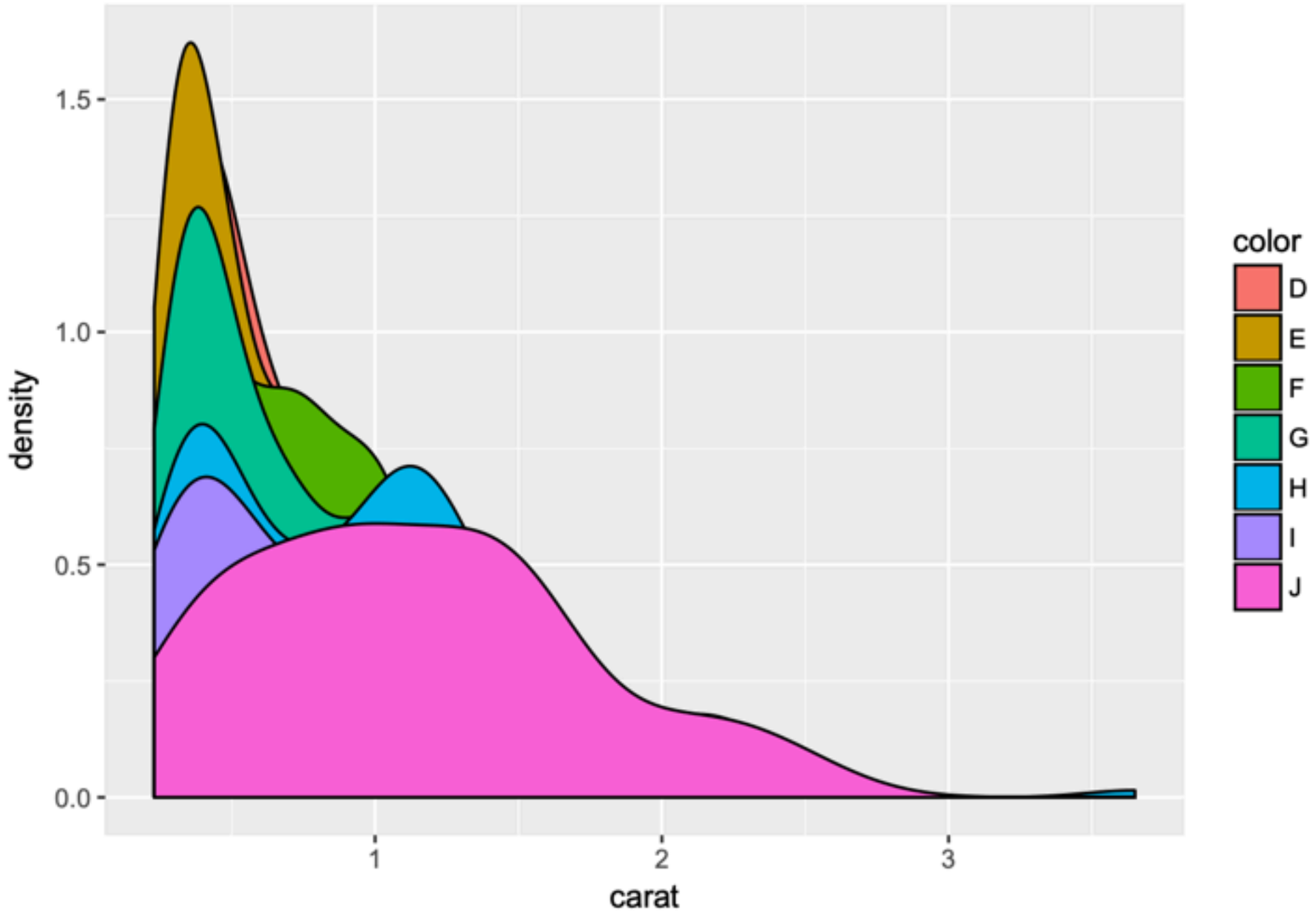
Line coloring

```
p <- ggplot(dsmall, aes(carat)) + geom_density(aes(color = color))
print(p)
```



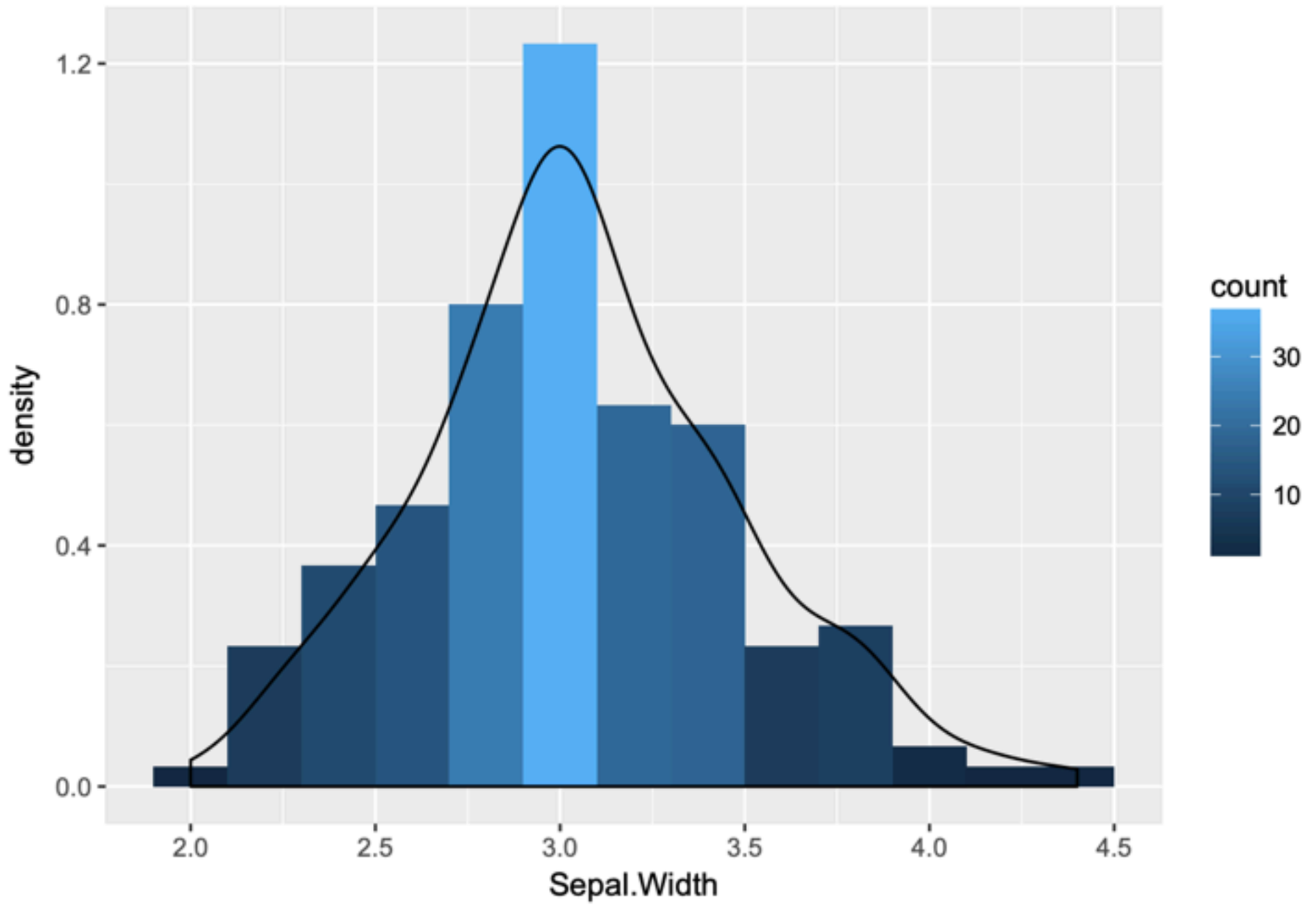
Area coloring

```
p <- ggplot(dsmall, aes(carat)) + geom_density(aes(fill = color))
print(p)
```



Histogram

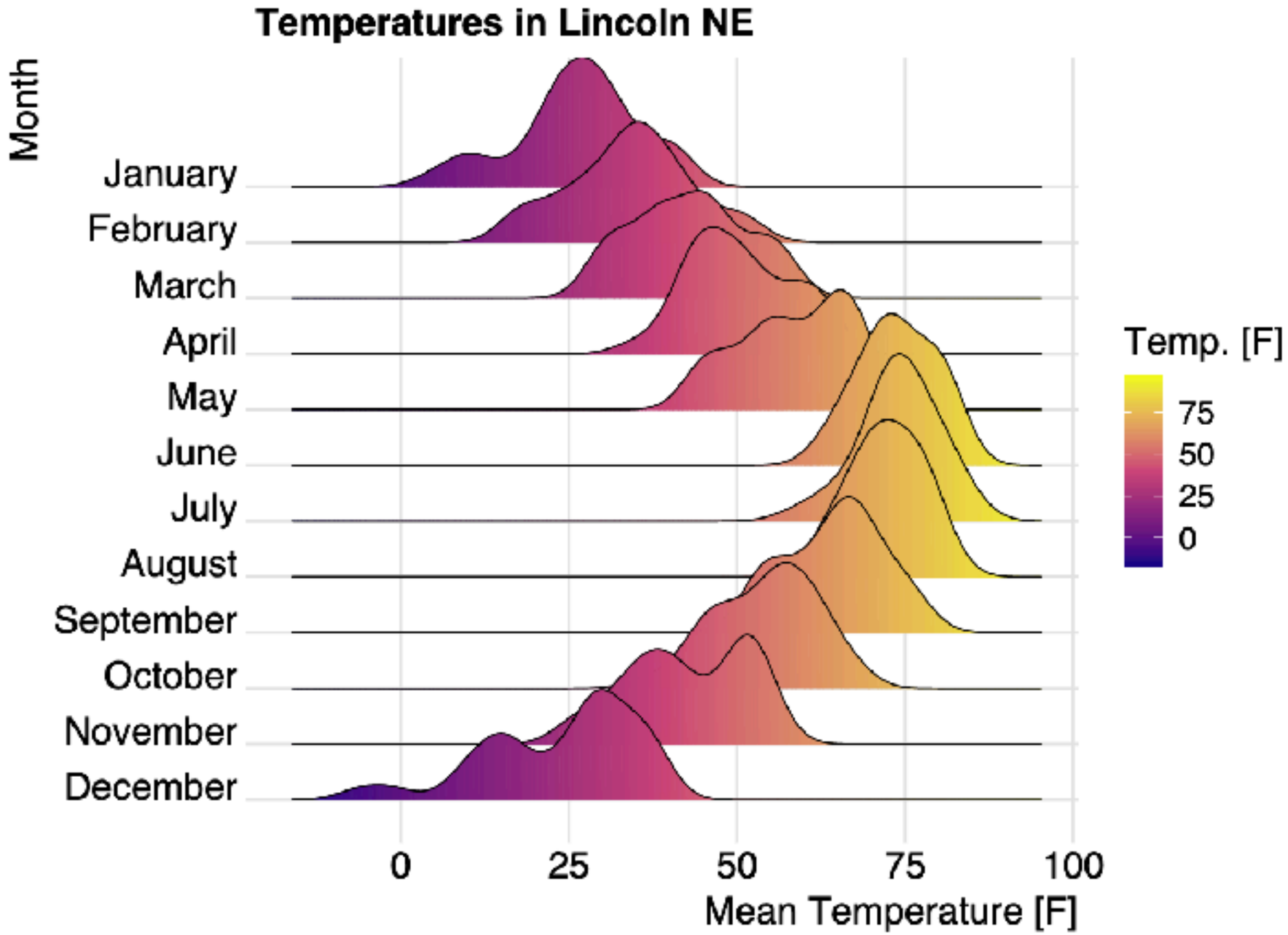
```
p <- ggplot(iris, aes(x=Sepal.Width)) + geom_histogram(aes(y = ..density..,
            fill = ..count..), binwidth=0.2) + geom_density()
print(p)
```



Ridgeline plot

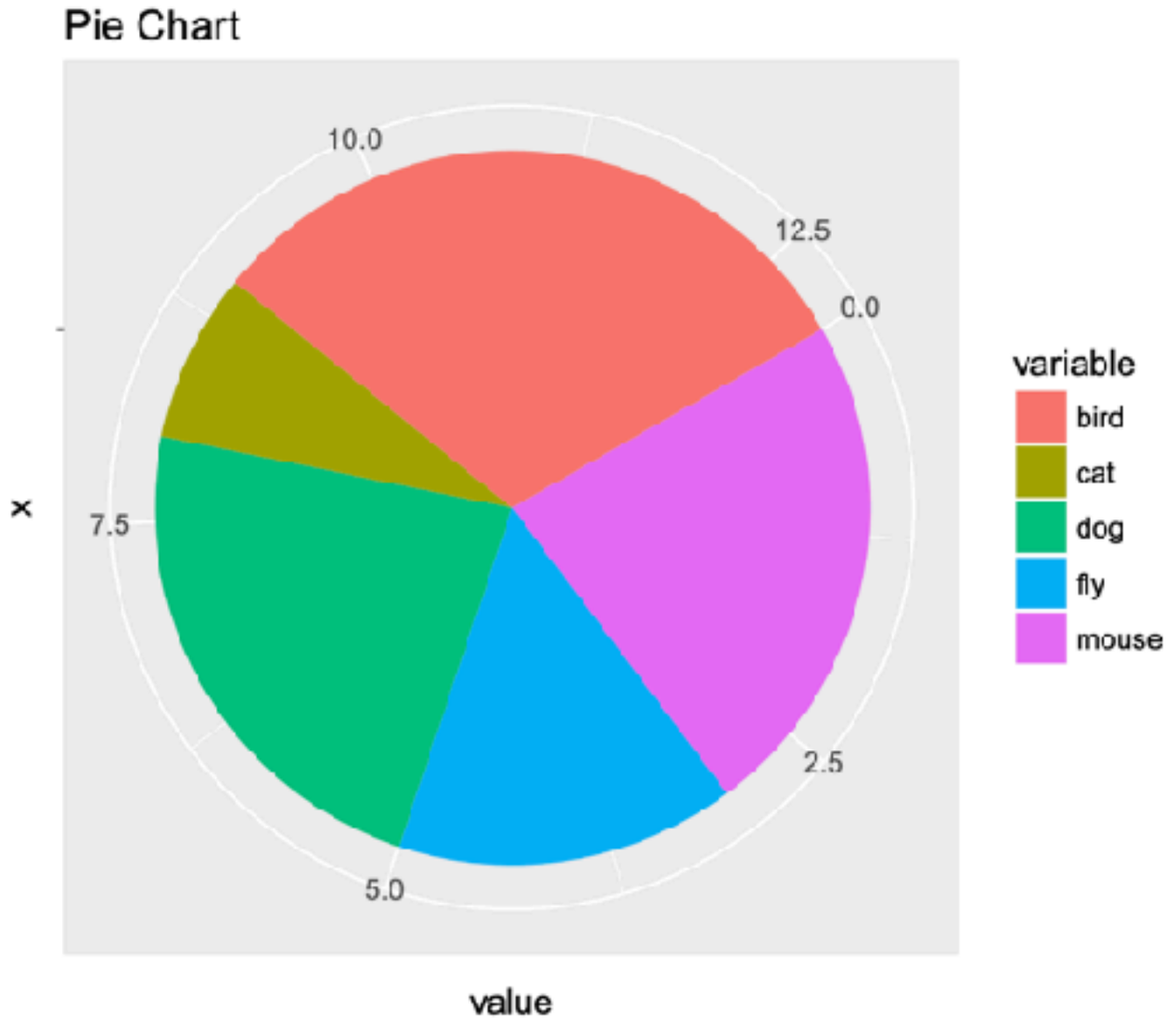
```
library(ggplot2)
library(ggridges)
theme_set(theme_ridges())
```

```
ggplot(
  lincoln_weather,
  aes(x = `Mean Temperature [F]`, y = `Month`)
  ) +
geom_density_ridges_gradient(
  aes(fill = ..x..), scale = 3, size = 0.3
  ) +
scale_fill_gradientn(
  colours = c("#0D0887FF", "#CC4678FF", "#F0F921FF"),
  name = "Temp. [F]"
  )+
labs(title = 'Temperatures in Lincoln NE')
```


Temperatures in Lincoln NE
Month
January
February
March
April
May
June
July
August
September
October
November
December
0
25
50
75
100
Mean Temperature [F]
Temp. [F]
75
50
25
0

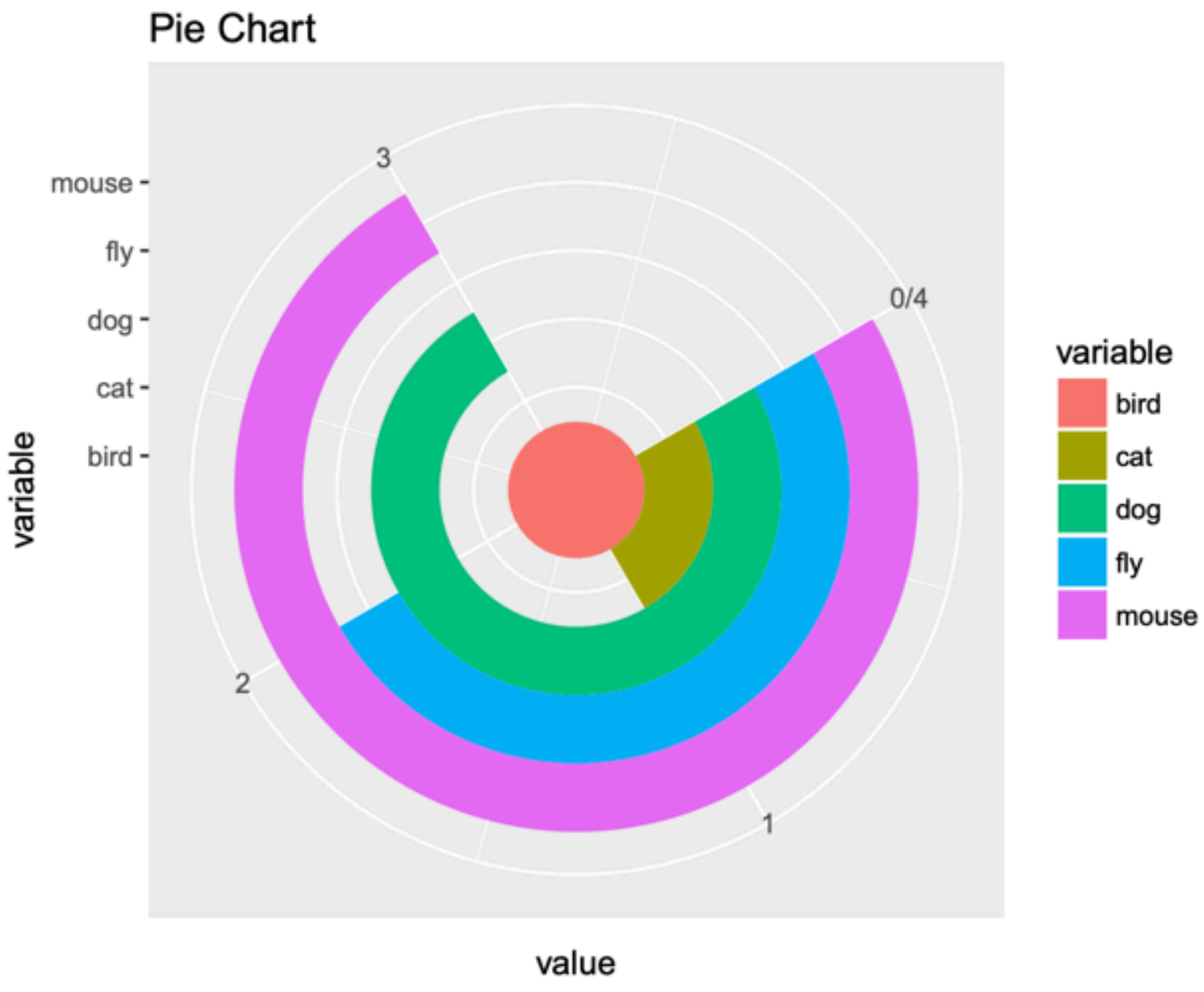
Pie chart

```
df <- data.frame(variable=rep(c("cat", "mouse", "dog", "bird", "fly")),
                 value=c(1,3,3,4,2))
p <- ggplot(df, aes(x = "", y = value, fill = variable)) +
             geom_bar(width = 1, stat="identity") +
             coord_polar("y", start=pi / 3) + ggtitle("Pie Chart")
print(p)
```



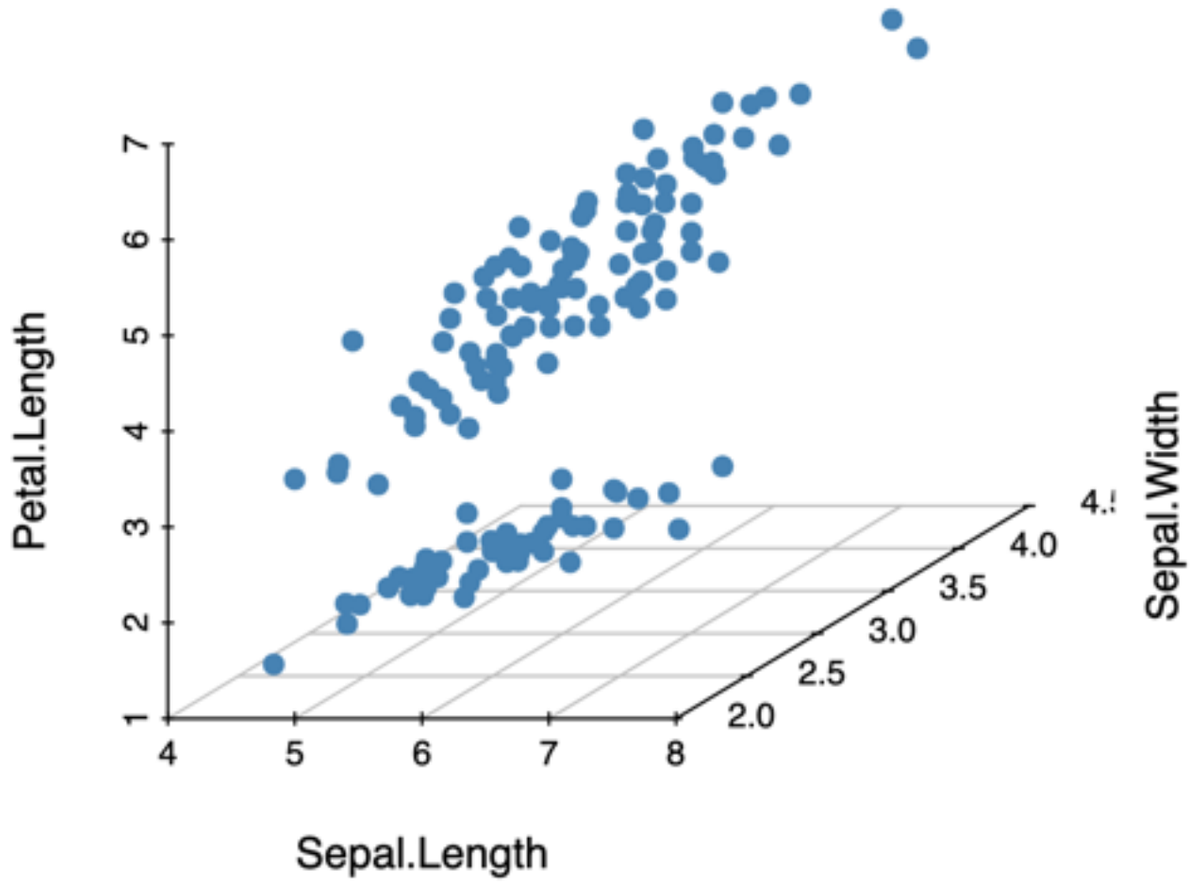
Wind Rose Pie Chart

```
p <- ggplot(df, aes(x = variable, y = value, fill = variable)) +
       geom_bar(width = 1, stat="identity") + coord_polar("y", start=pi / 3) +
       ggtitle("Pie Chart")
print(p)
```



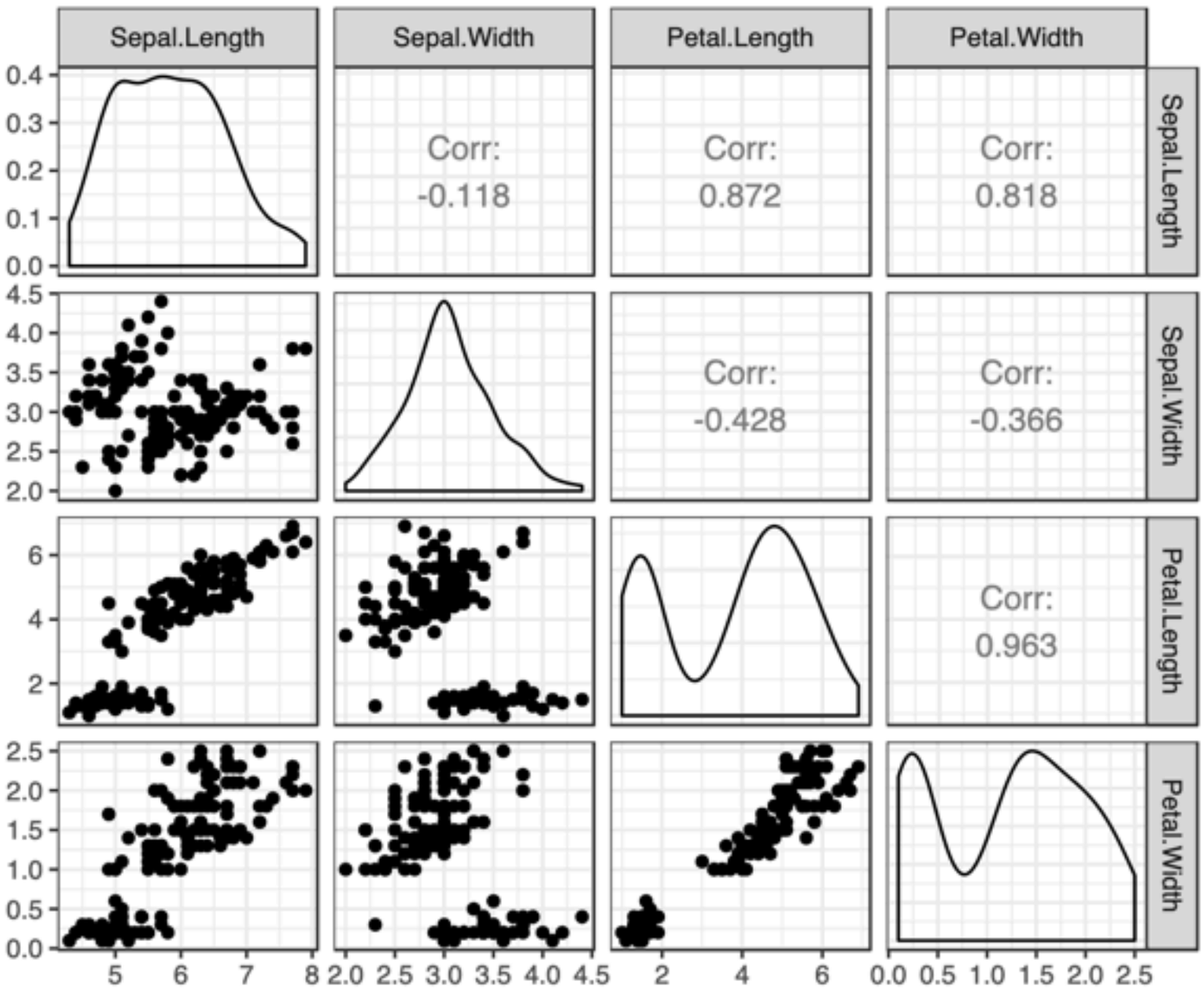
3d scatter plot

```
library(scatterplot3d)
scatterplot3d(
  iris[,1:3], pch = 19, color = "steelblue",
   grid = TRUE, box = FALSE,
   mar = c(3, 3, 0.5, 3)
  )
```



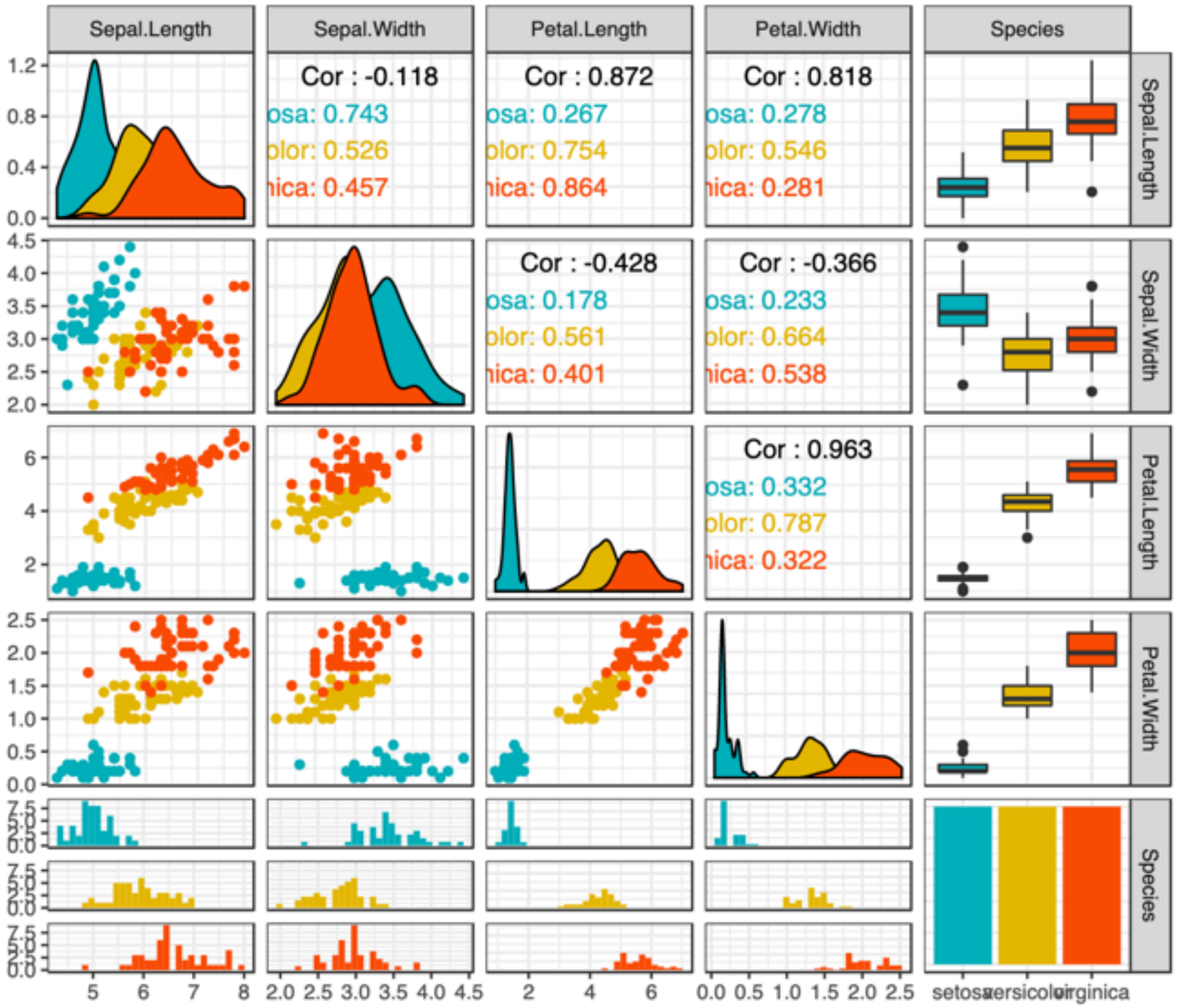
Scatter Plot Matrix

```
library(GGally)
library(ggplot2)
ggpairs(iris[,-5])+ theme_bw()
```

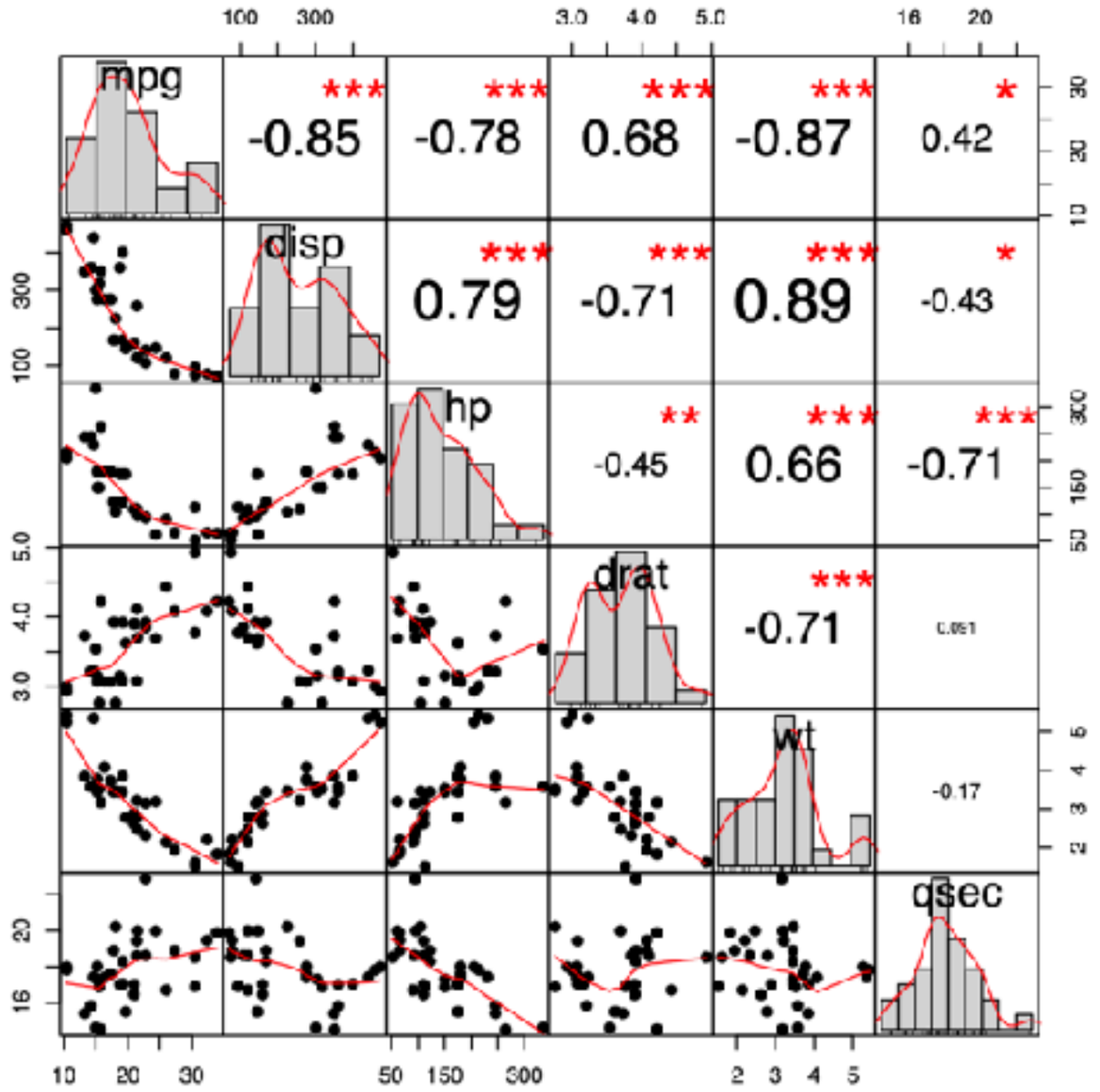


ggpairs

```
p <- ggpairs(iris, aes(color = Species))+ theme_bw()
# Change color manually.
# Loop through each plot changing relevant scales
for(i in 1:p$nrow) {
  for(j in 1:p$ncol){
    p[i,j] <- p[i,j] +
        scale_fill_manual(values=c("#00AFBB", "#E7B800", "#FC4E07")) +
        scale_color_manual(values=c("#00AFBB", "#E7B800", "#FC4E07"))
  }
}
p
```


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
Cor : -0.118
Cor : 0.872
Cor : 0.818
Cor : -0.428
Cor : -0.366
Cor : 0.963
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
setosa
versicolor
virginica

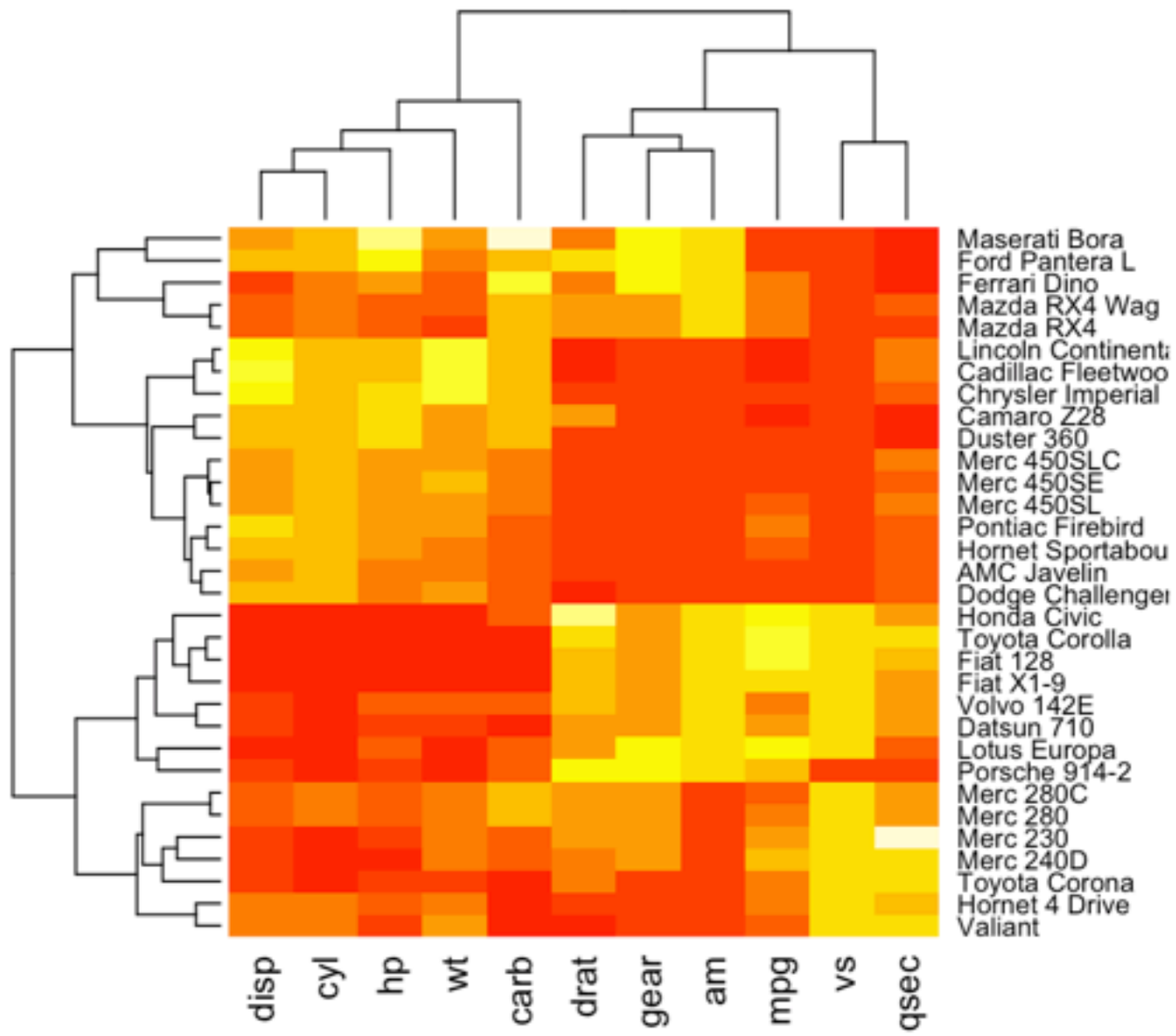
```r
# install.packages("PerformanceAnalytics")
library("PerformanceAnalytics")
my_data <- mtcars[, c(1,3,4,5,6,7)]
chart.Correlation(my_data, histogram=TRUE, pch=19)
```



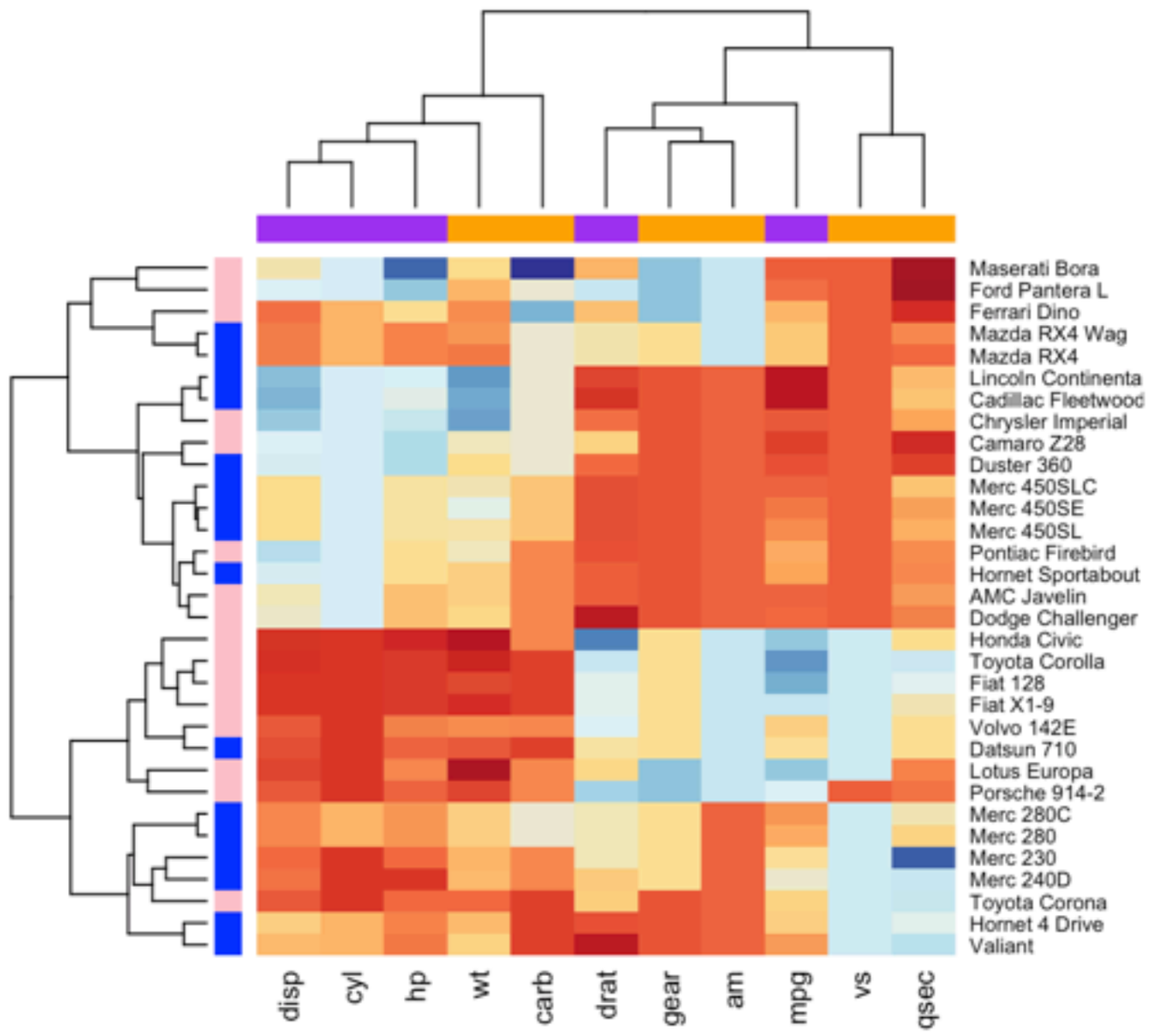
Heatmap

```
df <- scale(mtcars)
```

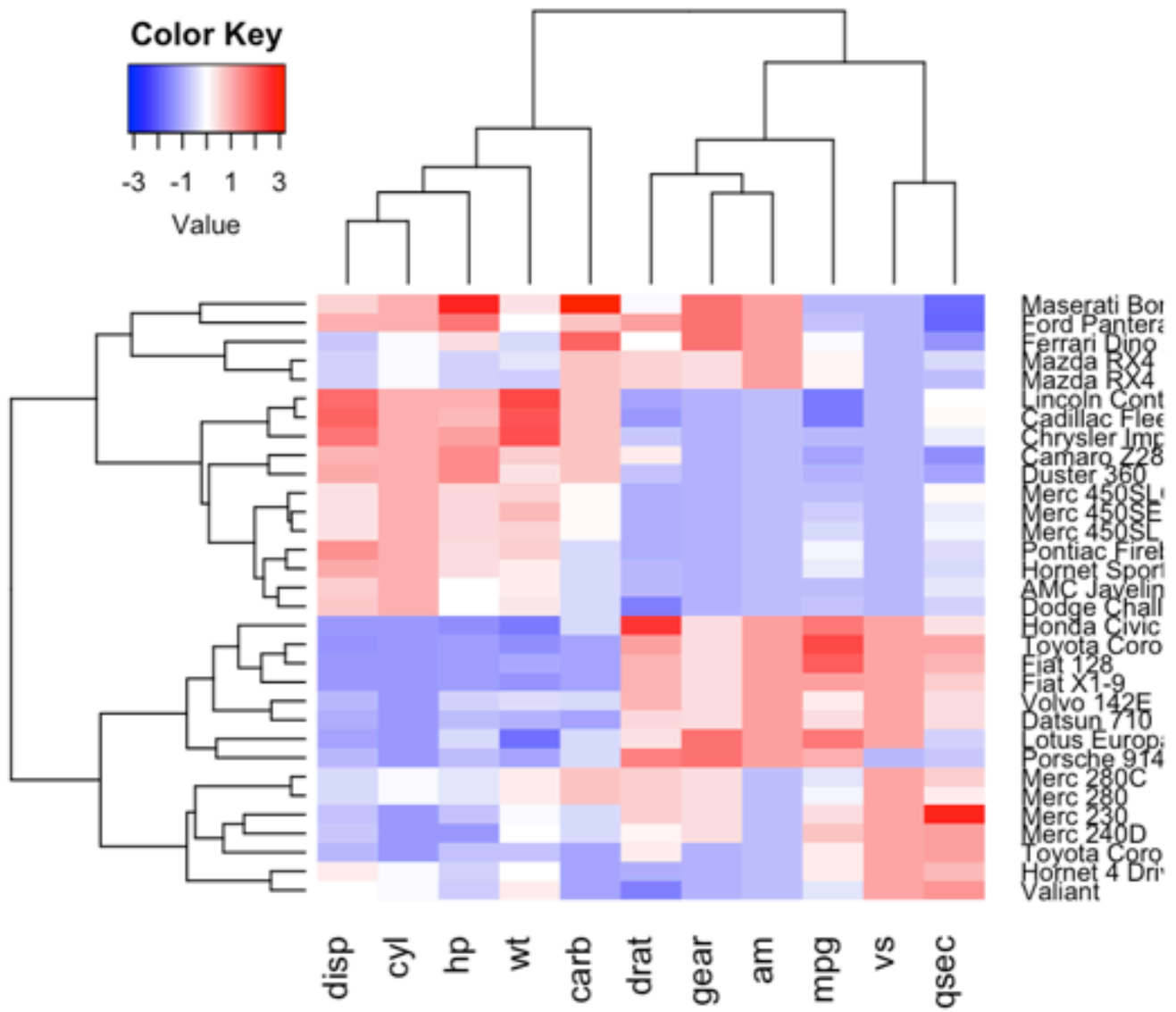
```
# Default plot
heatmap(df, scale = "none")
```



```r
# Use RColorBrewer color palette names
library("RColorBrewer")
col <- colorRampPalette(brewer.pal(10, "RdYlBu"))(256)
heatmap(df, scale = "none", col =  col,
        RowSideColors = rep(c("blue", "pink"), each = 16),
        ColSideColors = c(rep("purple", 5), rep("orange", 6)))
```



```r
# install.packages("gplots")
library("gplots")
heatmap.2(df, scale = "none", col = bluered(100),
          trace = "none", density.info = "none")
```



Workshop 2.2 Explore your data

From data you selected, use **basic plot** or **ggplot** to explore your data in each variables.

- **Continuous variables** use **Histogram**

- **Category variables** use **Bar Plot**

- Co-variation between **category and continuous variables** using **Box Plot**

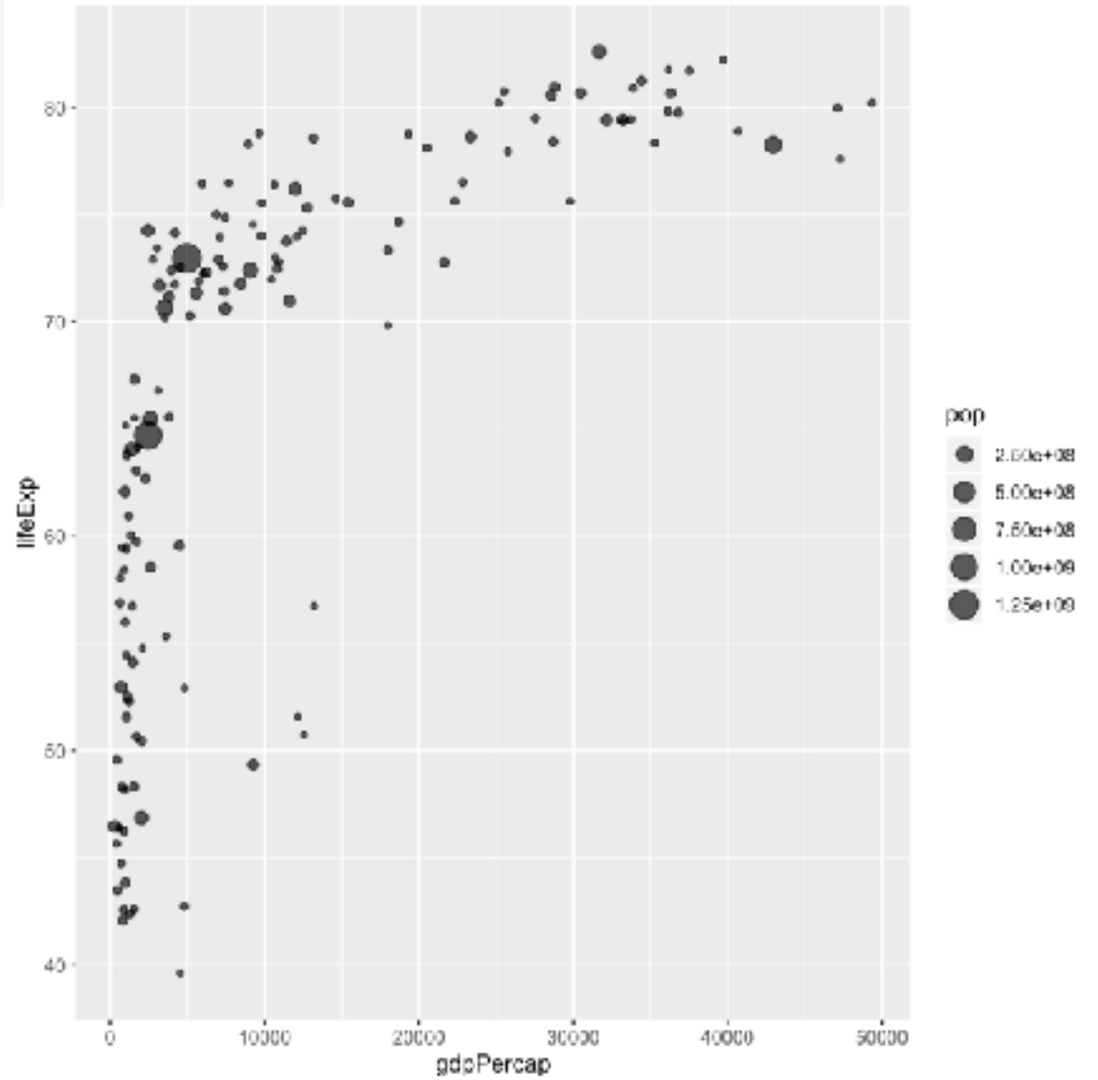
- Co-variation between **continuous variables** using **Scatter Plot**

- Co-variation between **category variables** using **Heat Map**

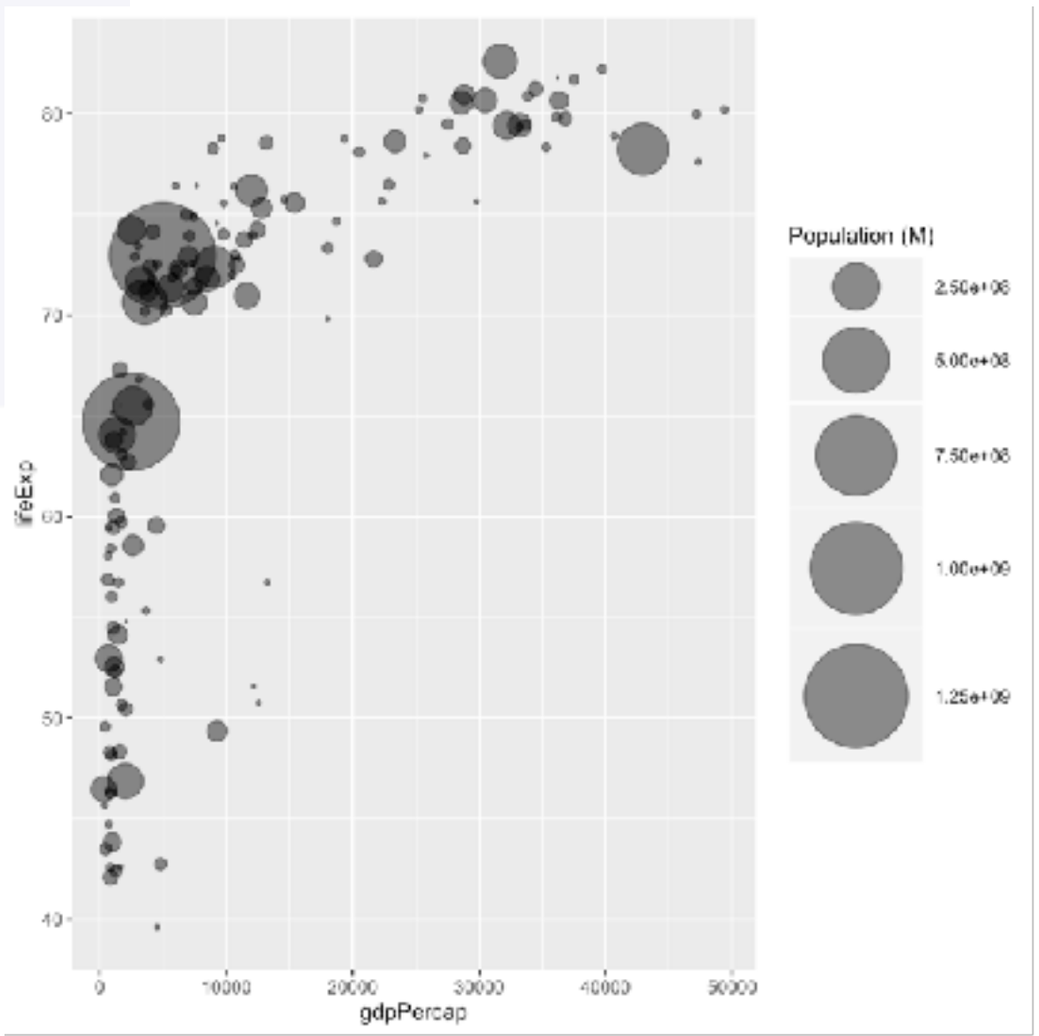
Interactive Chart

ggplot : bubble chart

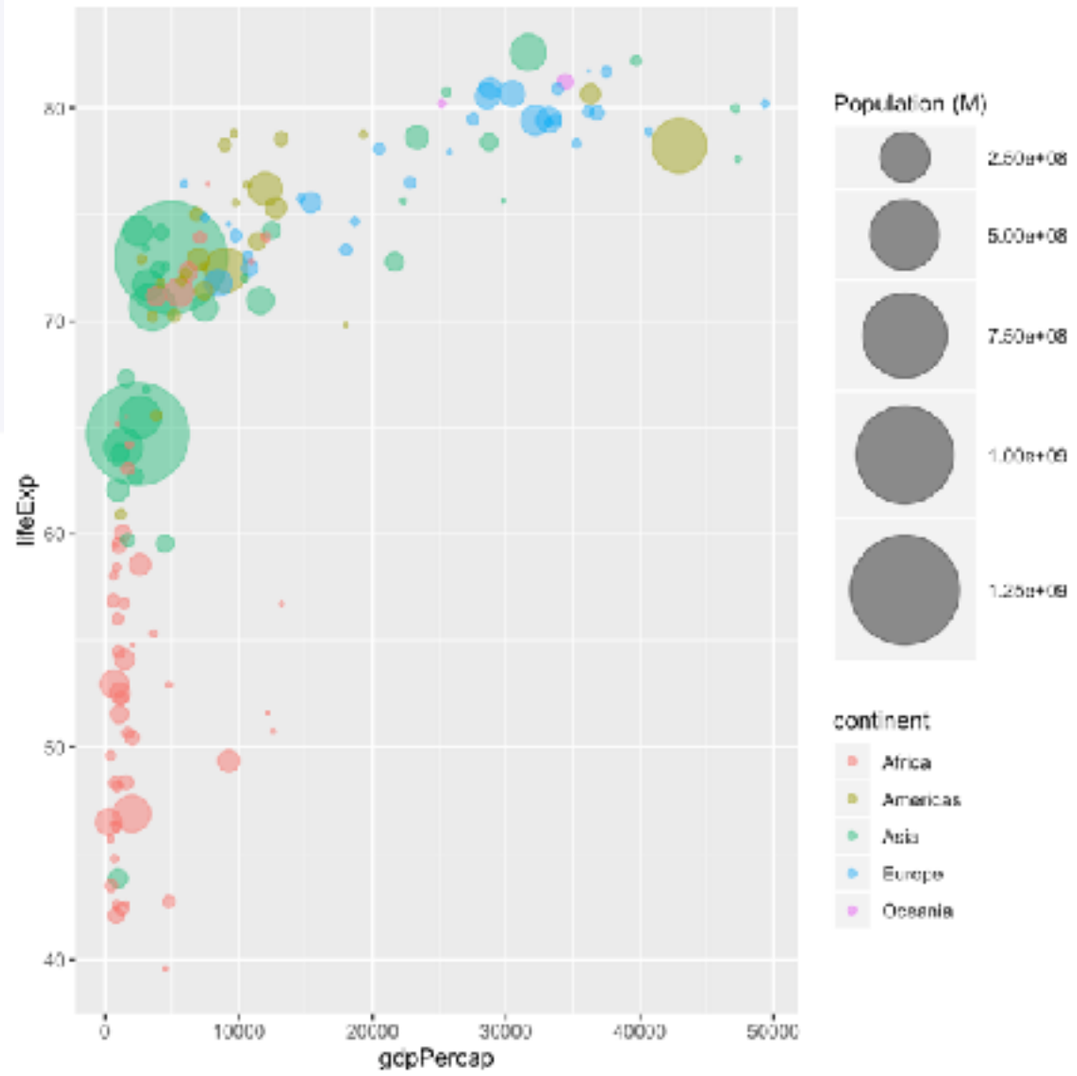
```r
# Libraries
library(ggplot2)
library(dplyr)

# The dataset is provided in the gapminder library
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

# Most basic bubble plot
ggplot(data, aes(x=gdpPercap, y=lifeExp, size = pop)) +
    geom_point(alpha=0.7)
```



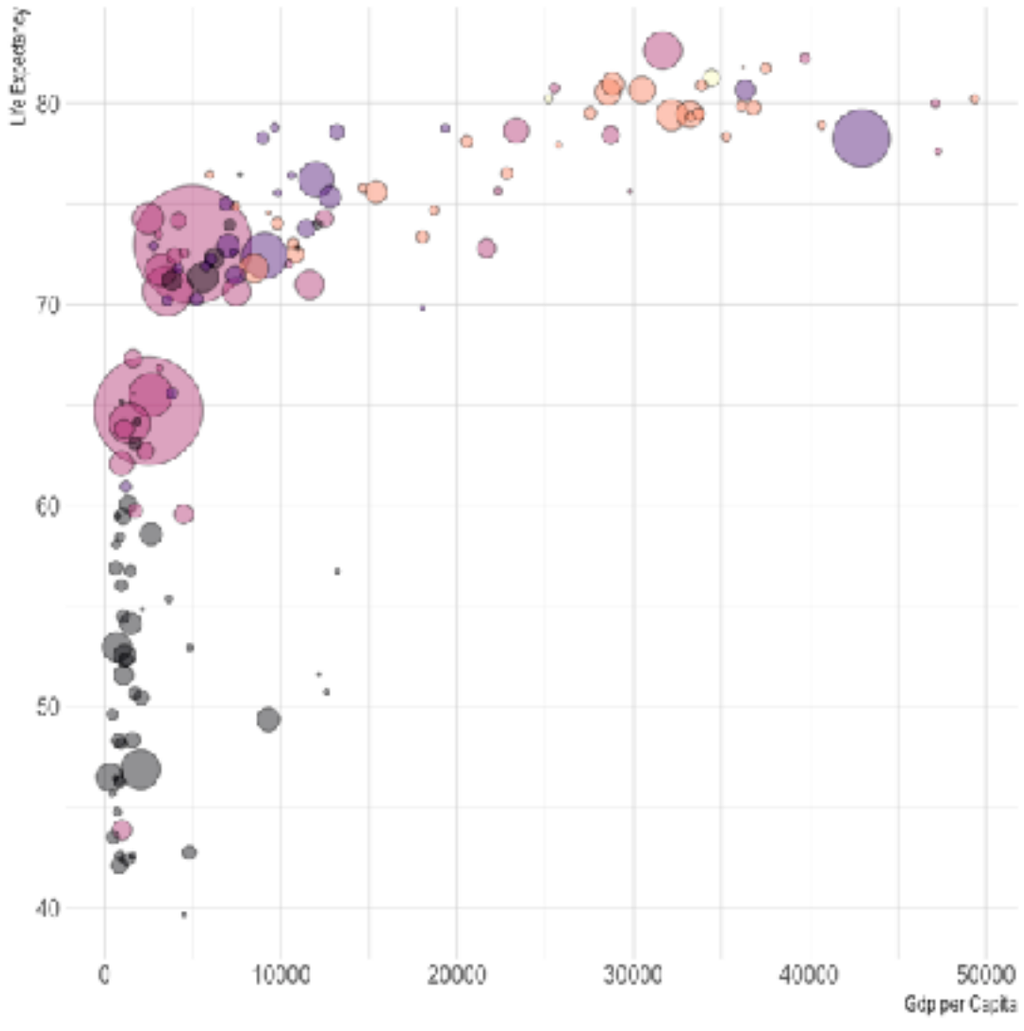
```r
# Libraries
library(ggplot2)
library(dplyr)

# The dataset is provided in the gapminder library
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

# Most basic bubble plot
data %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%
  ggplot(aes(x=gdpPercap, y=lifeExp, size = pop)) +
    geom_point(alpha=0.5) +
    scale_size(range = c(.1, 24), name="Population (M)")
```



```r
# Libraries
library(ggplot2)
library(dplyr)

# The dataset is provided in the gapminder library
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

# Most basic bubble plot
data %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%
  ggplot(aes(x=gdpPercap, y=lifeExp, size = pop)) +
    geom_point(alpha=0.5) +
    scale_size(range = c(.1, 24), name="Population (M)")
```

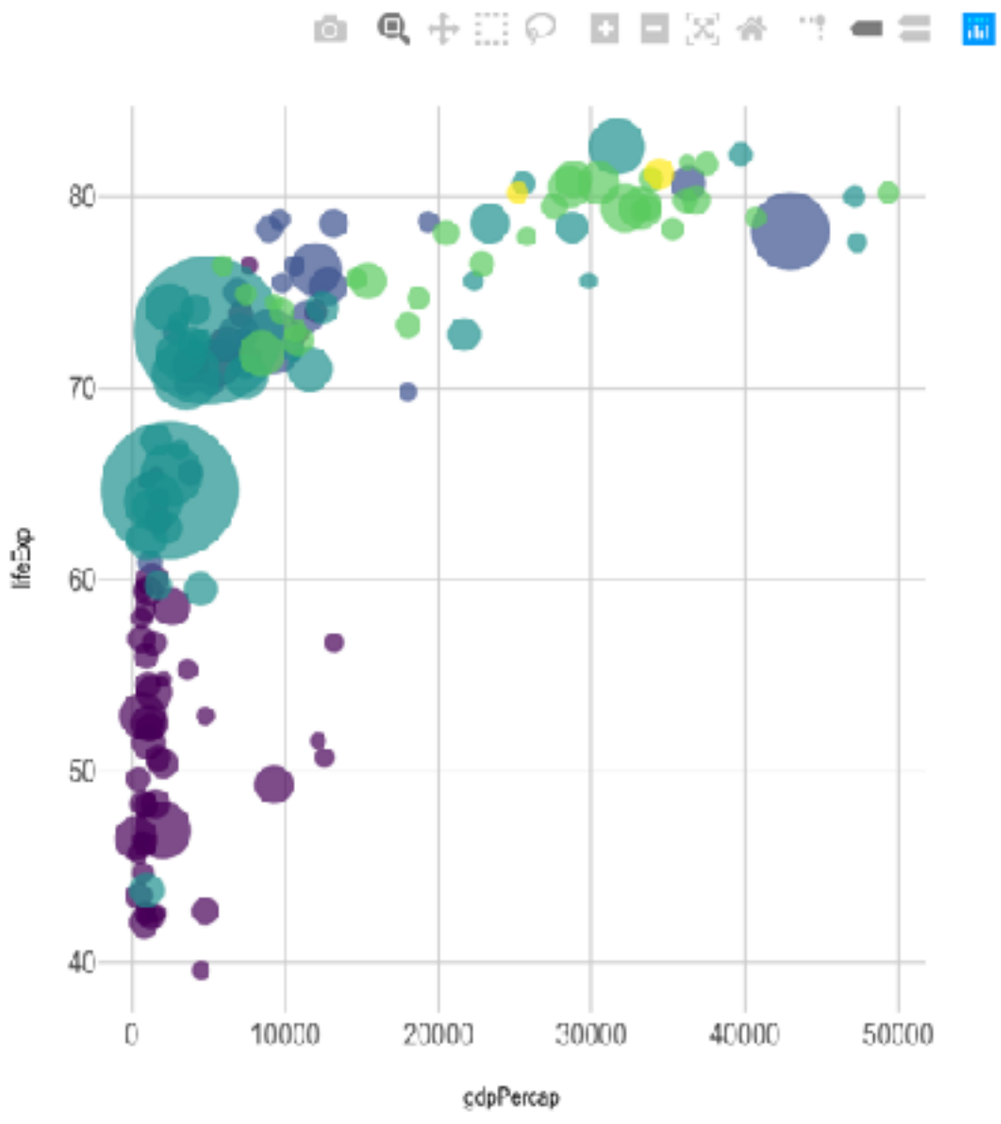


```r
# Libraries
library(ggplot2)
library(dplyr)
library(hrbrthemes)
library(viridis)

# The dataset is provided in the gapminder library
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

# Most basic bubble plot
data %>%
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%
  ggplot(aes(x=gdpPercap, y=lifeExp, size=pop, fill=continent)) +
    geom_point(alpha=0.5, shape=21, color="black") +
    scale_size(range = c(.1, 24), name="Population (M)") +
    scale_fill_viridis(discrete=TRUE, guide=FALSE, option="A") +
    theme_ipsum() +
    theme(legend.position="bottom") +
    ylab("Life Expectancy") +
    xlab("Gdp per Capita") +
    theme(legend.position = "none")
```



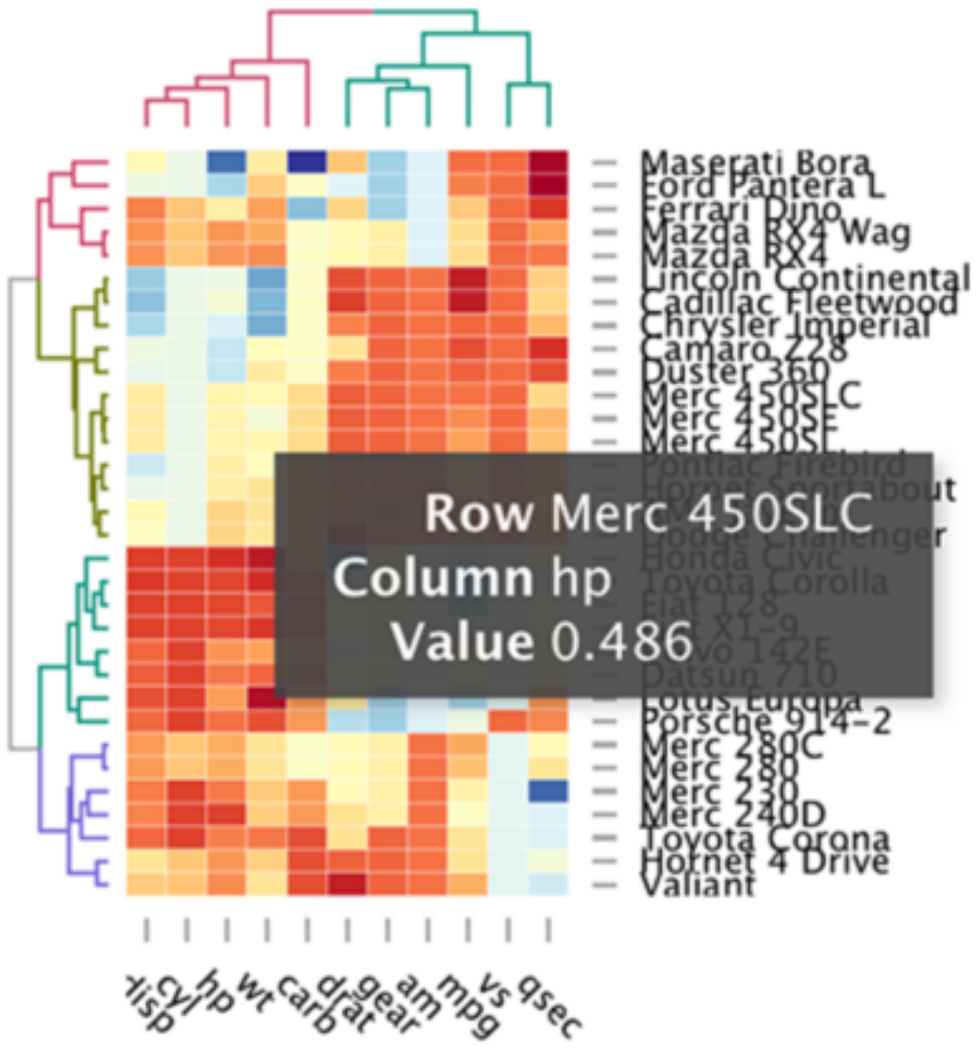
Make bublechart interactive

```r
# Libraries
library(ggplot2)
library(dplyr)
library(plotly)
library(viridis)
library(hrbrthemes)

# The dataset is provided in the gapminder library
library(gapminder)
data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

# Interactive version
p <- data %>%
  mutate(gdpPercap=round(gdpPercap,0)) %>%
  mutate(pop=round(pop/1000000,2)) %>%
  mutate(lifeExp=round(lifeExp,1)) %>%

  # Reorder countries to having big bubbles on top
  arrange(desc(pop)) %>%
  mutate(country = factor(country, country)) %>%

  # prepare text for tooltip
  mutate(text = paste("Country: ", country, "\nPopulation (M): ", pop, "\nLife Expectancy:
                      lifeExp, "\nGdp per capita: ", gdpPercap, sep="")) %>%

  # Classic ggplot
  ggplot( aes(x=gdpPercap, y=lifeExp, size = pop, color = continent, text=text)) +
  geom_point(alpha=0.7) +
  scale_size(range = c(1.4, 19), name="Population (M)") +
  scale_color_viridis(discrete=TRUE, guide=FALSE) +
  theme_ipsum() +
  theme(legend.position="none")

# turn ggplot interactive with plotly
pp <- ggplotly(p, tooltip="text")
pp
```



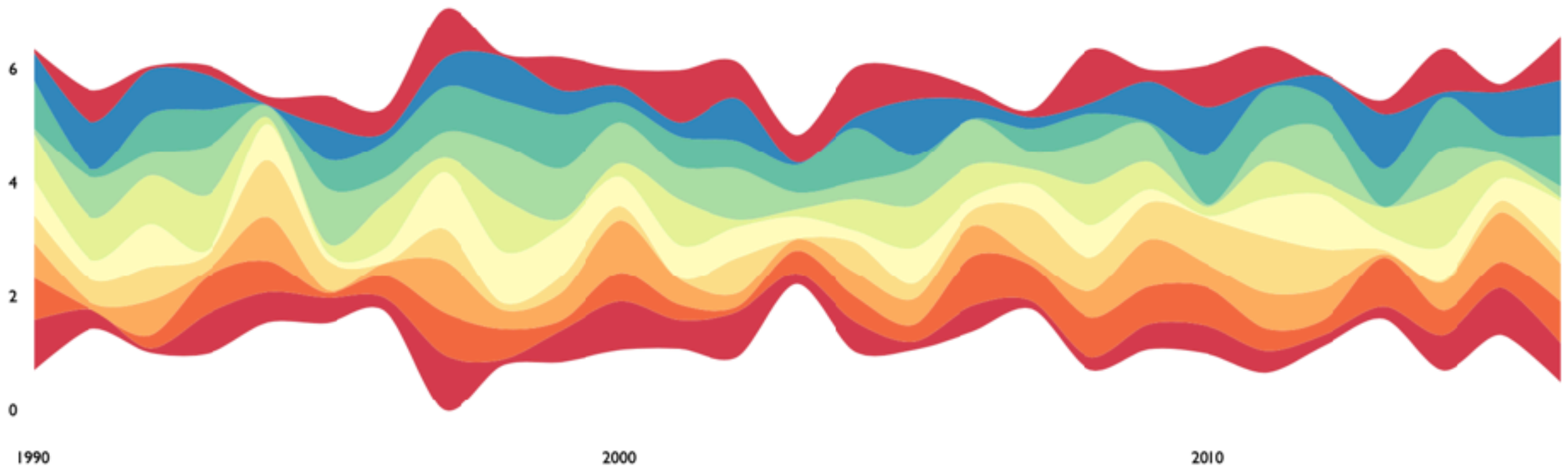
3D Heatmap

```
library("d3heatmap")
d3heatmap(scale(mtcars), colors = "RdYlBu",
          k_row = 4, # Number of groups in rows
          k_col = 2 # Number of groups in columns
          )
```



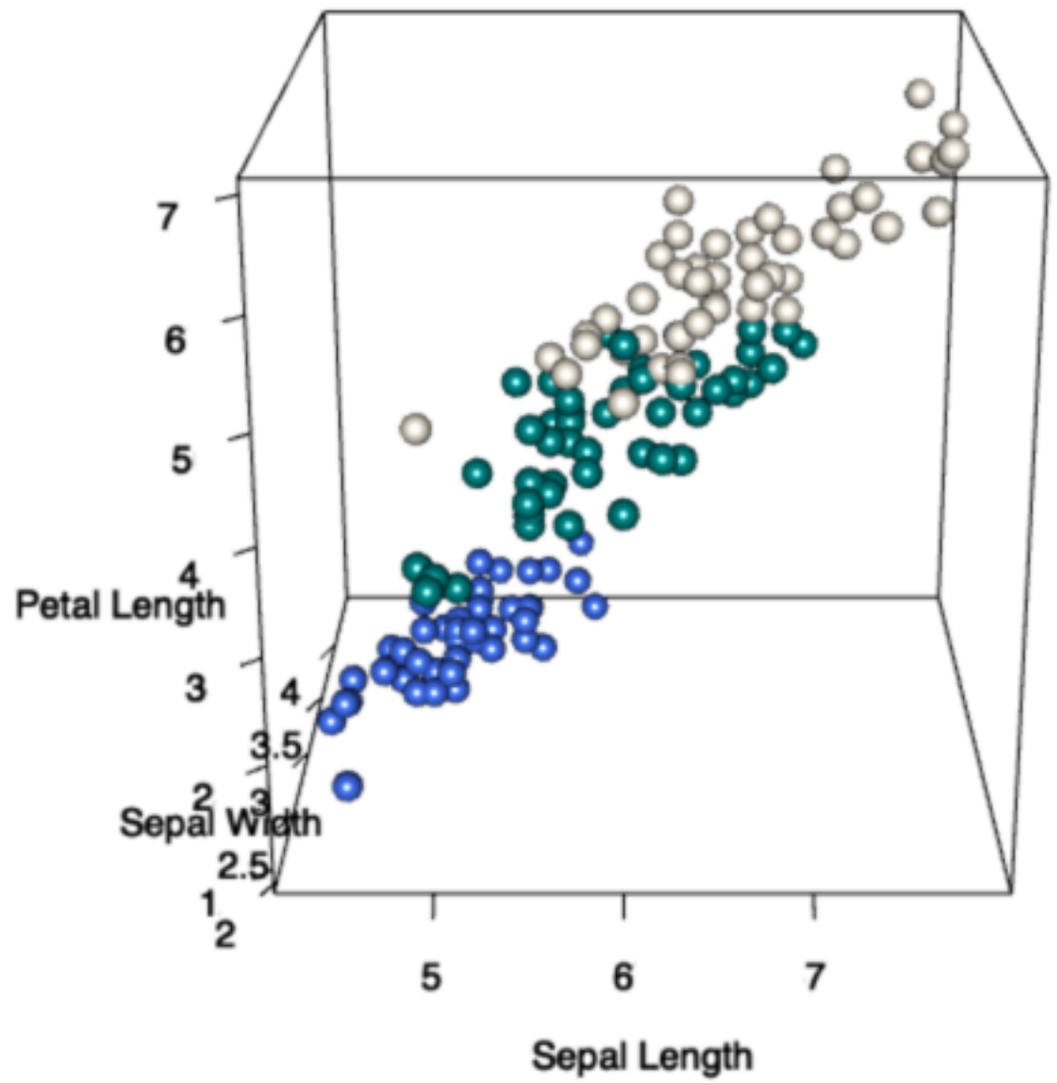
Stream graph

```r
# Library
library(streamgraph)

# Create data:
data <- data.frame(
  year=rep(seq(1990,2016) , each=10),
  name=rep(letters[1:10] , 27),
  value=sample( seq(0,1,0.0001) , 270)
)

# Stream graph with a legend
pp <- streamgraph(data, key="name", value="value", date="year", height="300px", width="1000px") %>%
  sg_legend(show=TRUE, label="names: ")
```



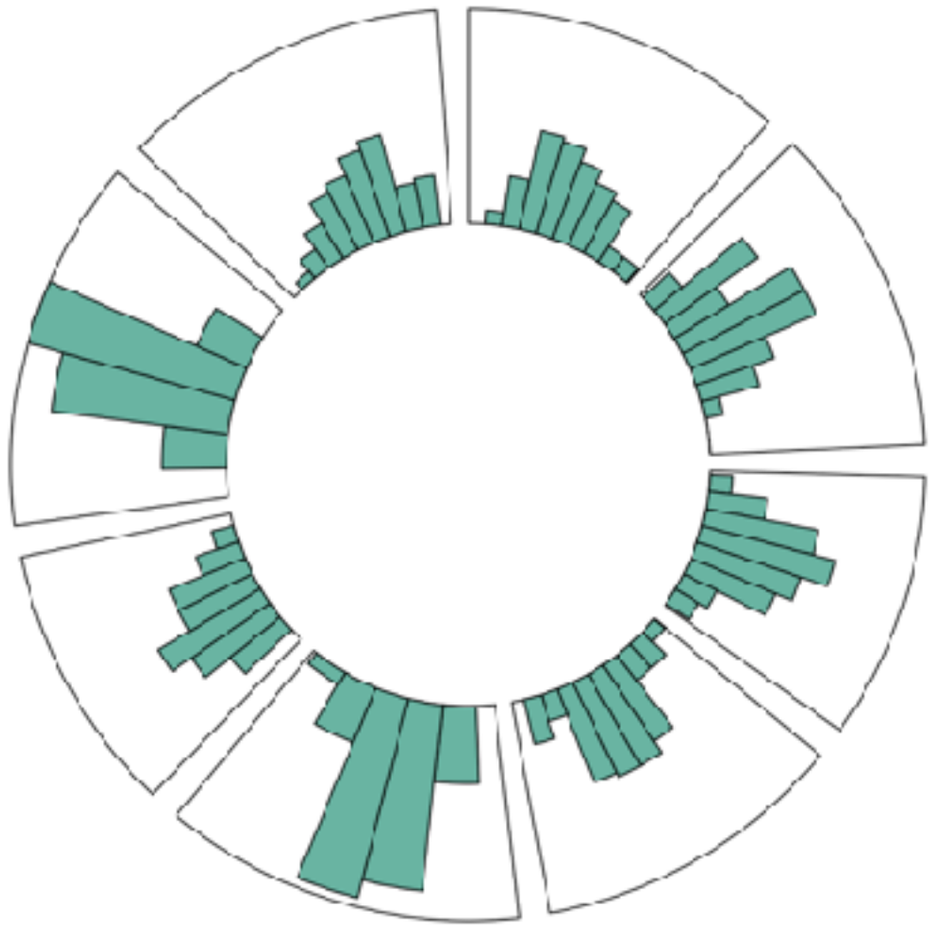
rgl

```r
 # library
library(rgl)

# This is to output a rgl plot in a rmarkdown document. Note that you must add webgl=TRU
#library(knitr)
#knit_hooks$set(webgl = hook_webgl)

# Data: the iris data is provided by R
data <- iris

# Add a new column with color
mycolors <- c('royalblue1', 'darkcyan', 'oldlace')
data$color <- mycolors[ as.numeric(data$Species) ]

# Plot
par(mar=c(0,0,0,0))
plot3d(
  x=data$`Sepal.Length`, y=data$`Sepal.Width`, z=data$`Petal.Length`,
  col = data$color,
  type = 's',
  radius = .1,
  xlab="Sepal Length", ylab="Sepal Width", zlab="Petal Length")

writeWebGL( filename="HtmlWidget/3dscatter.html" ,  width=600, height=600)
```


7
6
5
4
Petal Length
3
4
3.5
2
3
Sepal Width
2.5
1
2
5
6
7
Sepal Length

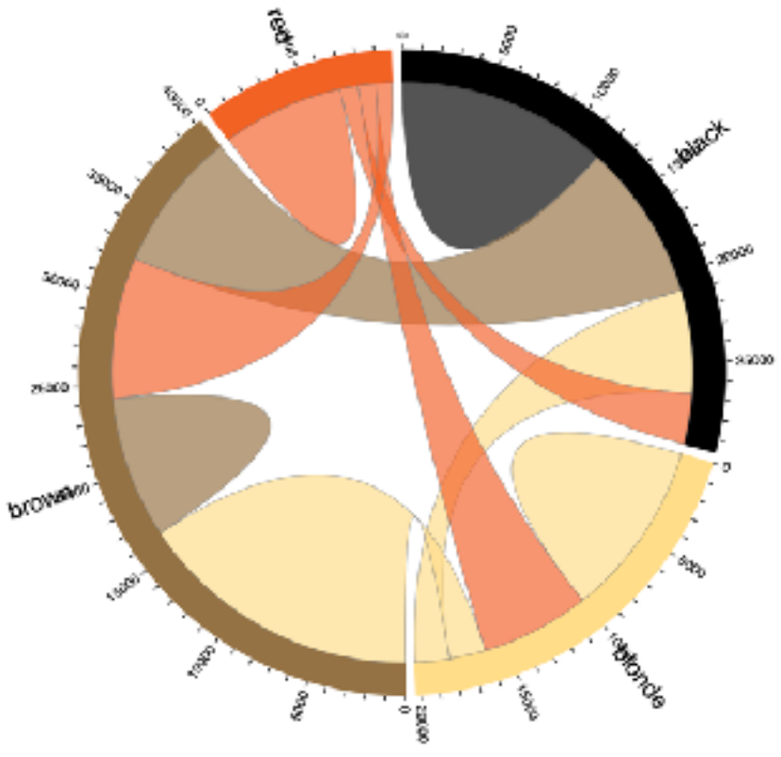
Cercos Chart

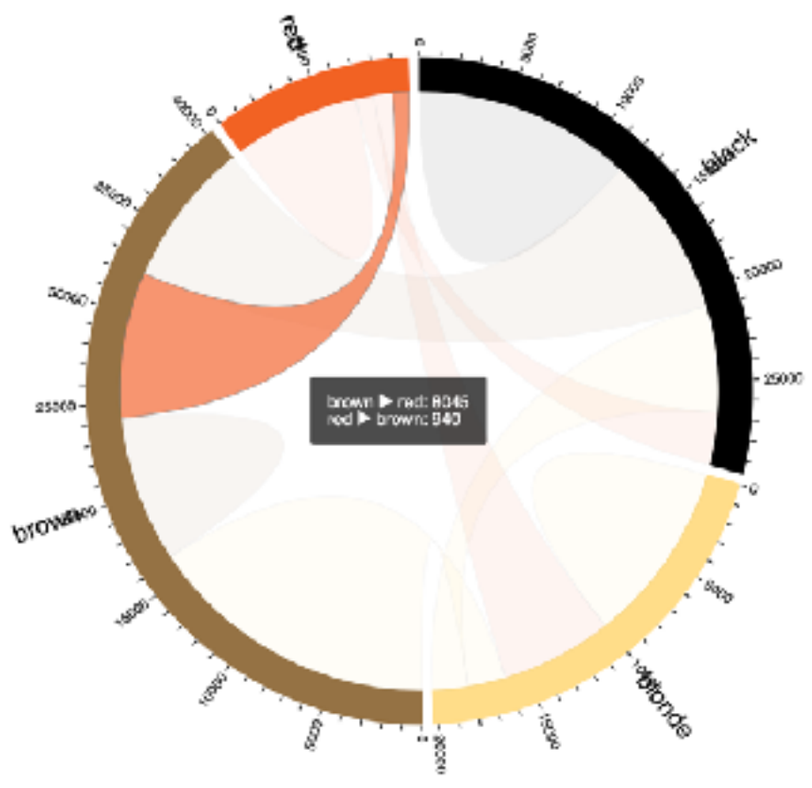
```r
# Upload library
library(circlize)
circos.par("track.height" = 0.4)

# Create data
data = data.frame(
    factor = sample(letters[1:8], 1000, replace = TRUE),
    x = rnorm(1000),
    y = runif(1000)
    )

# Step1: Initialise the chart giving factor and x-axis.
circos.initialize( factors=data$factor, x=data$x )

circos.trackHist(data$factor, data$x, bg.col = "white", col = "#69b3a2")
```



Chorddiag Chart

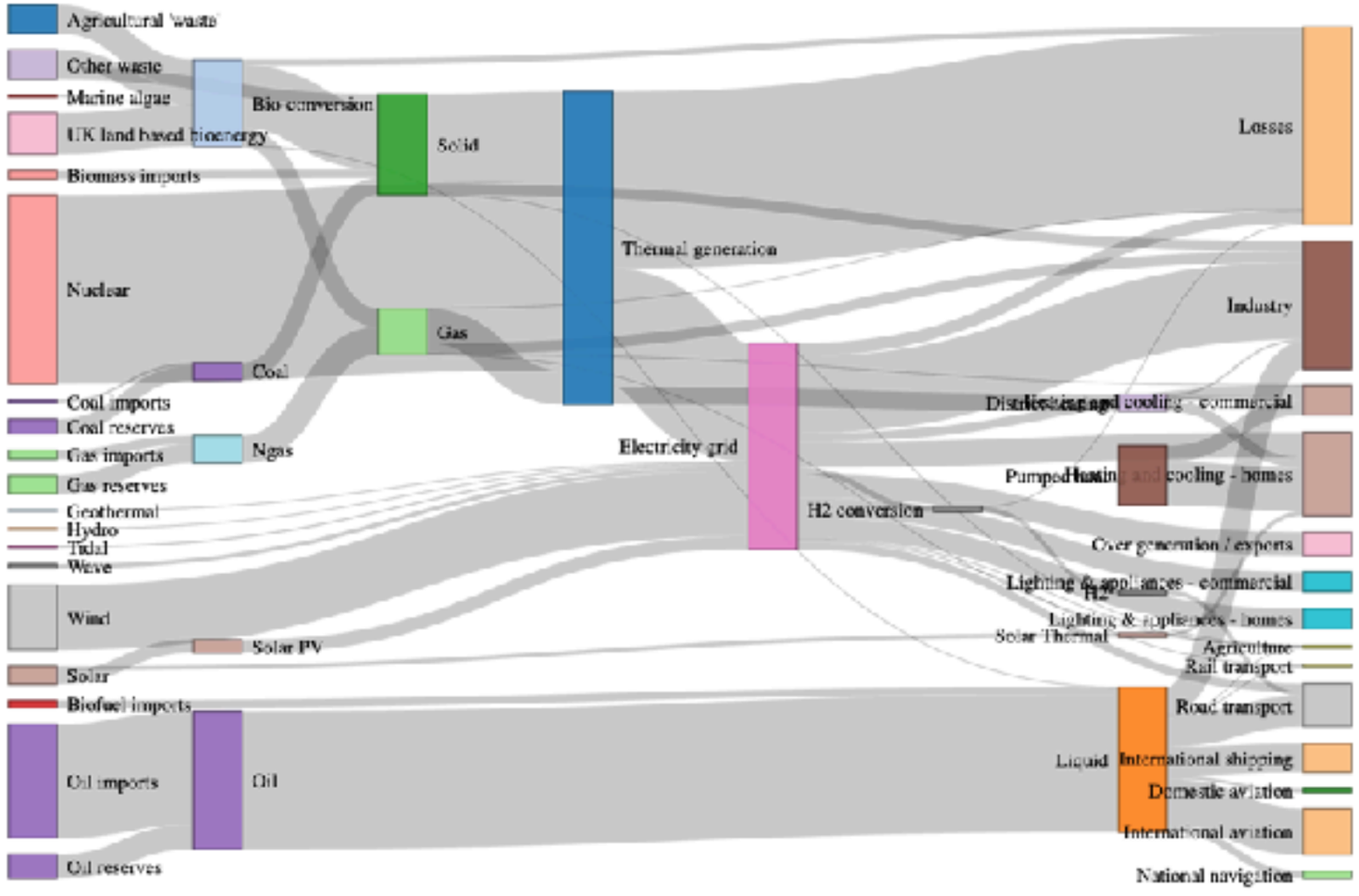
```
library(chorddiag)
m <- matrix(c(11975,  5871, 8916, 2868,
              1951, 10048, 2060, 6171,
              8010, 16145, 8090, 8045,
              1013,   990,  940, 6907),
            byrow = TRUE,
            nrow = 4, ncol = 4)
haircolors <- c("black", "blonde", "brown", "red")
dimnames(m) <- list(have = haircolors,
                    prefer = haircolors)

groupColors <- c("#000000", "#FFDD89", "#957244", "#F26223")
chorddiag(m, groupColors = groupColors, groupnamePadding = 20)
```





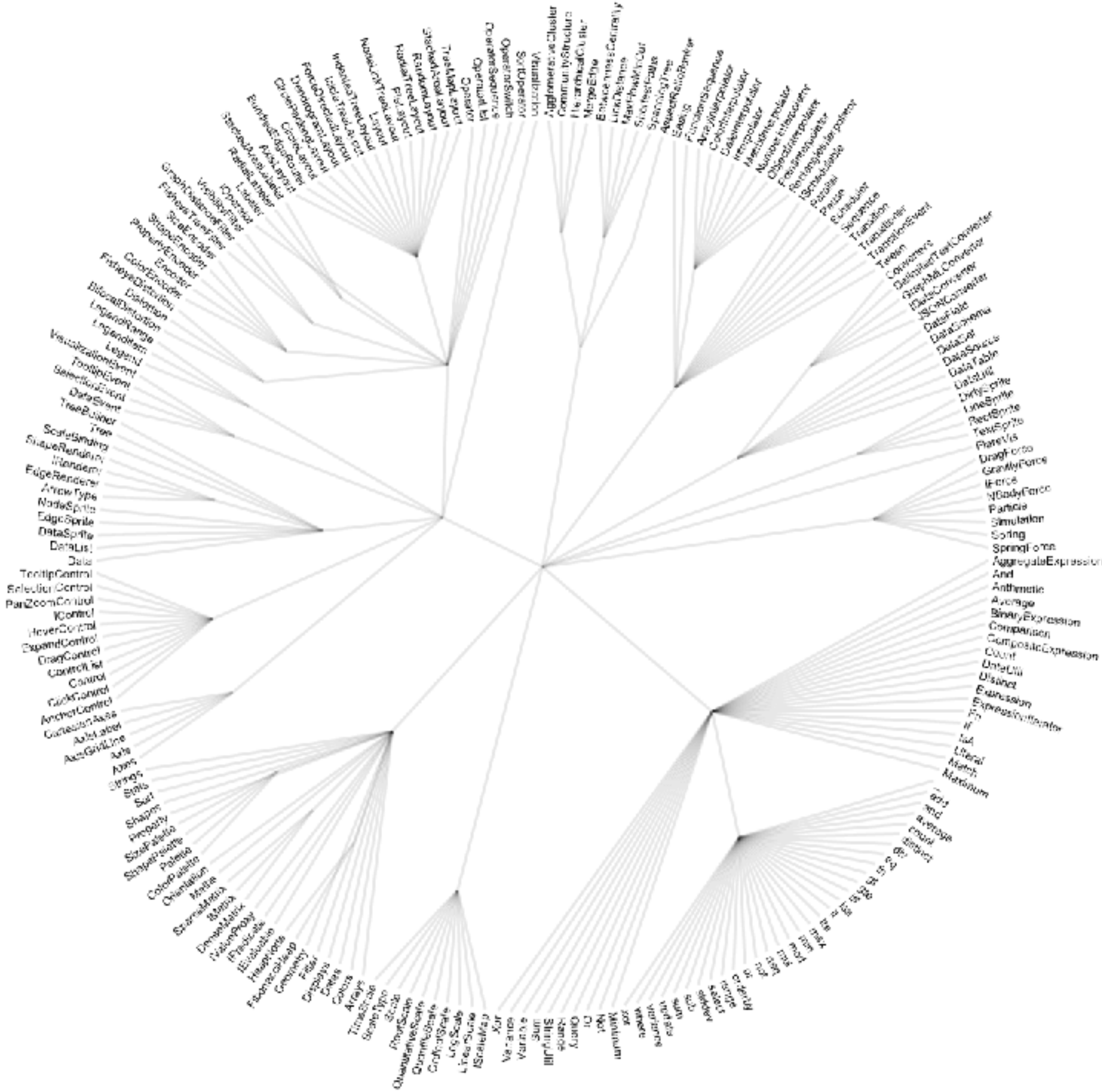
Sankey Chart

```r
# Load package
library(networkD3)

# Load energy projection data
URL <- "https://cdn.rawgit.com/christophergandrud/networkD3/master/JSONdata/energy.json"
Energy <- jsonlite::fromJSON(URL)

# Now we have 2 data frames: a 'links' data frame with 3 columns (from, to, value), and a
that gives the name of each node.

# Thus we can plot it
sankeyNetwork(Links = Energy$links, Nodes = Energy$nodes, Source = "source",
              Target = "target", Value = "value", NodeID = "name",
              units = "TWh", fontSize = 12, nodeWidth = 30)
```

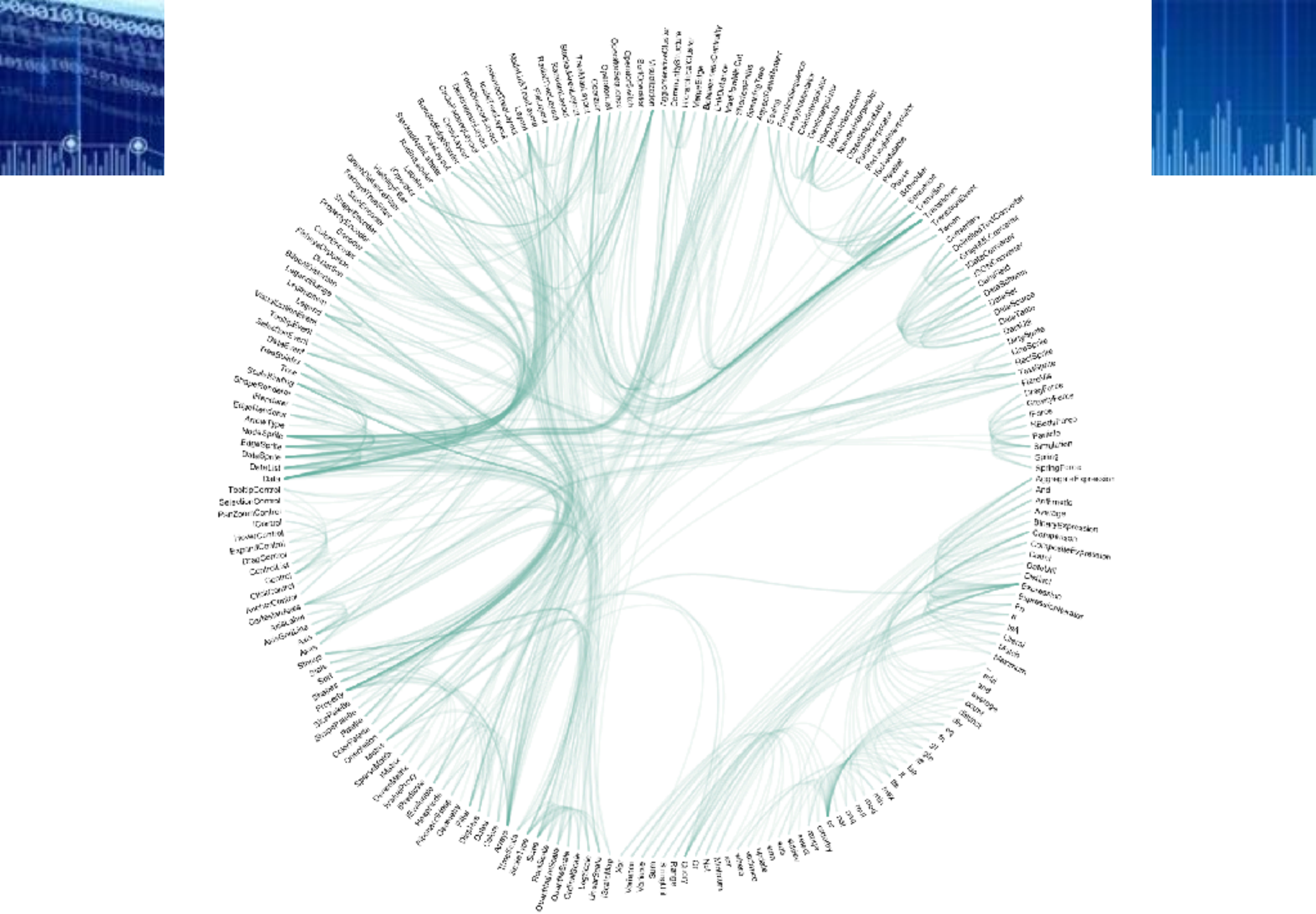


Hierarchical Edge Bundling

```r
# Libraries
library(tidyverse)
library(viridis)
library(patchwork)
library(hrbrthemes)
library(ggraph)
library(igraph)

# The flare dataset is provided in ggraph
edges <- flare$edges
vertices <- flare$vertices %>% arrange(name) %>% mutate(name=factor(name, name))
connections <- flare$imports

# Preparation to draw labels properly:
vertices$id=NA
myleaves=which(is.na( match(vertices$name, edges$from) ))
nleaves=length(myleaves)
vertices$id[ myleaves ] = seq(1:nleaves)
vertices$angle= 90 - 360 * vertices$id / nleaves
vertices$hjust<-ifelse( vertices$angle < -90, 1, 0)
vertices$angle<-ifelse(vertices$angle < -90, vertices$angle+180, vertices$angle)

# Build a network object from this dataset:
mygraph <- graph_from_data_frame(edges, vertices = vertices)

# The connection object must refer to the ids of the leaves:
from = match( connections$from, vertices$name)
to = match( connections$to, vertices$name)

# Basic dendrogram
ggraph(mygraph, layout = 'dendrogram', circular = TRUE) +
    geom_edge_link(size=0.4, alpha=0.1) +
    geom_node_text(aes(x = x*1.01, y=y*1.01, filter = leaf, label=shortName, angle = angle, hjust=hjust), siz
e=1.5, alpha=1) +
    coord_fixed() +
    theme_void() +
    theme(
      legend.position="none",
      plot.margin=unit(c(0,0,0,0),"cm"),
    ) +
    expand_limits(x = c(-1.2, 1.2), y = c(-1.2, 1.2))
```


AggregateExpression
And
Arithmetic
Average
BinaryExpression
Comparison
CompositeExpression
Count
DateUtil
Distinct
Expression
Literal
Match
Maximum
SpringForce
Spring
Simulation
Particle
NBodyForce
IForce
GravityForce
DragForce
Data
DataList
DataSprite
EdgeSprite
NodeSprite
ArrowType
EdgeRenderer
TooltipControl
SelectionControl
PanZoomControl
IControl
HoverControl
ExpandControl
DragControl
ControlList
Control
ClickControl
AnchorControl

```r
# Make the plot
ggraph(mygraph, layout = 'dendrogram', circular = TRUE) +
    geom_conn_bundle(data = get_con(from = from, to = to), alpha = 0.1, colour="#69b3a2") +
    geom_node_text(aes(x = x*1.01, y=y*1.01, filter = leaf, label=shortName, angle = angle, hjust=hjust), siz
e=1.5, alpha=1) +
    coord_fixed() +
    theme_void() +
    theme(
      legend.position="none",
      plot.margin=unit(c(0,0,0,0),"cm"),
    ) +
    expand_limits(x = c(-1.2, 1.2), y = c(-1.2, 1.2))
```



Workshop 2.3 Explore your data

From the data you selected, using interactive chart to render your variables

Infographic

Install the packages used for Infographic

1.waffle

2.extrafont

3.tidyverse

4.echarts4r

5.echarts4r.assets

devtools::install_github(“JohnCoene/echarts4r.assets"

waffle square pie chart

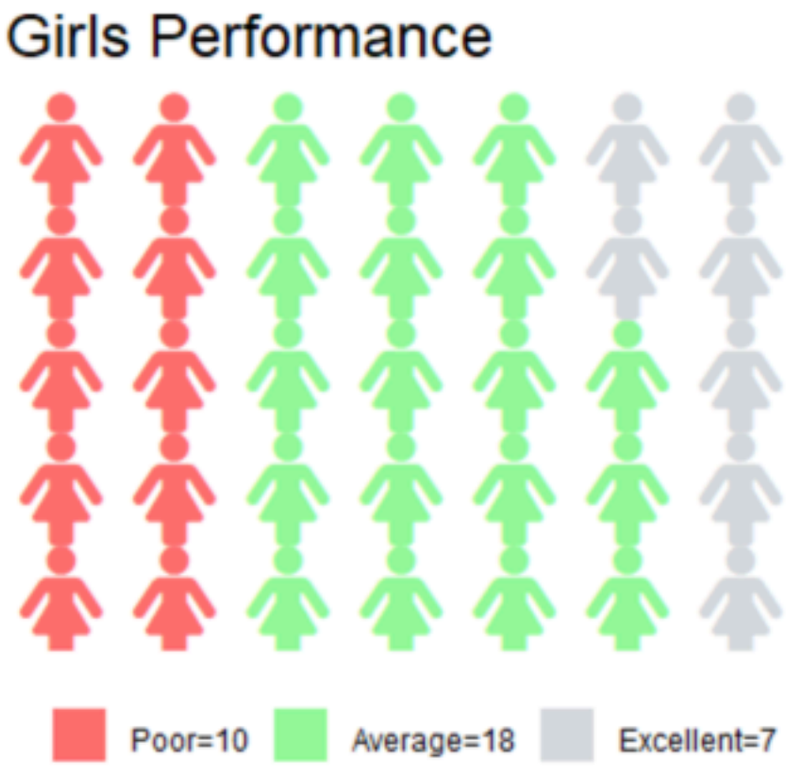
```
library(waffle)
waffle(
  c('Yes=70%' = 70, 'No=30%' = 30), rows = 10, colors = c("#FD6F6F", "#93FB98"),
  title = 'Responses', legend_pos="bottom"
)
```

Use icon for waffle chart

Steps to download and install fontawesome fonts

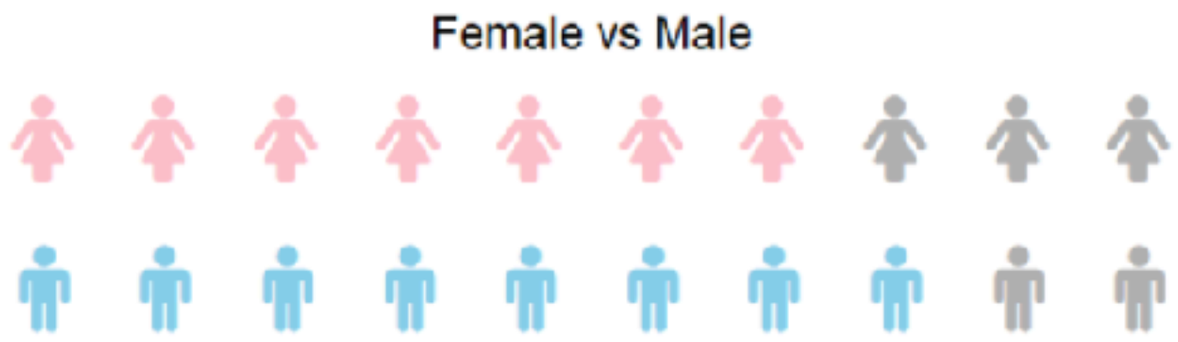
1. First step is to load extrafont library by running this command `library(extrafont)`
2. Download and install `fontawesome` fonts from this URL `https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/fonts/fontawesome-webfont.ttf`
3. Import downloaded fontawesome font by using this command. Make sure to specify your folder location containing fontawesome. `extrafont::font_import (path="C:\\Users\\DELL\\Downloads", pattern = "awesome", prompt = FALSE)`
4. Load fonts by using the command `loadfonts(device = "win")`
5. Check whether font awesome is installed successfully by running this command `fonts()[grep("Awesome", fonts())]` . It should return `FontAwesome`

```
waffle(
  c(`Poor=10` =10, `Average=18` = 18, `Excellent=7` =7), rows = 5, colors = c("#FD6F6F", "#93FB98", "#D5D9DD"),
  use_glyph = "female", glyph_size = 12 ,title = 'Girls Performance', legend_pos="bottom"
)
```



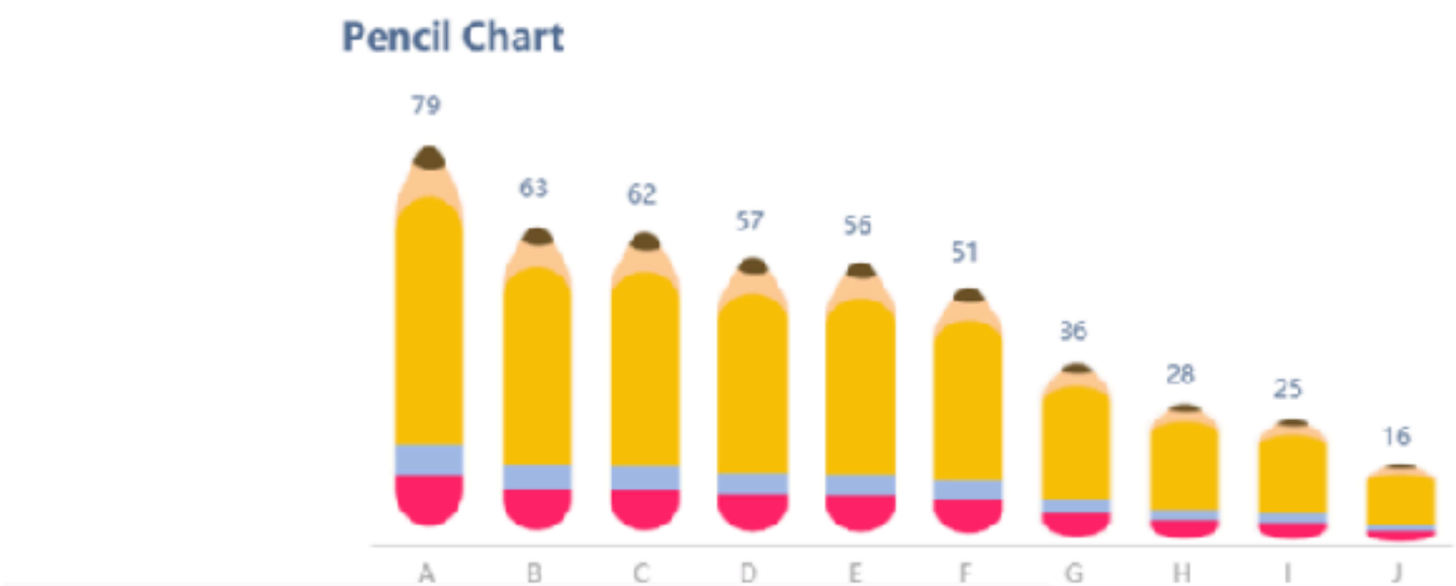
iron()

```
iron(
  waffle(
    c('TRUE' = 7, 'FALSE' = 3),
    colors = c("pink", "grey70"),
    use_glyph = "female",
    glyph_size = 12,
    title = "Female vs Male",
    rows = 1,
    legend_pos = "none"
  ) + theme(plot.title = element_text(hjust = 0.5))
  ,
  waffle(
    c('TRUE' = 8, 'FALSE' = 2),
    colors = c("skyblue", "grey70"),
    use_glyph = "male",
    glyph_size = 12,
    rows = 1,
    legend_pos = "none"
  )
)
```



Add image to chart

```
df02 <- data.frame(
  x = LETTERS[1:10],
  y = sort(sample(10:80,10), decreasing = TRUE)
)

df02 %>%
  e_charts(x) %>%
  e_pictorial(y, symbol = paste0("image://","https://1.bp.blogspot.com/-
klwxpFekdEQ/XOublhkalyI/AAAAAAAAHlE/25psl9x4oNkbJoLc2CKTXgV2pEj6tAvigCLcBGAs/s1600/pencil.png"))
%>%
e_theme("westeros") %>%
e_title("Pencil Chart",  padding=c(10,0,0,50))%>%
e_labels(show = TRUE)%>%
e_legend(show = FALSE) %>%
e_x_axis(splitLine=list(show = FALSE)) %>%
e_y_axis(show=FALSE, splitLine=list(show = FALSE))
```



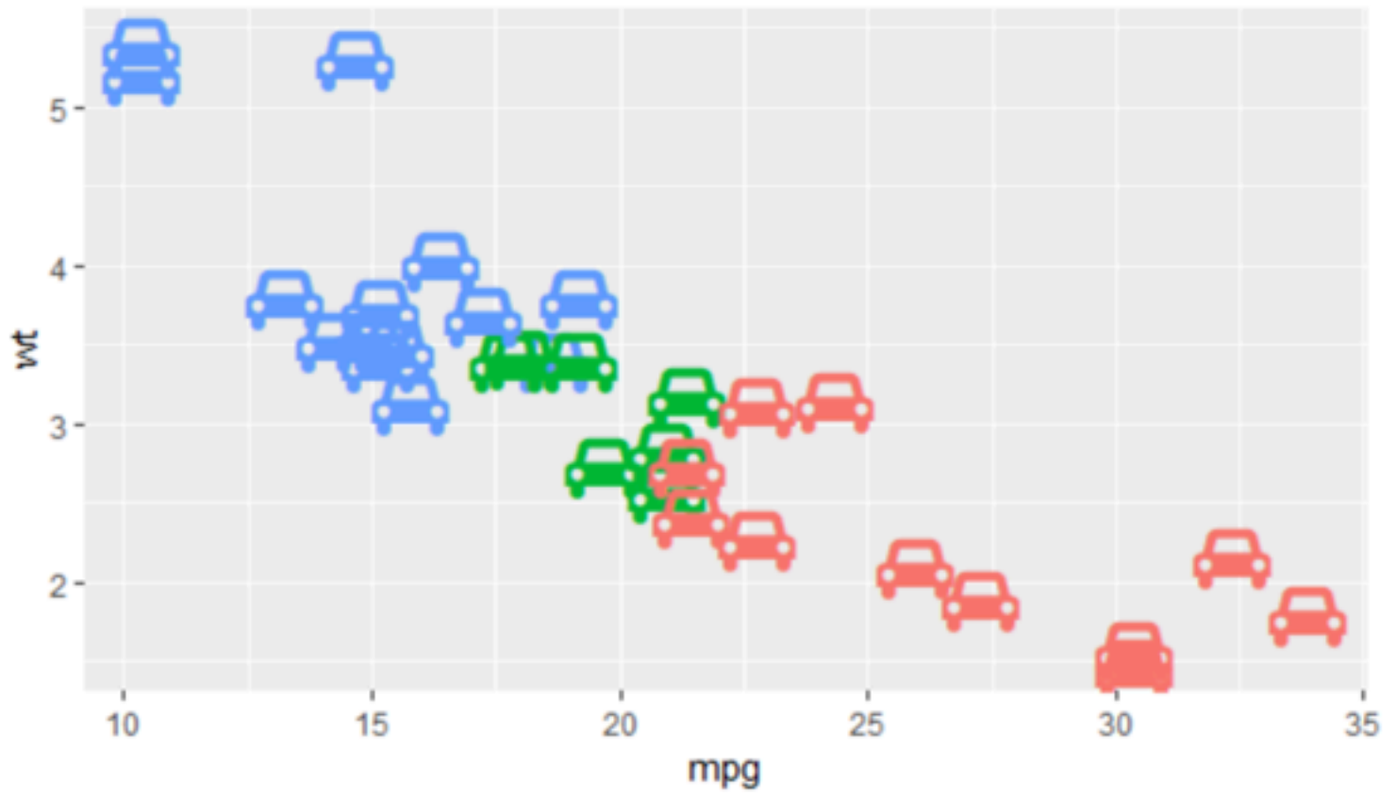
```
gender = data.frame(gender=c("Male", "Female"), value=c(65, 35),
                    path = c('path://M18.2629891,11.7131596 L6.8091608,11.7131596 C1.6685112,11.7131596 0,13.032145
0,18.6237673 L0,34.9928467 C0,38.1719847 4.28388932,38.1719847 4.28388932,34.9928467
L4.65591984,20.0216948 L5.74941883,20.0216948 L5.74941883,61.000787 C5.74941883,65.2508314
11.5891201,65.1268798 11.5891201,61.000787 L11.9611506,37.2137775 L13.1110872,37.2137775
L13.4831177,61.000787 C13.4831177,65.1268798 19.3114787,65.2508314 19.3114787,61.000787
L19.3114787,20.0216948 L20.4162301,20.0216948 L20.7882606,34.9928467 C20.7882606,38.1719847
25.0721499,38.1719847 25.0721499,34.9928467 L25.0721499,18.6237673 C25.0721499,13.032145
23.4038145,11.7131596 18.2629891,11.7131596 M12.5361629,1.11022302e-13 C15.4784742,1.11022302e-13
17.8684539,2.38997966 17.8684539,5.33237894 C17.8684539,8.27469031 15.4784742,10.66467
12.5361629,10.66467 C9.59376358,10.66467 7.20378392,8.27469031 7.20378392,5.33237894
C7.20378392,2.38997966 9.59376358,1.11022302e-13 12.5361629,1.11022302e-13',
'path://M28.9624207,31.5315864 L24.4142575,16.4793596 C23.5227152,13.8063773 20.8817445,11.7111088
17.0107398,11.7111088 L12.112691,11.7111088 C8.24168636,11.7111088 5.60080331,13.8064652
4.70917331,16.4793596 L0.149791395,31.5315864 C-0.786976655,34.7595013 2.9373074,35.9147532
3.9192135,32.890727 L8.72689855,19.1296485 L9.2799493,19.1296485 C9.2799493,19.1296485
2.95992025,43.7750224 2.70031069,44.6924335 C2.56498417,45.1567684 2.74553639,45.4852068
3.24205501,45.4852068 L8.704461,45.4852068 L8.704461,61.6700801 C8.704461,64.9659872
13.625035,64.9659872 13.625035,61.6700801 L13.625035,45.360657 L15.5097899,45.360657
L15.4984835,61.6700801 C15.4984835,64.9659872 20.4191451,64.9659872 20.4191451,61.6700801
L20.4191451,45.4852068 L25.8814635,45.4852068 C26.3667633,45.4852068 26.5586219,45.1567684
26.4345142,44.6924335 C26.1636859,43.7750224 19.8436568,19.1296485 19.8436568,19.1296485
L20.3966199,19.1296485 L25.2043926,32.890727 C26.1862111,35.9147532 29.9105828,34.7595013
28.9625083,31.5315864 L28.9624207,31.5315864 Z M14.5617154,0 C17.4960397,0 19.8773132,2.3898427
19.8773132,5.33453001 C19.8773132,8.27930527 17.4960397,10.66906 14.5617154,10.66906
C11.6274788,10.66906 9.24611767,8.27930527 9.24611767,5.33453001 C9.24611767,2.3898427 11.6274788,0
14.5617154,0 L14.5617154,0 Z'))
```

```
gender %>%
e_charts(gender) %>%
e_x_axis(splitLine=list(show = FALSE),
        axisTick=list(show=FALSE),
        axisLine=list(show=FALSE),
        axisLabel= list(show=FALSE)) %>%
e_y_axis(max=100,
        splitLine=list(show = FALSE),
        axisTick=list(show=FALSE),
        axisLine=list(show=FALSE),
        axisLabel=list(show=FALSE)) %>%
e_color(color = c('#69cce6','#eee')) %>%
e_pictorial(value, symbol = path, z=10, name= 'realValue',
          symbolBoundingData= 100, symbolClip= TRUE) %>%
e_pictorial(value, symbol = path, name= 'background',
          symbolBoundingData= 100) %>%
e_labels(position = "bottom", offset= c(0, 10),
        textStyle =list(fontSize= 20, fontFamily= 'Arial',
                    fontWeight ='bold',
                    color= '#69cce6'),
        formatter="{@[1]}%{@[0]}") %>%
e_legend(show = FALSE) %>%
e_theme("westeros")
```


65% Male
35% Female

icon in ggplot2

```
library(ggplot2)
ggplot (mtcars)  +
  geom_text( aes ( mpg , wt , colour = factor ( cyl )),
             label =  "\uf1b9" ,
             family =  "FontAwesome" ,
             size =  7)
```



Workshop 2.4 Create Info Graphic

From your imported data, create infographic chart using waffle and chart