Lecture 3: Elegant Graphics with ggplot2

Xiao Guo

2023/3/4

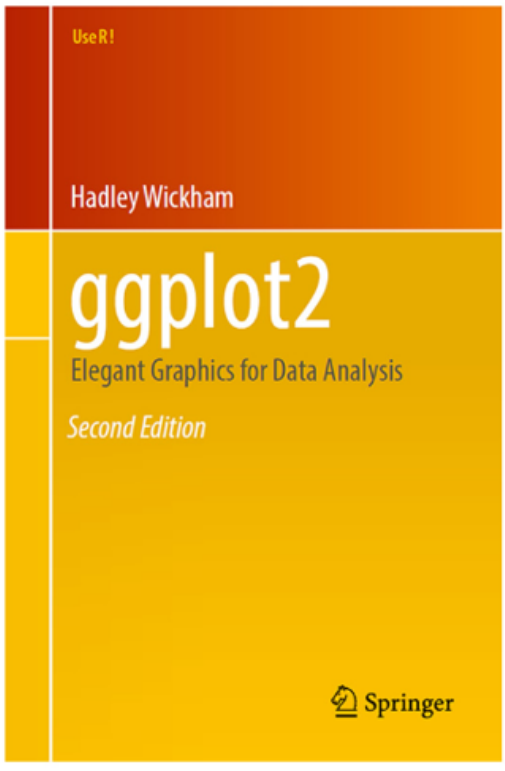
3.1. Grammar of Graphics

Hadley Wickham and R:ggplot2

- Chief scientist at RStudio, Creator of popular R packages: ggplot2, dplyr, tidyr, devtools, etc; “The man who r evolutionized R”.
- R graphics: base -> lattice -> ggplot2

  “ggplot2, started in 2005, is an attempt to take the good things about base and lattice graphics and improve on them with a strong underlying model” (Hadley Wickham).
- R:ggplot2 is one of most commonly downloaded R packages.
- Based on Grammar of Graphics by Wilkinson (2005; Springer 2ed).

http://hadley.nz/



Springer (2016; 2ed)

Grammar of Graphics (GG) (图形语法)

Wilkinson(2005)创建了一套用来描述所有统计图形深层特性的语法规则，该语法回答了“什么是统计图形”这一问题。

一张统计图形就是从数据到**几何对象**(geometric object，缩写为geom，包括点、线、条形等)的**图形属性**(aesthetic object，缩写为aes，包括颜色、形状、大小等)的一个映射。此外，图形中还可能包含数据的**统计变换**(statistical transformation，缩写为stats)，最后绘制在某个特定的**坐标系**(coordinate system，缩写为coord)中，而**分面**(facet，指将数据绘图窗口划分为若干个子窗口)则可用来生成数据的不同子集的图形。

- 最基础的部分是你想要可视化的**数据**以及一系列将数据中的变量对应到图形属性的**映射**;
- **几何对象**代表在图中实际看到的图形元素，如点、线、多边形等;
- **统计变换**是对数据进行的某种汇总。例如，将数据分组计数以创建直方图;
- **标度**的作用是将数据的取值映射到图形空间。占线标度的常见做法是绘制图例和坐标轴;
- **坐标系**描述了数据如何映射到图形所在平面，它同时提供了看图所需的坐标轴和网格线;
- **分面**描述了如何将数据分解成各个子集，以及如何对子集作图并联合进行展示。分面也叫条件作图或网格作图。

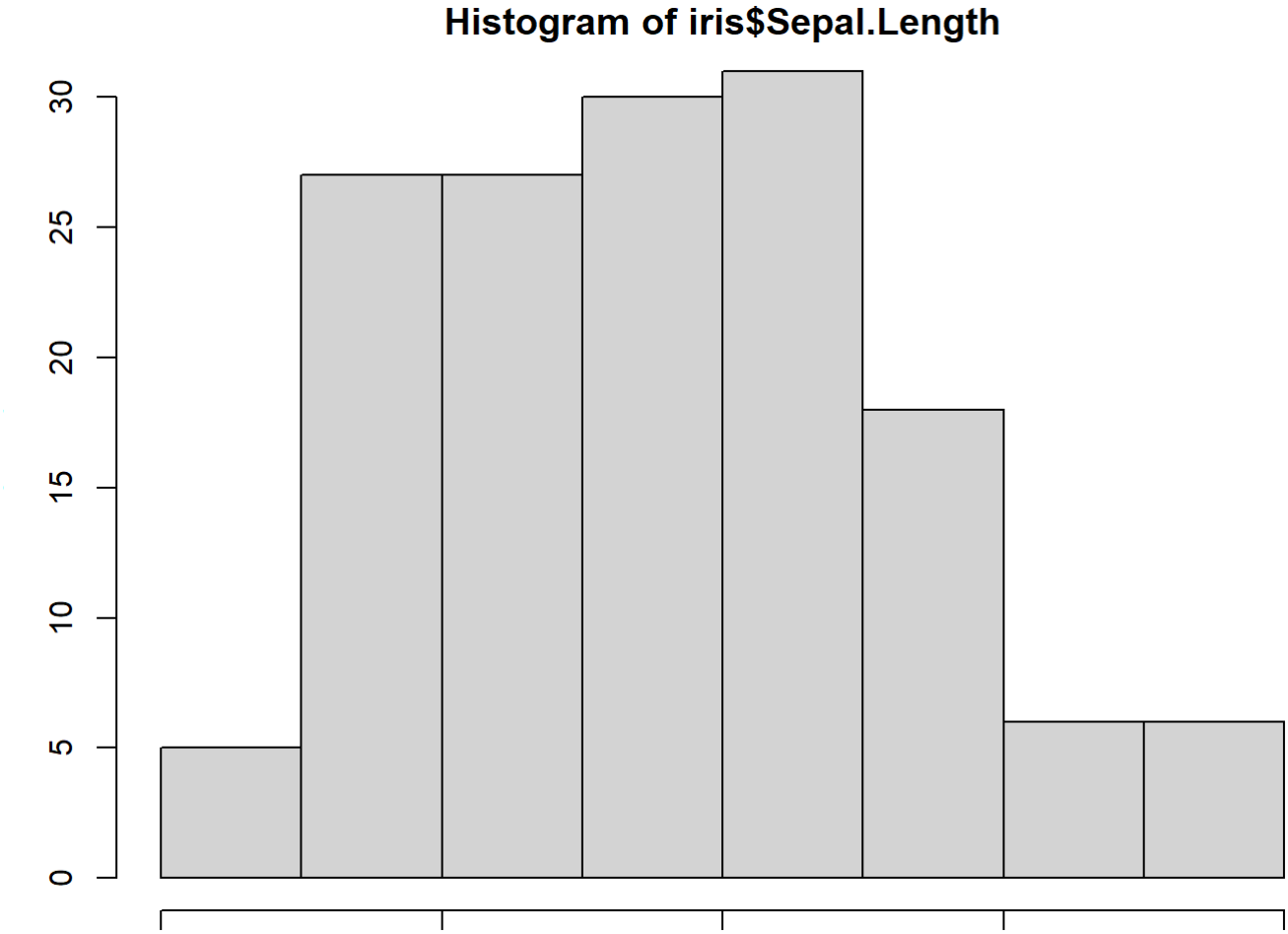
R:ggplot2 package

- The most popular package for producing static visualizations in R; New upgrade to Version 3.2.1; See CRAN for updated information.
- Online documentation at https://ggplot2.tidyverse.org/ (https://ggplot2.tidyverse.org/)

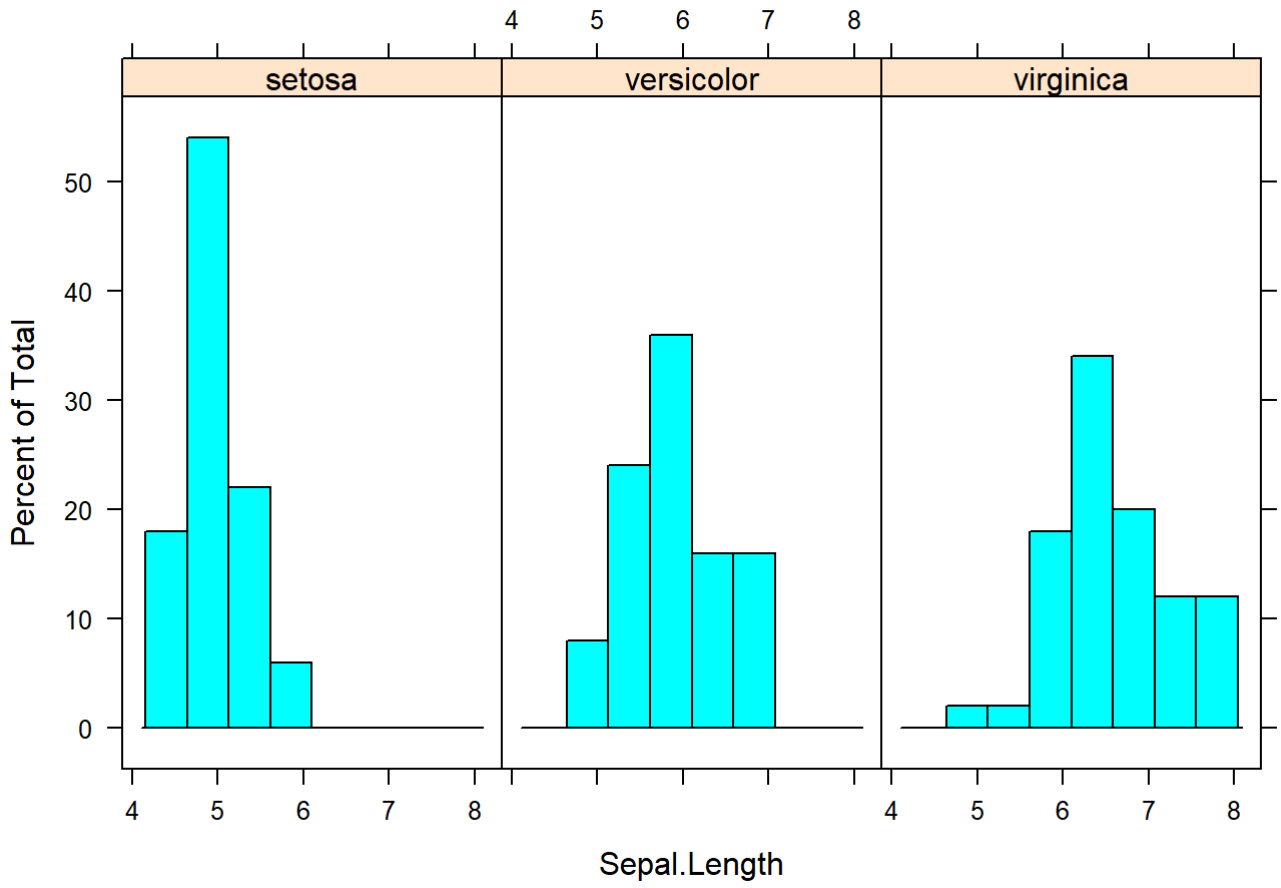
- Download the useful cheatsheet created by Rstudio at https://github.com/rstudio/cheatsheets/blob/main/data-visualization.pdf (https://github.com/rstudio/cheatsheets/blob/main/data-visualization.pdf)
- Also available in Python.

Base, Lattice and ggplot2 styles (first impression)

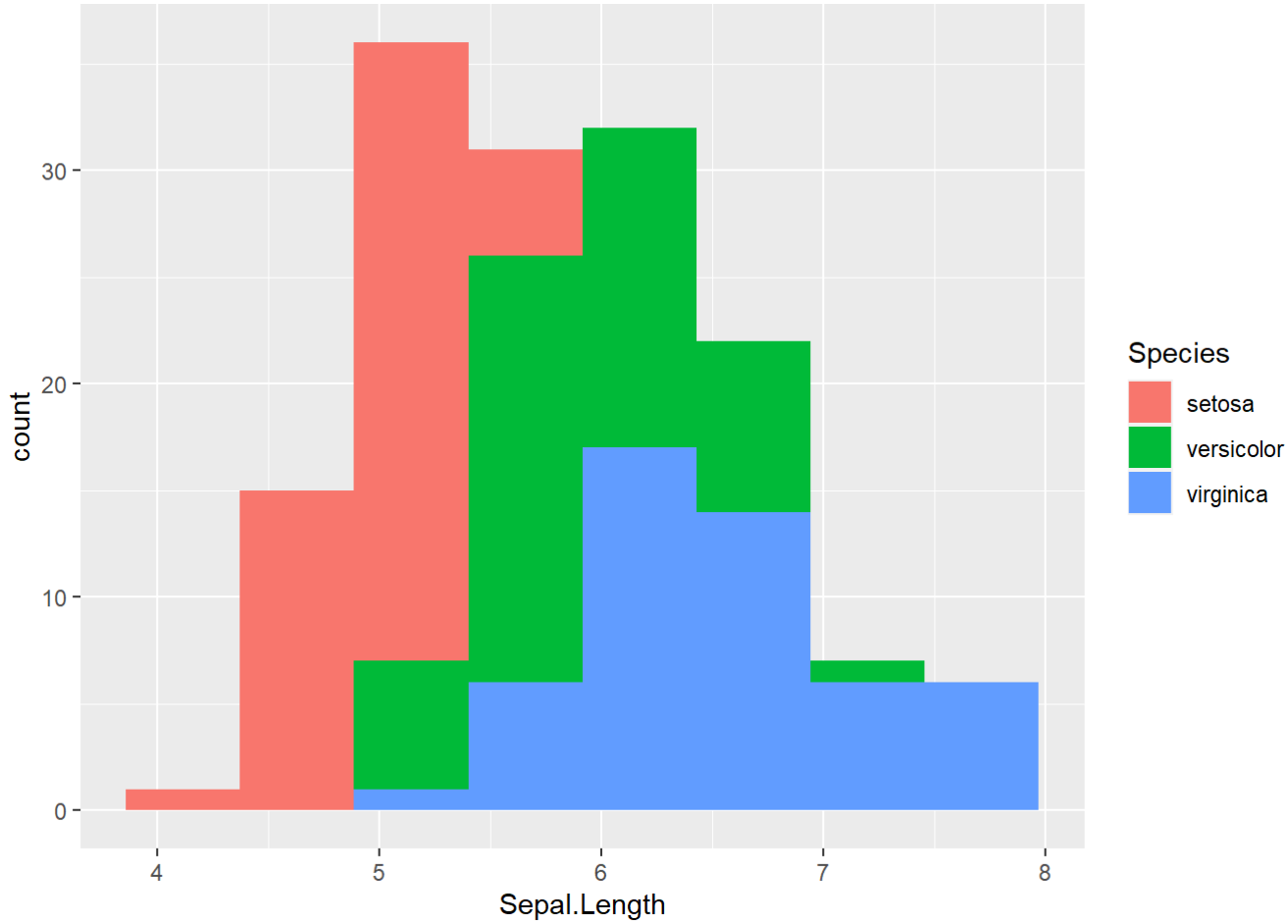
```
par(mar=c(1,3,1,0))
hist(iris$Sepal.Length) # Base graphics
```



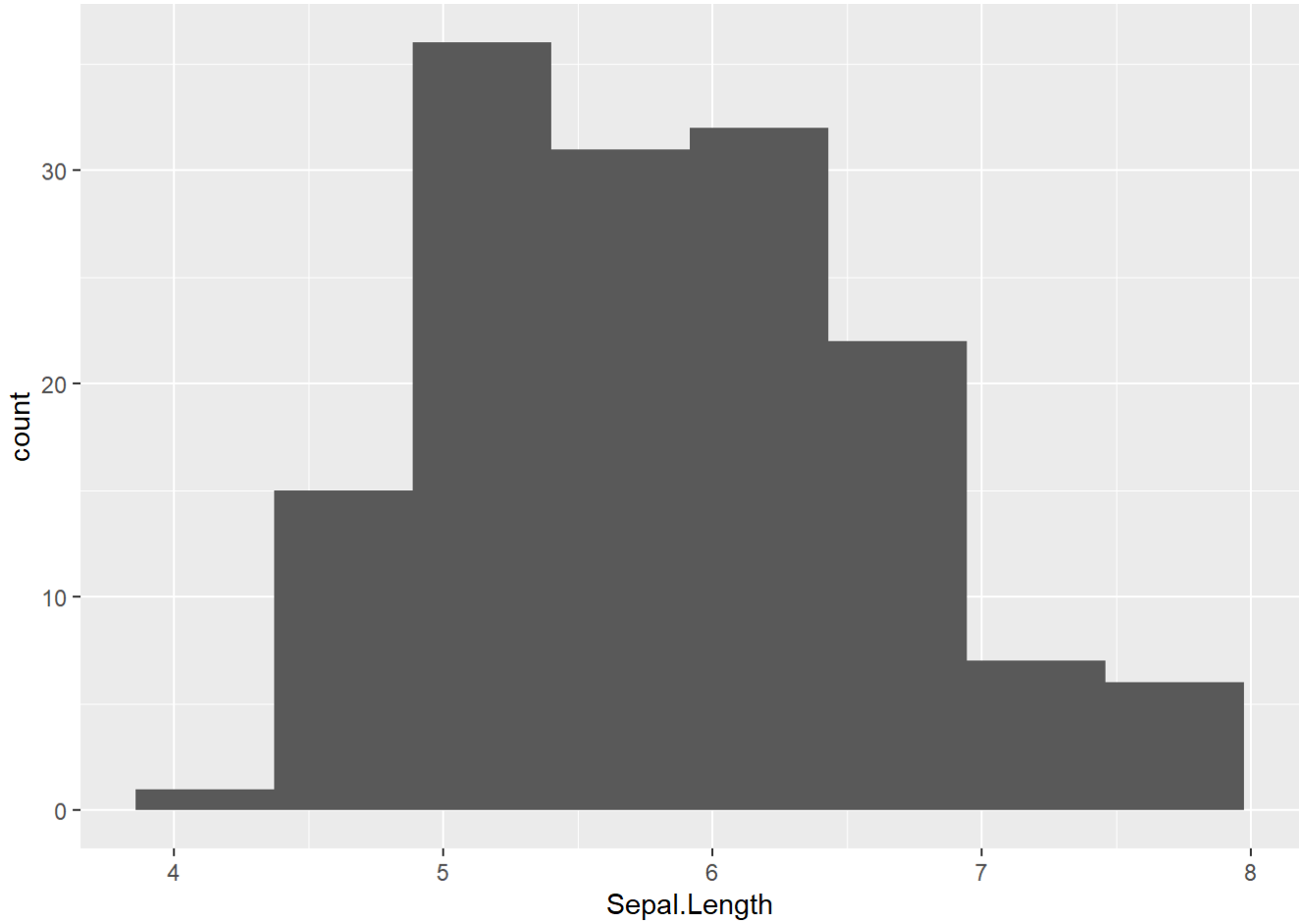

```
library(lattice)
histogram(data=iris, ~Sepal.Length|Species)
```




```
library(ggplot2)
ggplot(data=iris,
       aes(x=Sepal.Length, fill=Species)) +
  geom_histogram(bins=8)
```




```
library(ggplot2)
ggplot(data=iris,
       aes(x=Sepal.Length)) +
  geom_histogram(bins=8)
```



You Will Learn …

- R:ggplot2 provides two ways/levels to build graphs:
  - qplot() - quick plot, supplies many defaults
  - ggplot() - grammar of graphics plot, with more controls
- Options and themes for making sophisticated ggplot2 graphs
- Later in this course, ggplot2 will also be used for animated/interactive plots

3.2 Quick plots with qplot()

- qplot() is analog to base plot(), where “q” means quick
- qplot() may create a quick plot with minimum typing
- It defines a plot in a single call with the basic syntax:
  `qplot(dataframe, variables, [geom], options)`
- Automatic use of default settings to make life easier

- A sensible geom will be picked by default if it is not supplied.

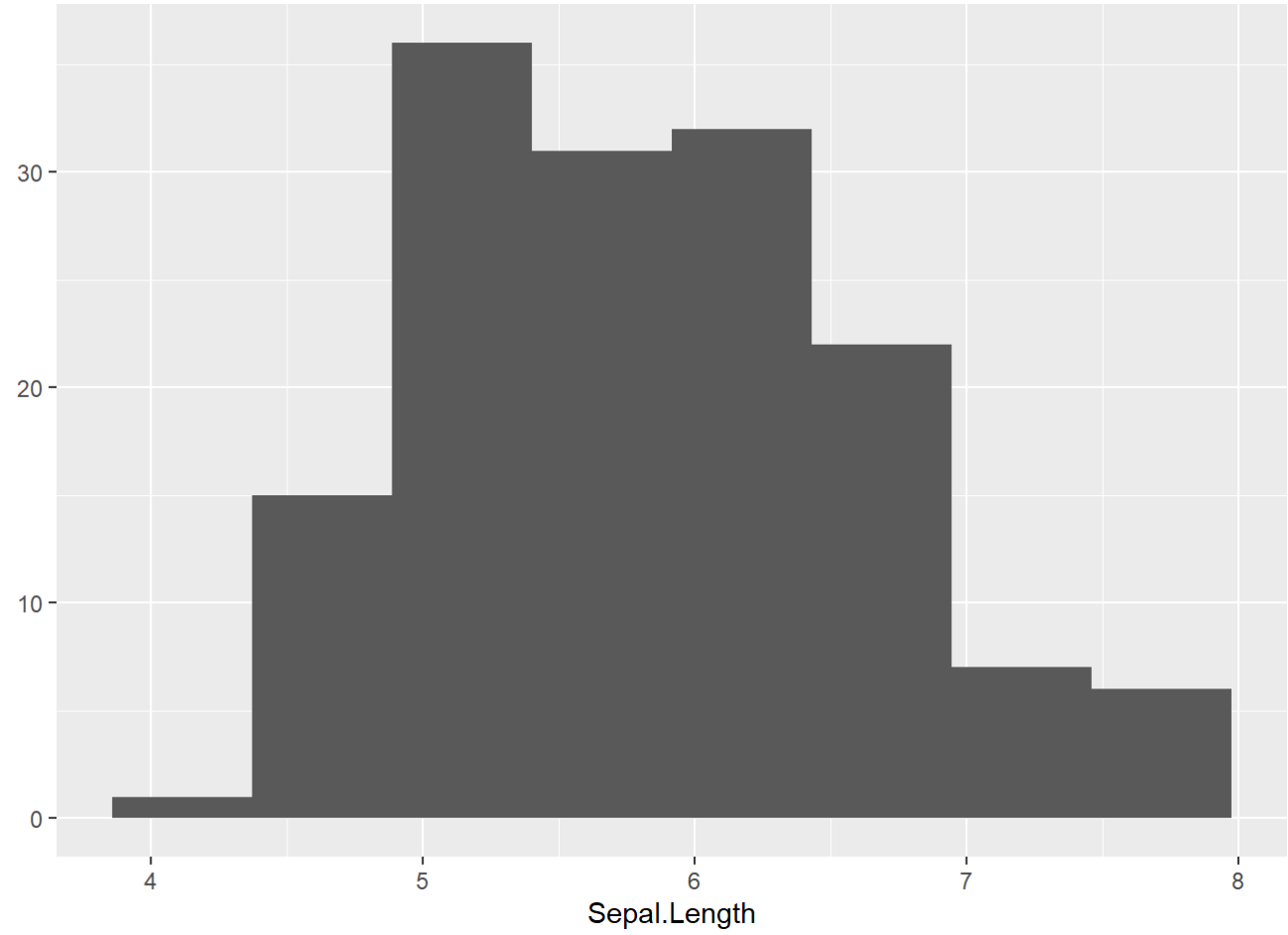
Histogram

```
library(gridExtra)
```

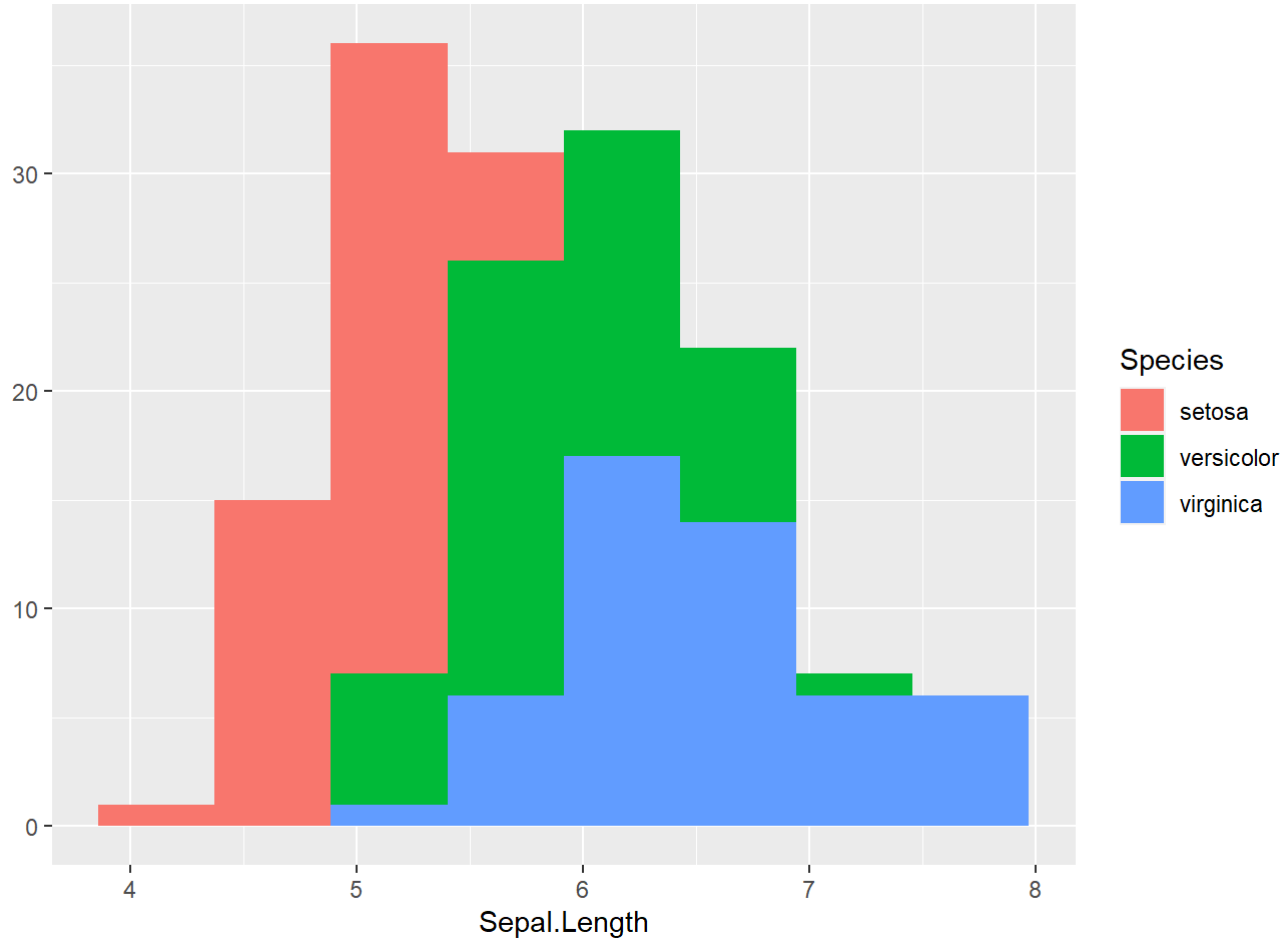
```
## Warning: package 'gridExtra' was built under R version 4.0.5
```

```
qplot(data = iris, Sepal.Length, geom="histogram", bins = 8)
```

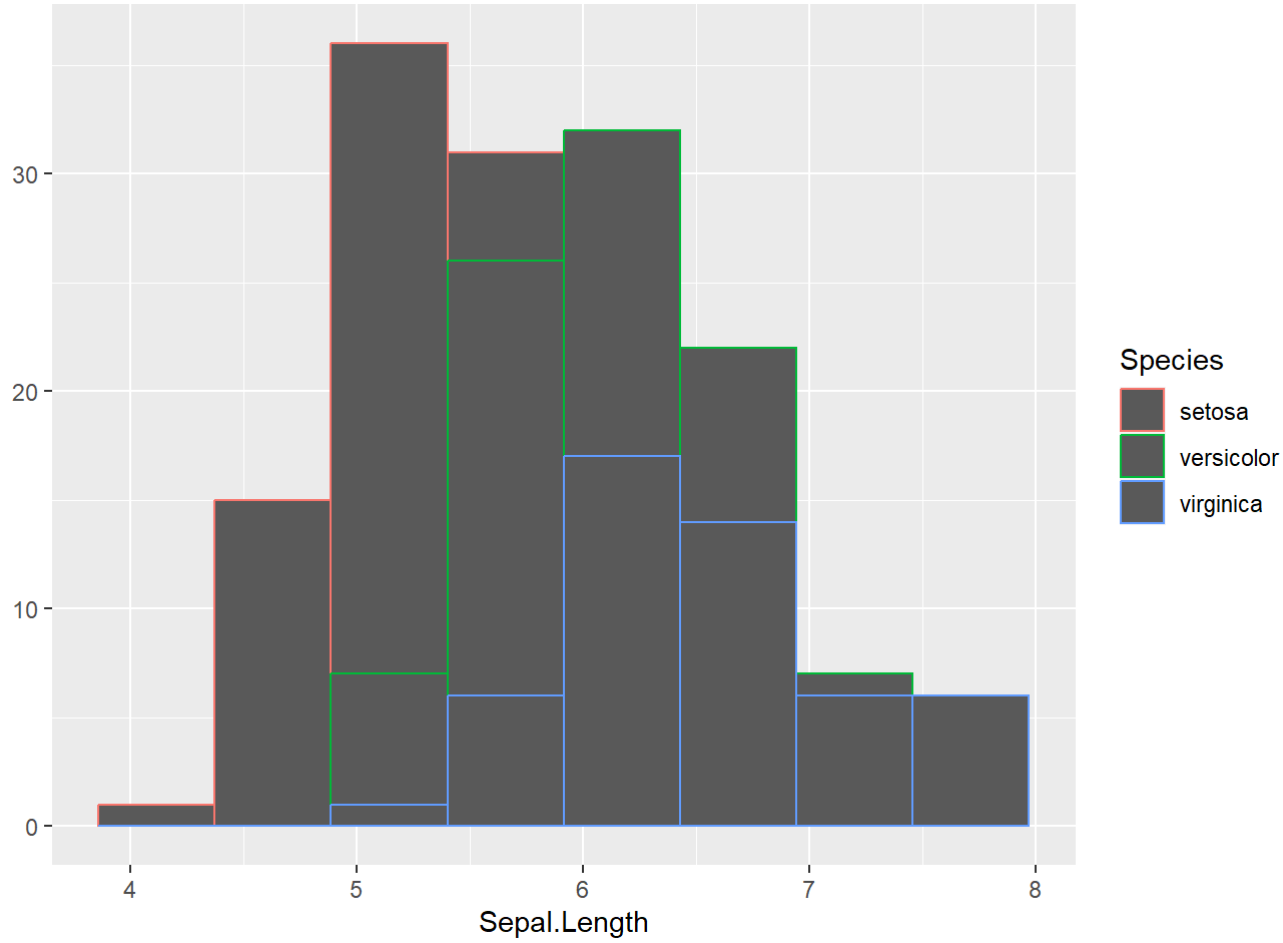
```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



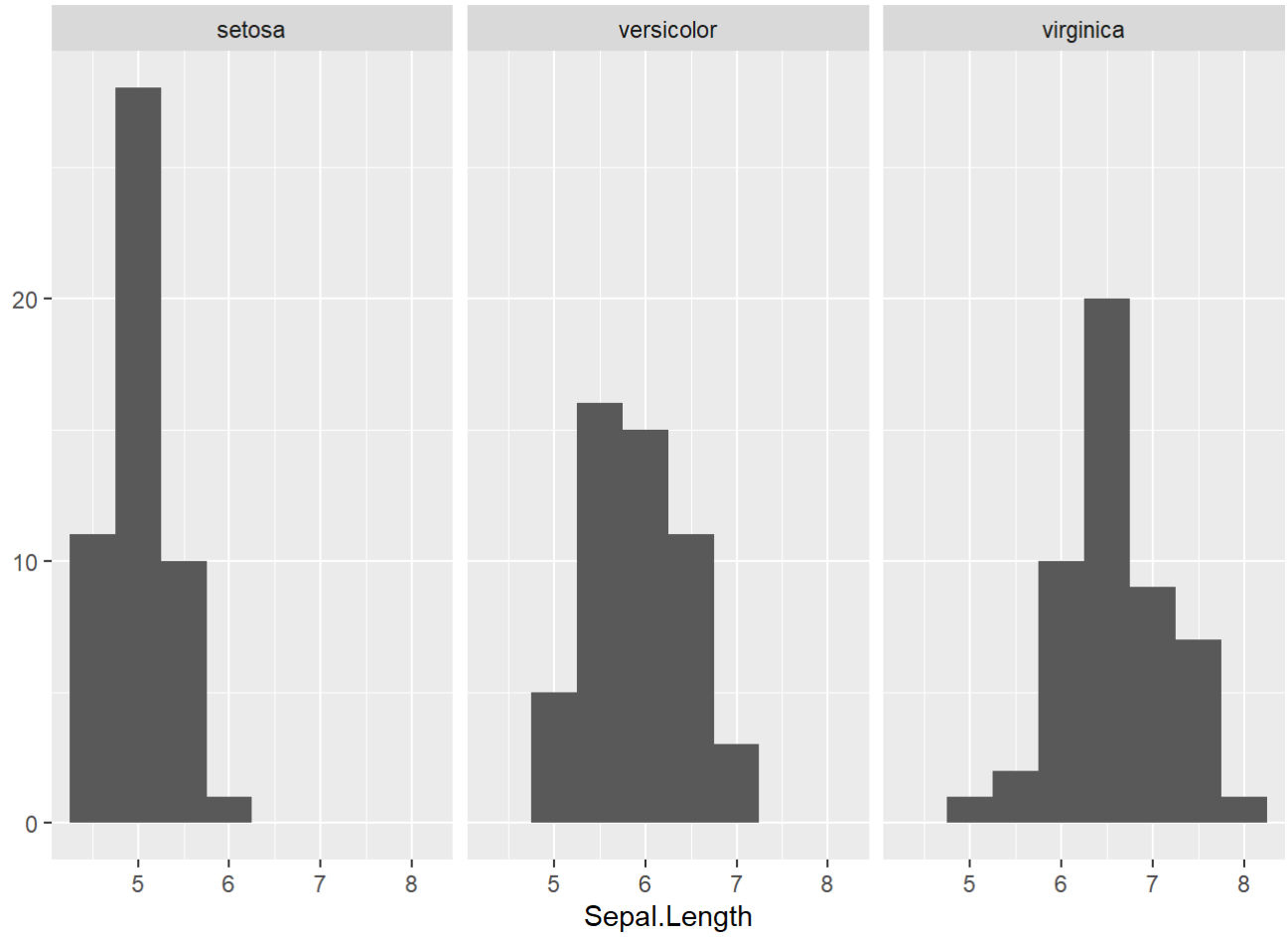

```
qplot(data = iris, Sepal.Length, fill=Species, bins = 8)  # defaul
t geom
```




```
qplot(data = iris, Sepal.Length, color=Species, bins = 8)
```



```
qplot(data = iris, Sepal.Length, facets = .~Species, binwidth = 0.5)
```

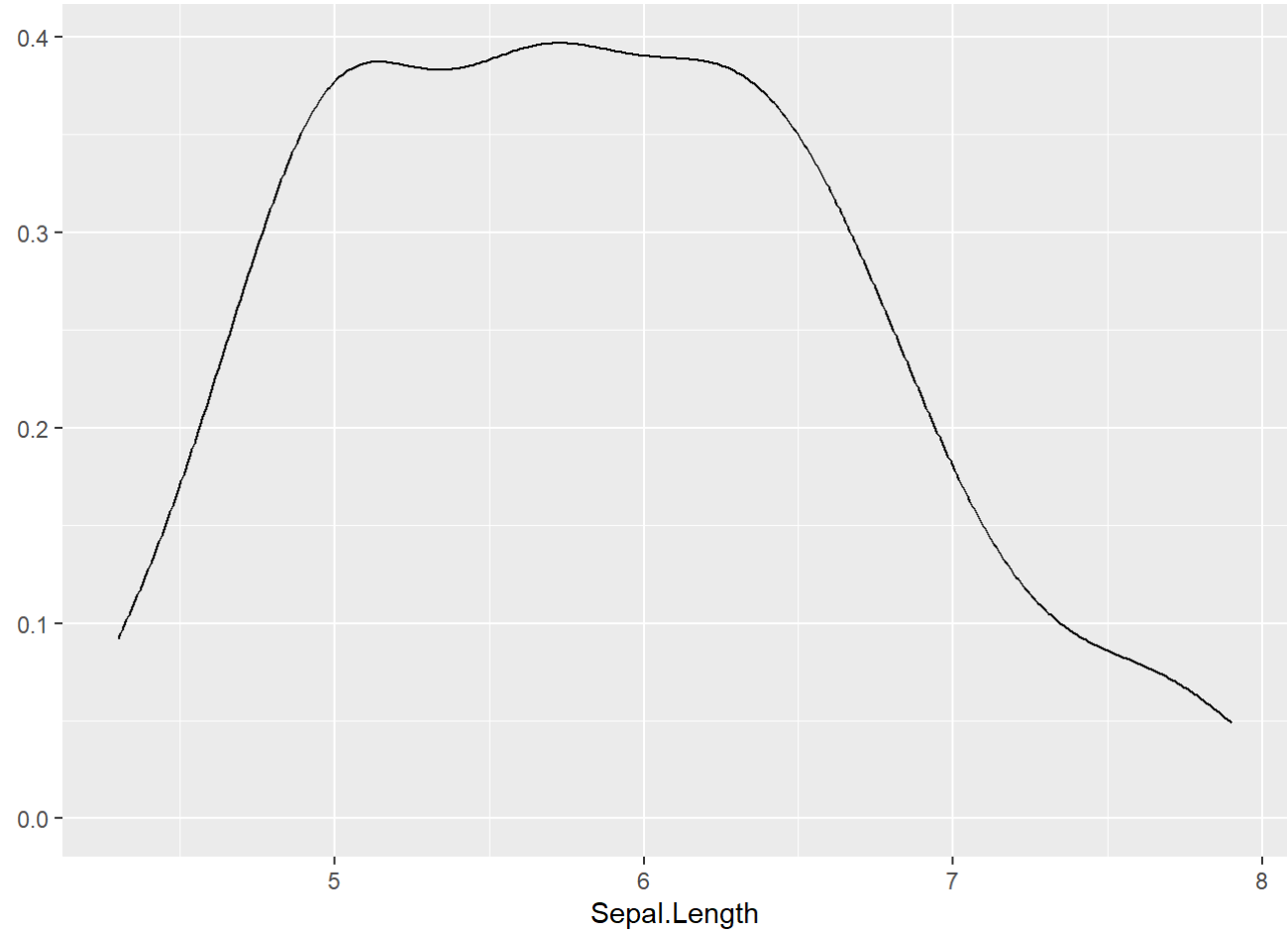



```
# grid.arrange(p1,p2,p3, ncol=3)
```

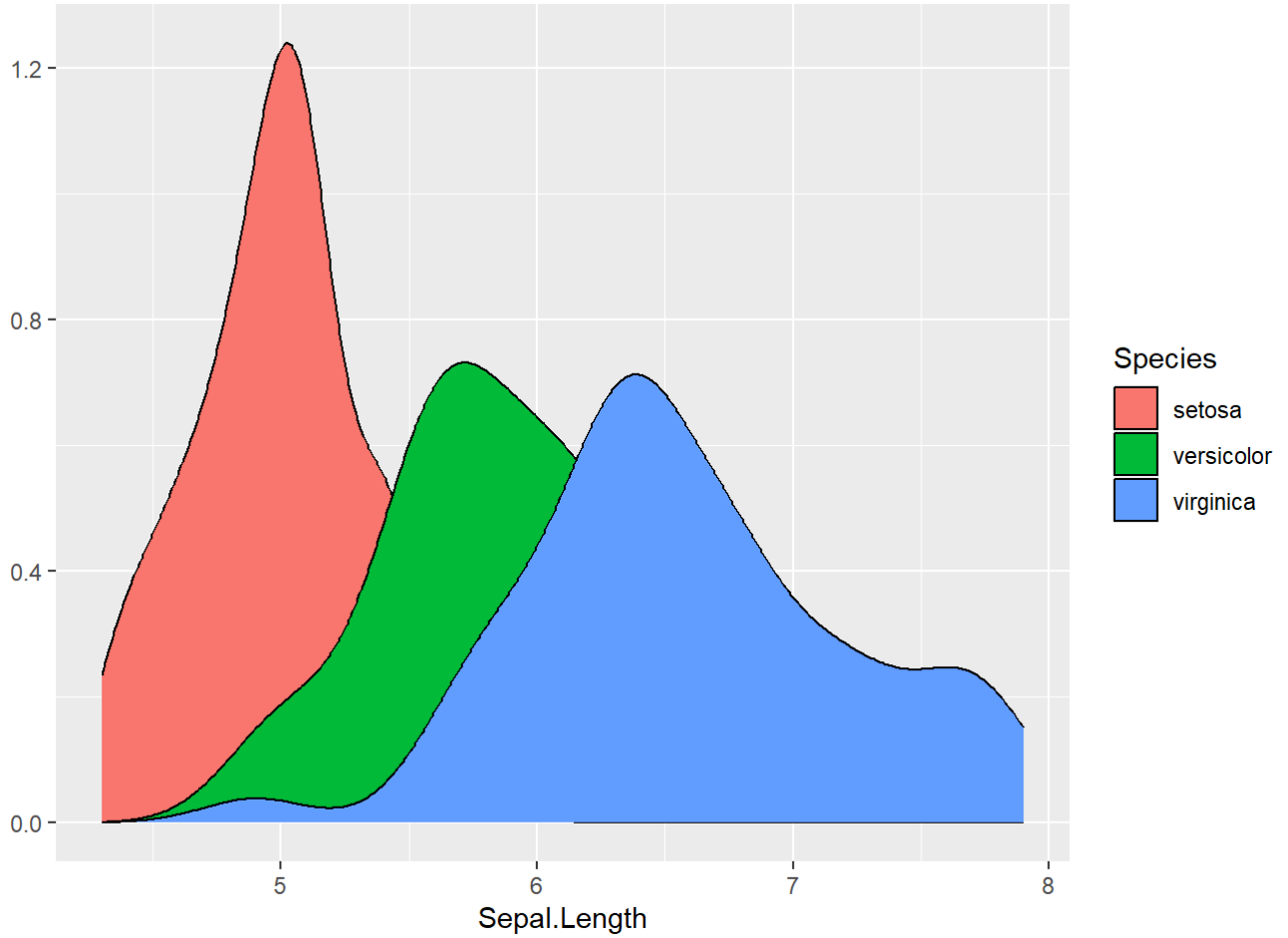
- Automatic color setting (color/fill are grouping variables in ggplot2)
- Faceting is similar to the conditioning function in Lattice

Density plot

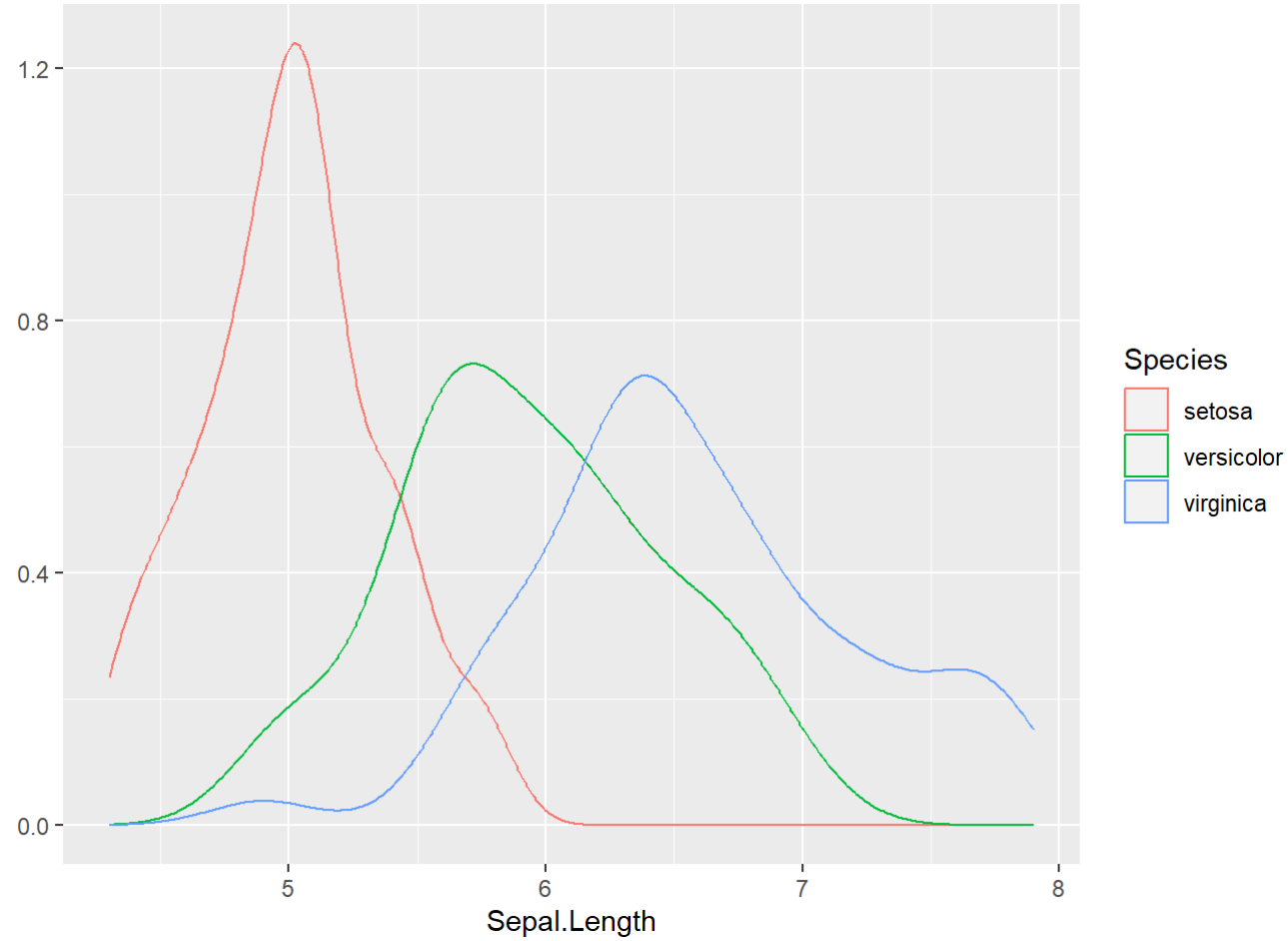
```
qplot(data = iris, Sepal.Length, geom = "density")
```




```
qplot(data = iris, Sepal.Length, geom = "density",  fill = Species)
```

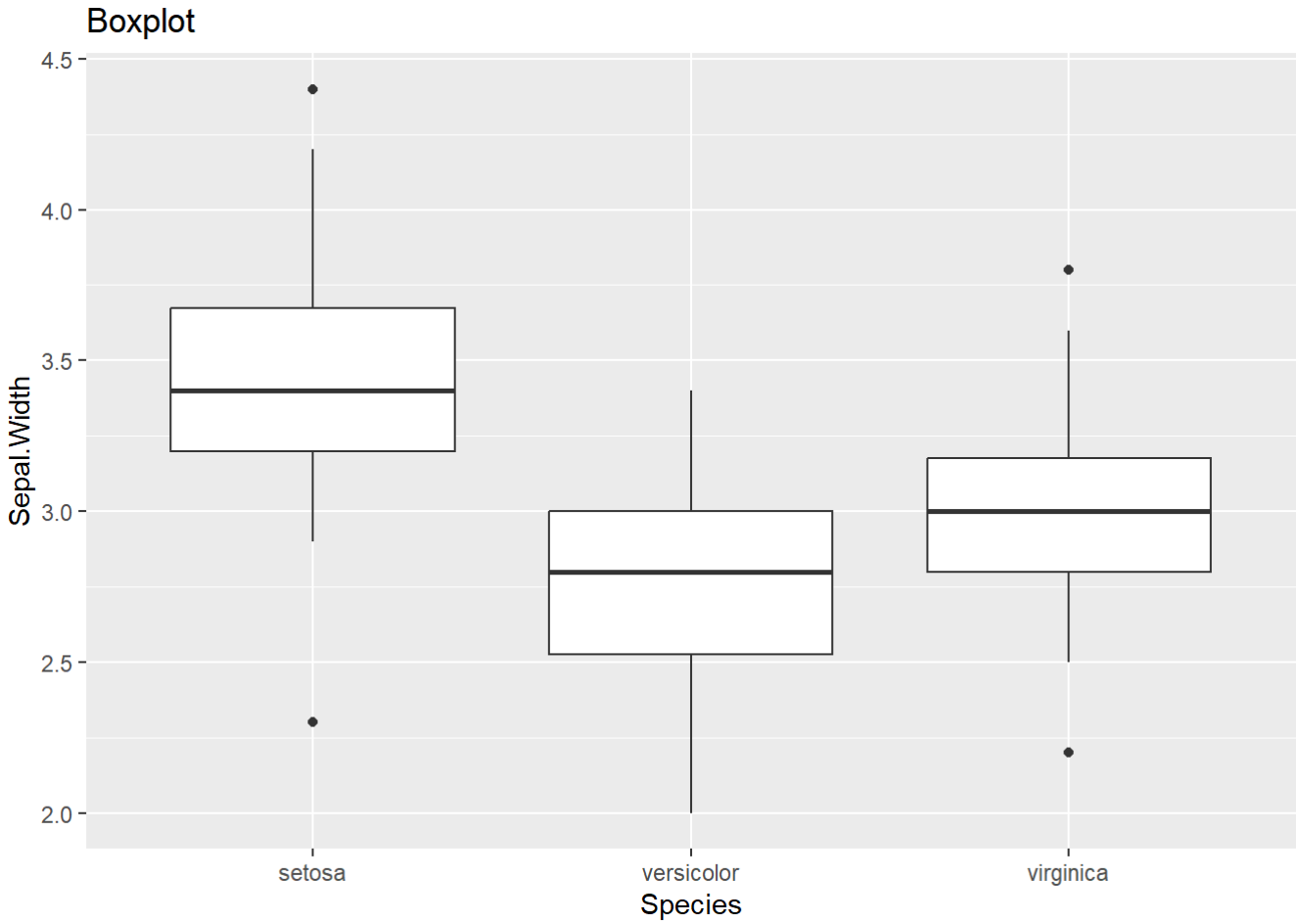



```
qplot(data = iris, Sepal.Length, geom = "density",  color = Species)
```

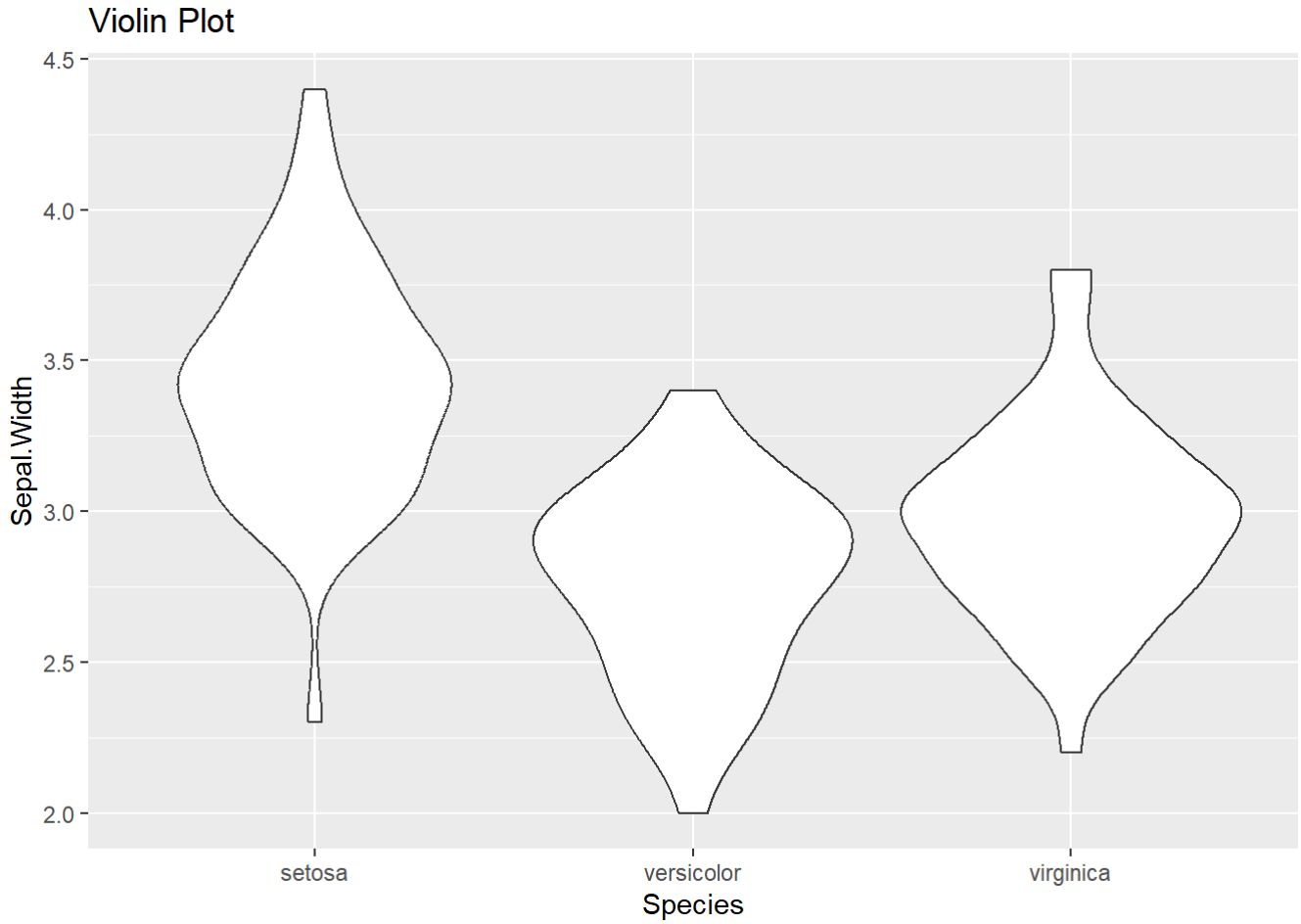


Boxplot with Grouping

```
qplot(data = iris, Species, Sepal.Width, geom="boxplot", main="Box plot")
```



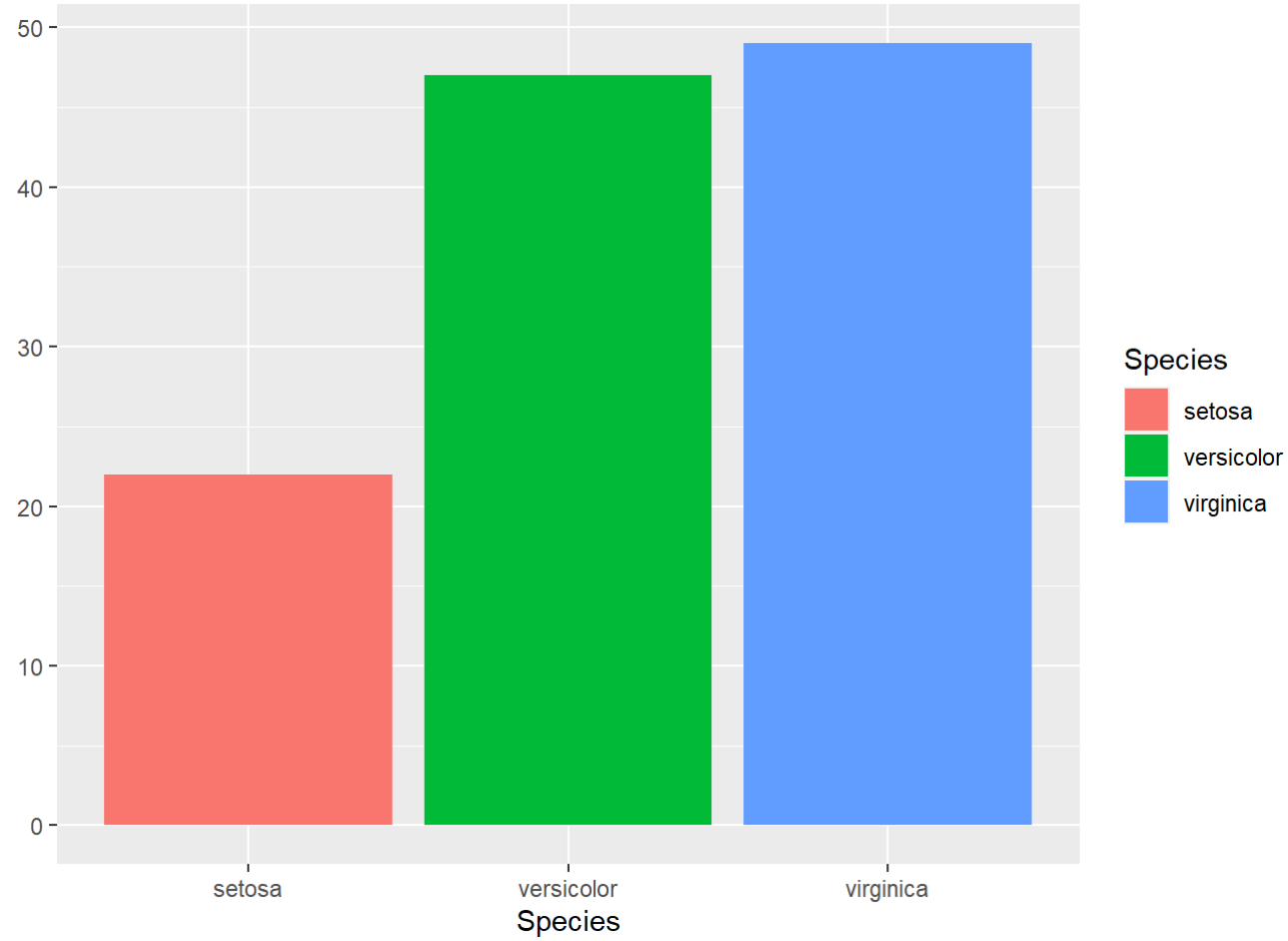

```
qplot(data = iris, Species, Sepal.Width, geom="violin", main="Violin Plot")
```



- Following data are x (grouping) and y (response) variables

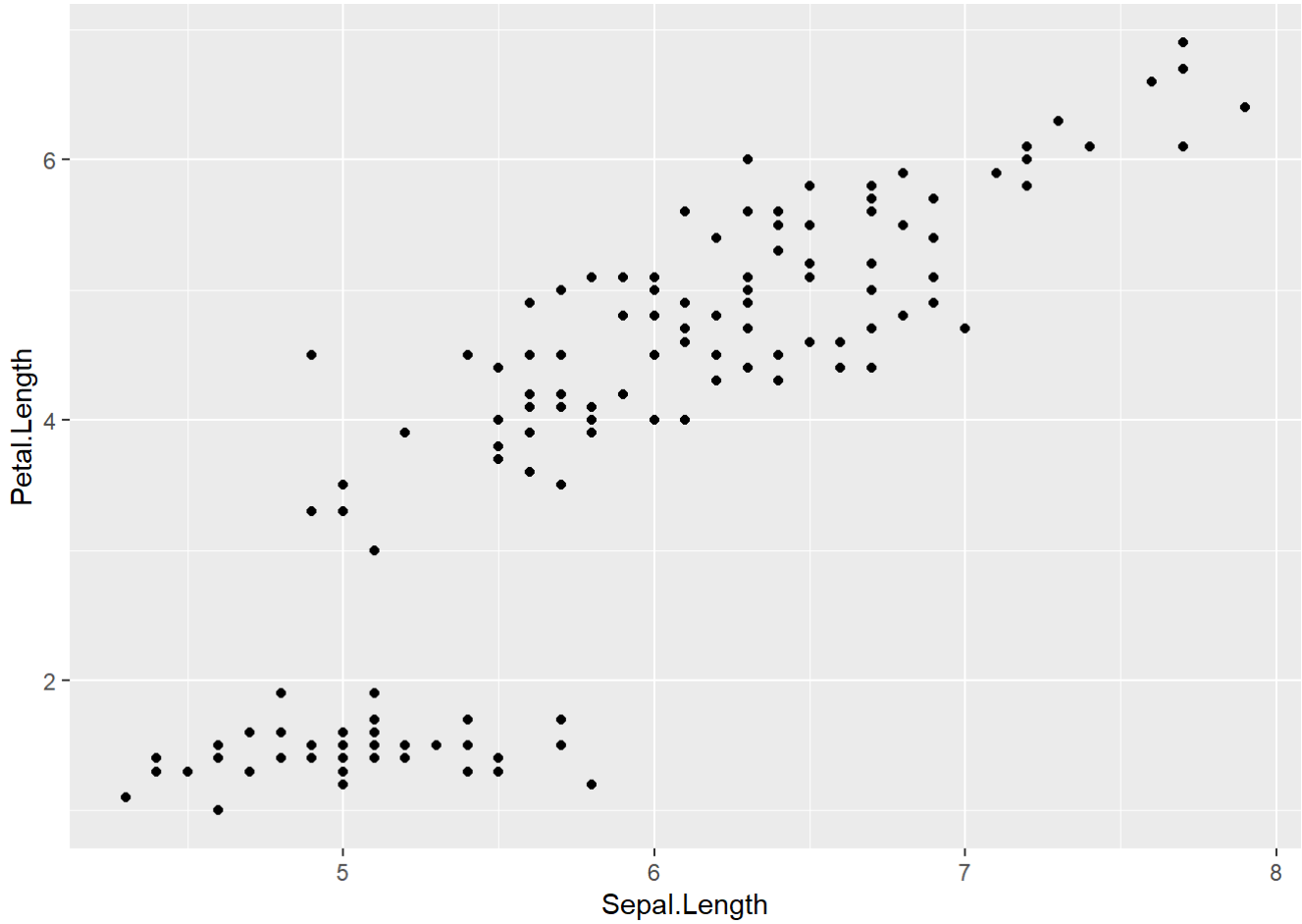
Bar plot for categoricla variables

```
qplot(data = subset(iris, Sepal.Length>5), Species, geom="bar",  fill=Species)
```



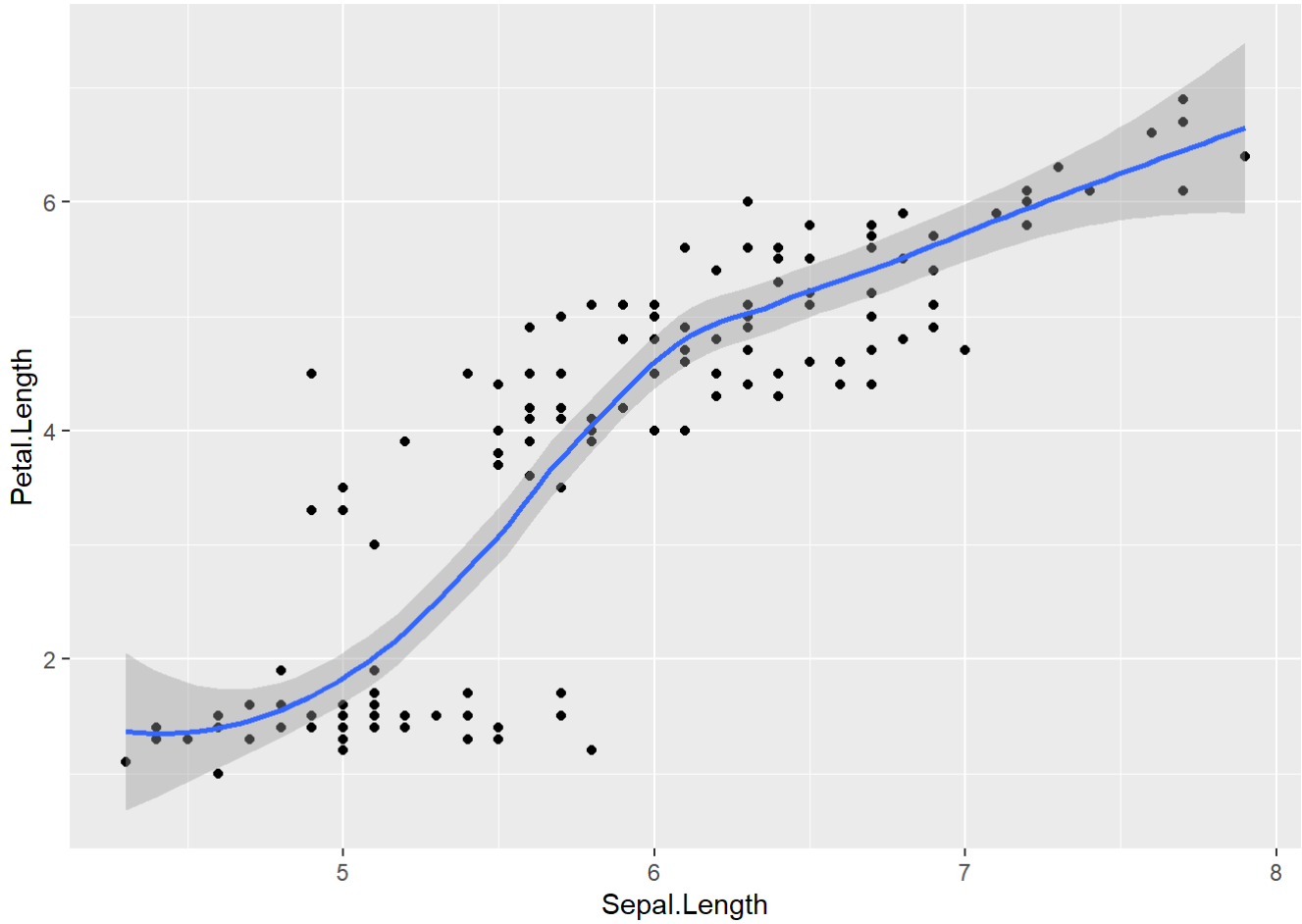
Scatter plot

```
qplot(data = iris, Sepal.Length, Petal.Length, geom = "point")
```



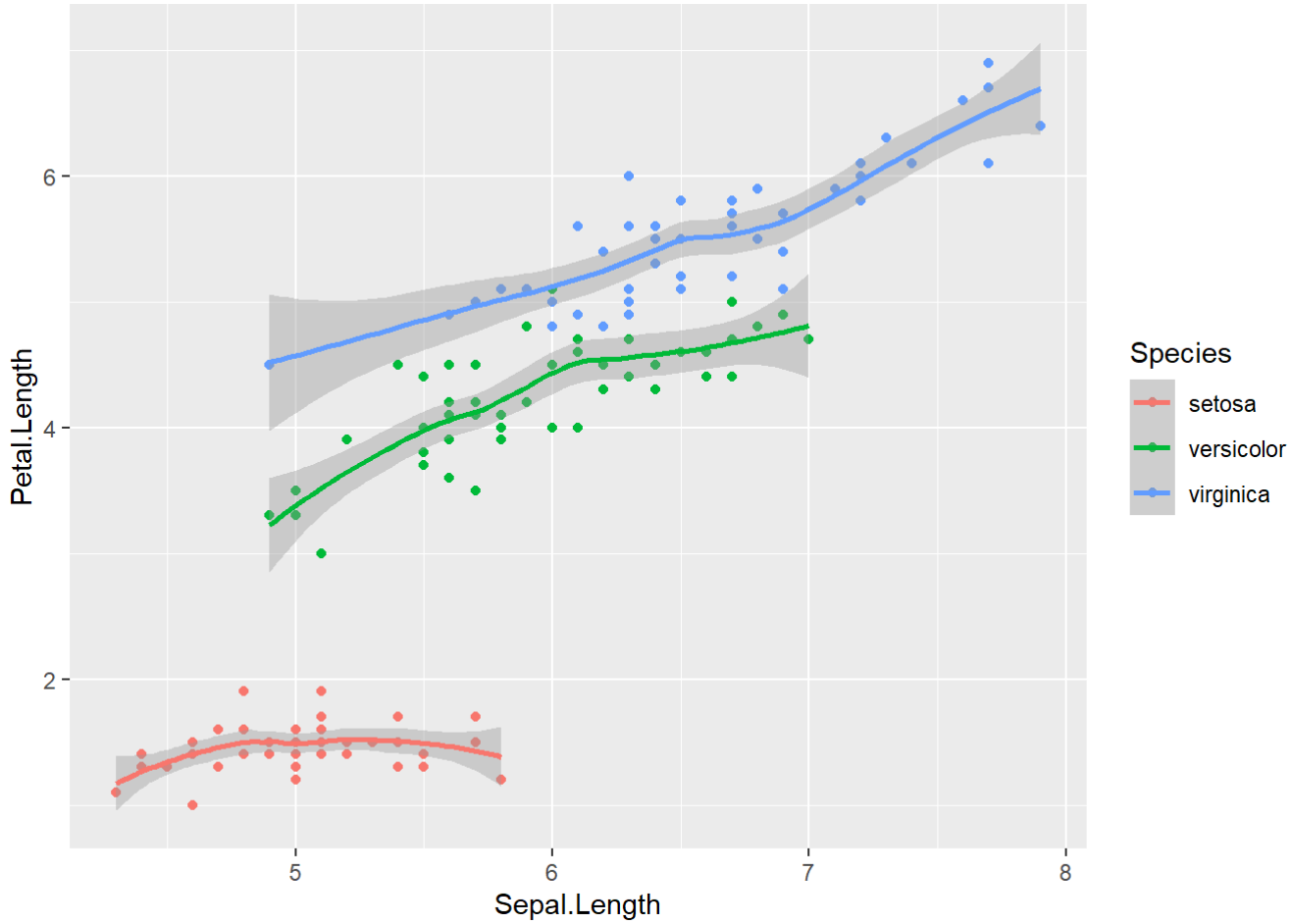

```
qplot(data = iris, Sepal.Length, Petal.Length, geom = c("point",
"smooth"))
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



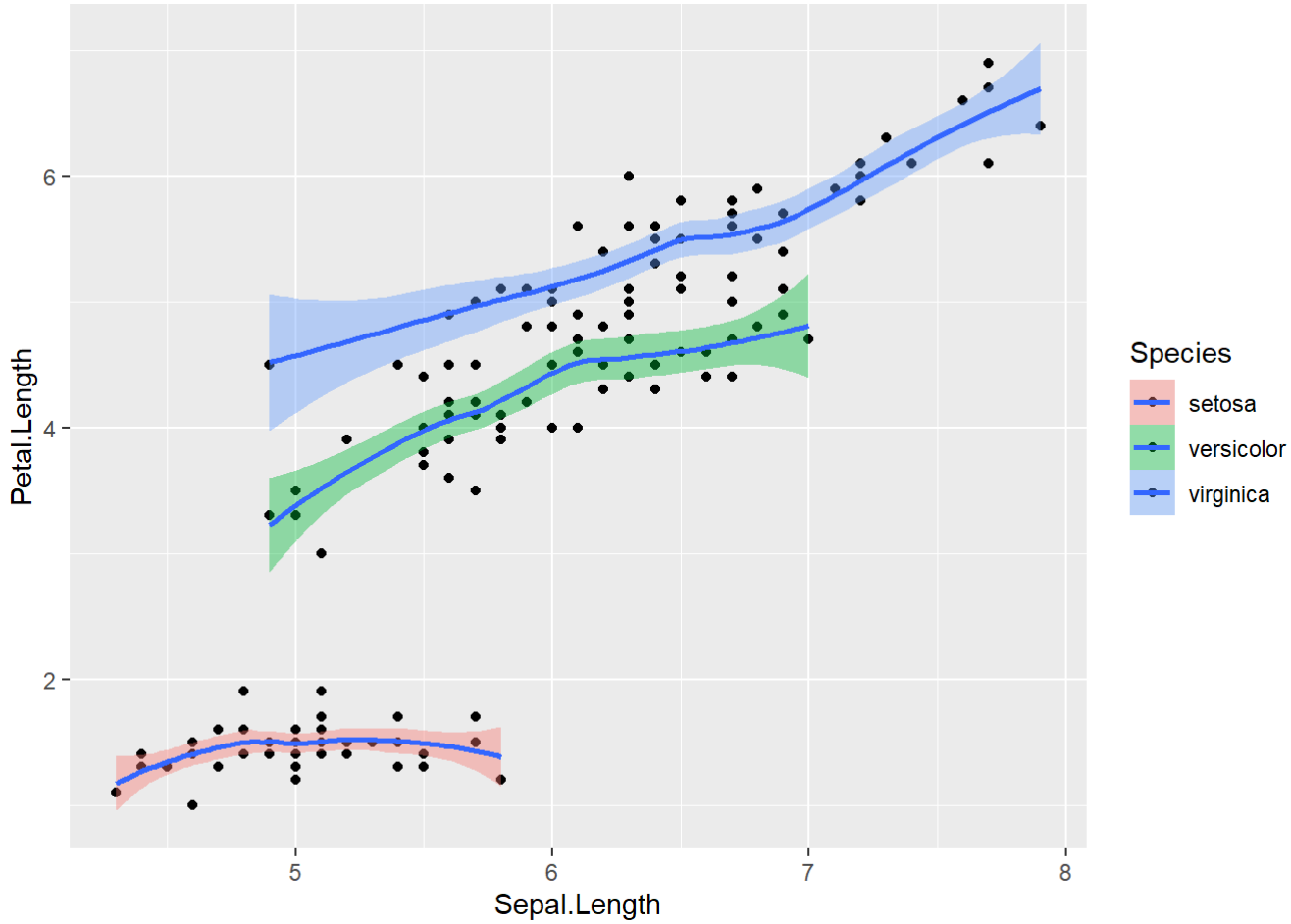

```
qplot(data = iris, Sepal.Length, Petal.Length, geom = c("point",
"smooth"), color = Species)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
qplot(data = iris, Sepal.Length, Petal.Length, geom = c("point",
"smooth"), fill = Species)
```

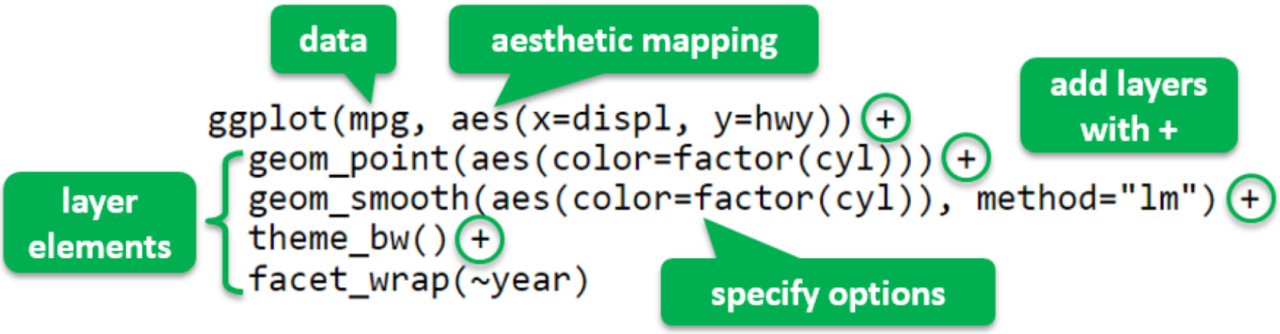
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



3.3 ggplot

Layer-by-layer Syntax

- ggplot() builds a plot layer by layer, with the syntax:

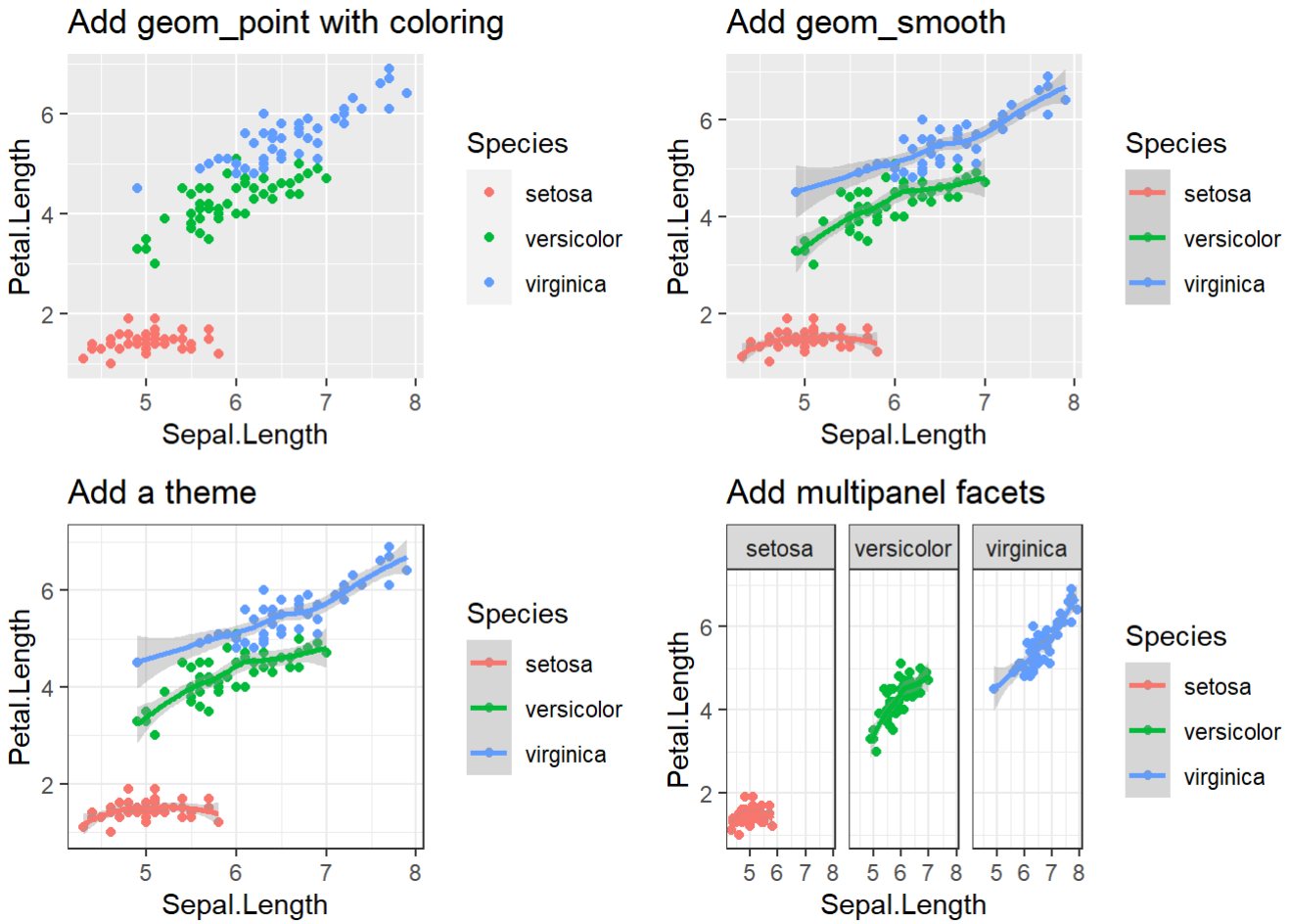


- ggplot() provides more control than qplot().

Layer-by-Layer Scatterplot

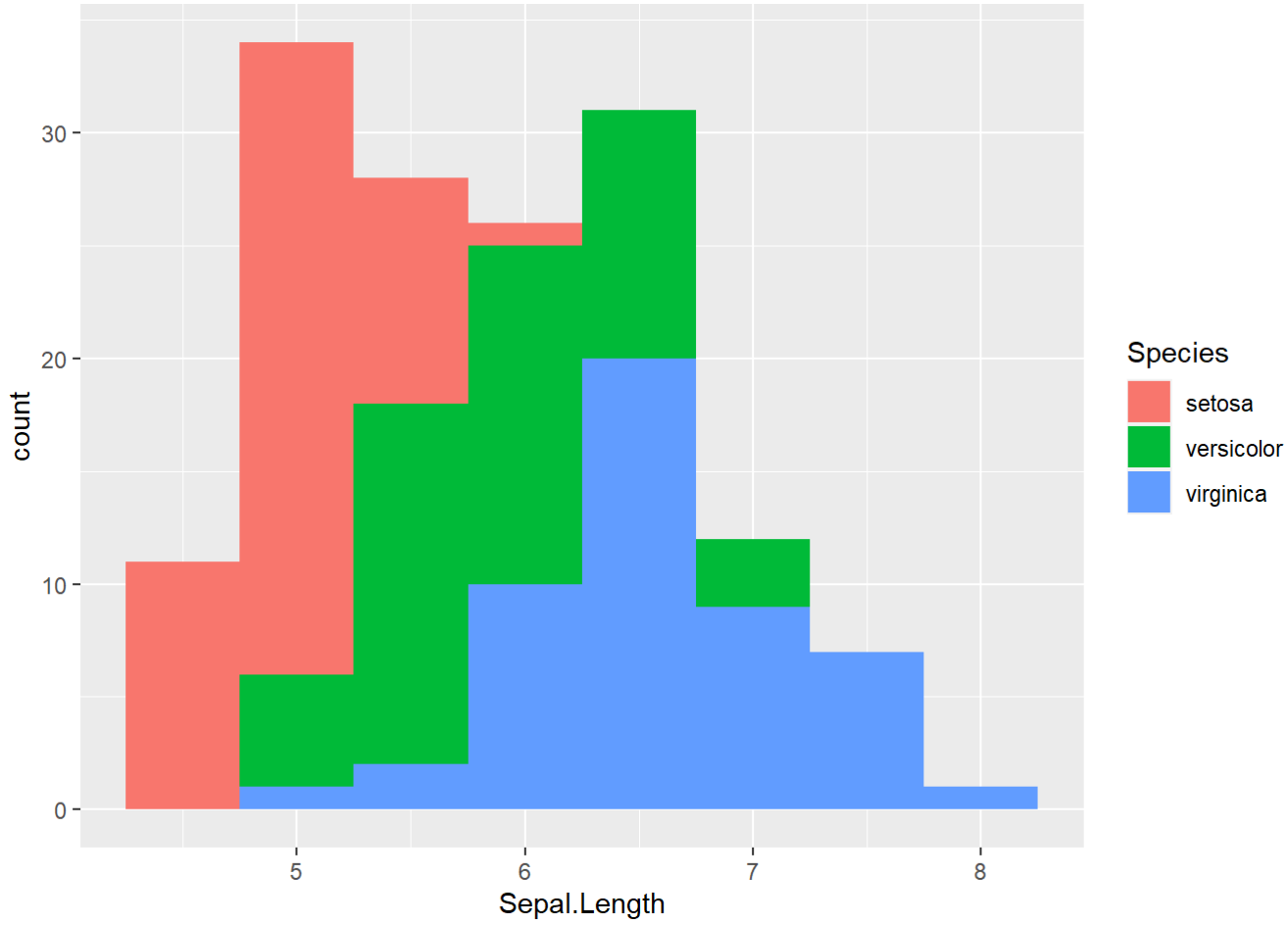
```
p0 = ggplot(iris, aes(x=Sepal.Length, y=Petal.Length))
p1 = p0 + geom_point(aes(color=Species)) + ggtitle("Add geom_point with coloring")
p2 = p1 + geom_smooth(aes(color=Species)) + ggtitle("Add geom_smooth")
p3 = p2 + theme_bw() + ggtitle("Add a theme")
p4 = p3 + facet_wrap(~Species) + ggtitle("Add multipanel facets")
grid.arrange(p1, p2, p3, p4, nrow=2, ncol=2)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

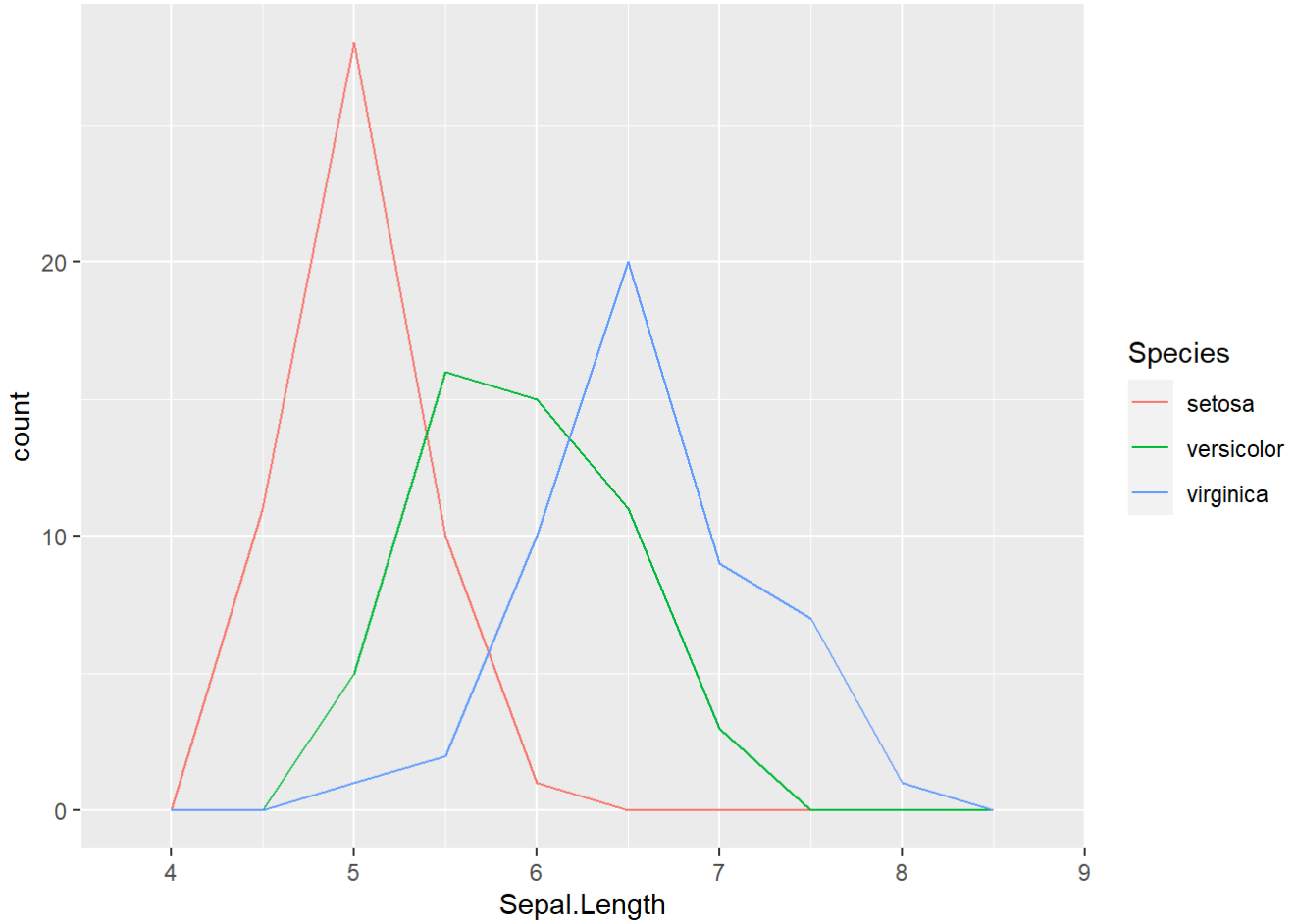


Histogram, Freqpoly and Density plots

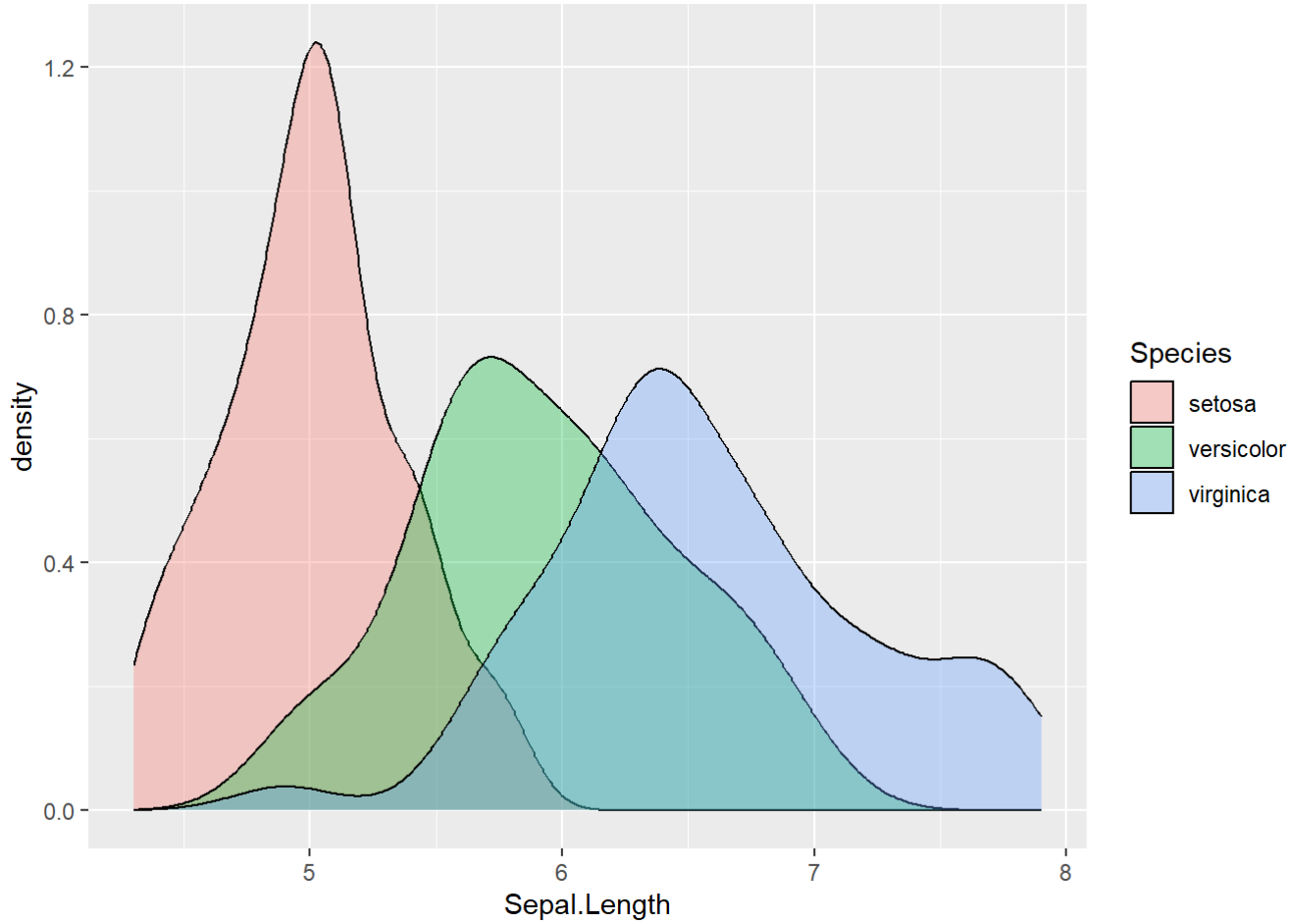
```
ggplot(iris, aes(Sepal.Length, fill=Species)) + geom_histogram(binwidth = 0.5)
```




```
ggplot(iris, aes(Sepal.Length, color=Species)) + geom_freqpoly(bin
width = 0.5)
```

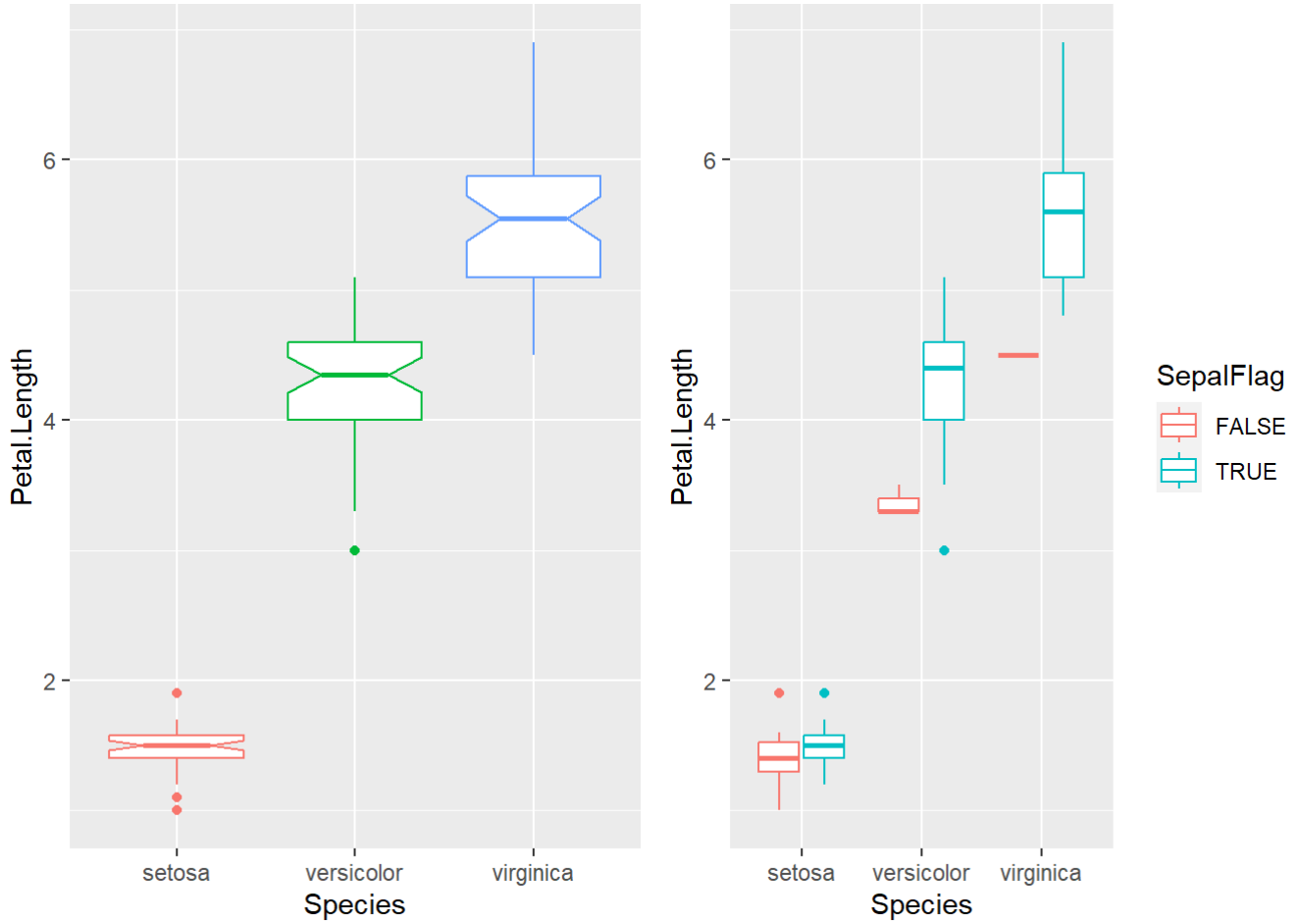



```
ggplot(iris, aes(Sepal.Length, fill=Species)) + geom_density(alpha=1/3)
```



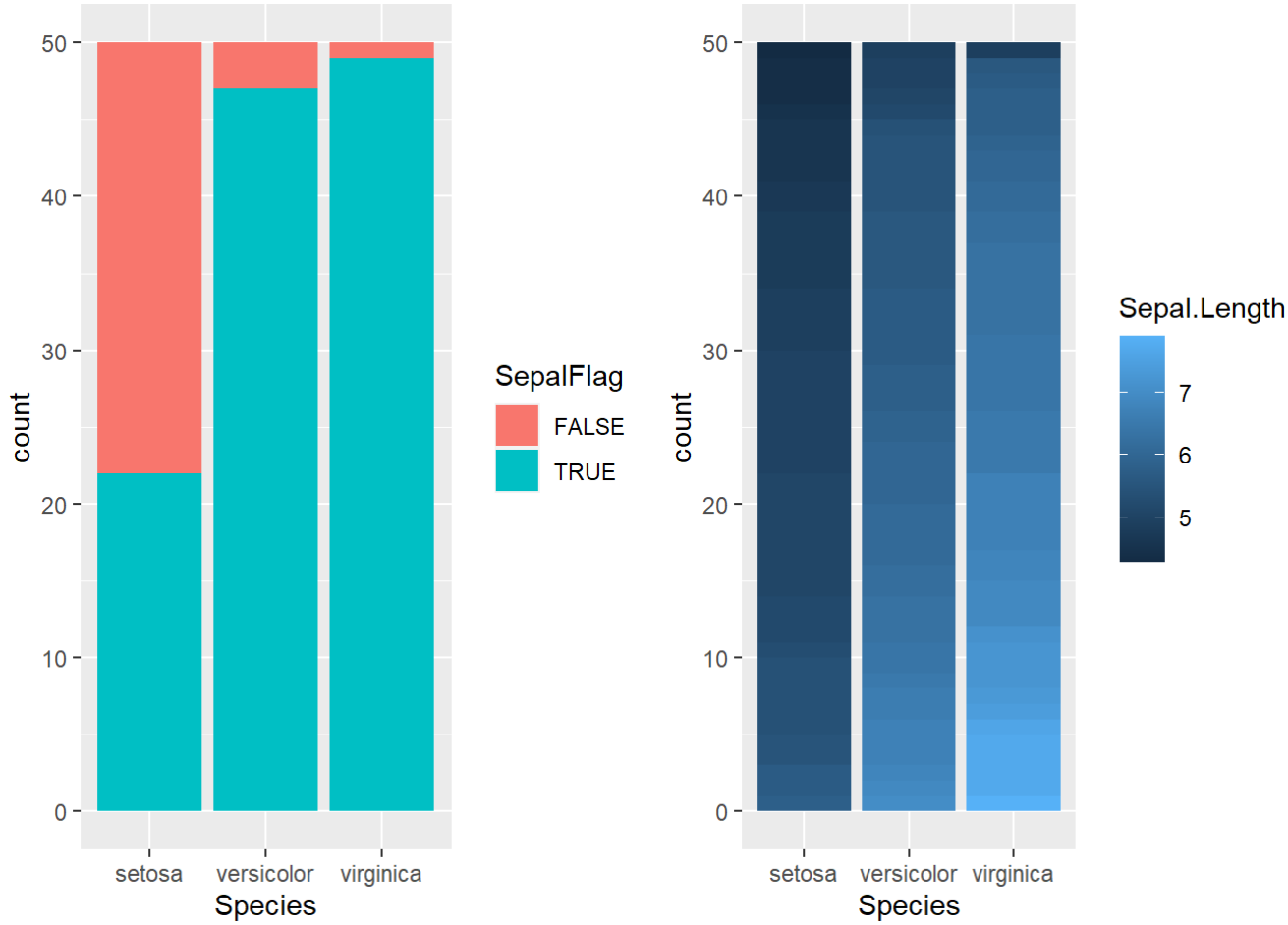
Boxplot

```
iris$SepalFlag = iris$Sepal.Length>5
p1 = ggplot(iris, aes(x=Species, y=Petal.Length, color=Species)) +
  geom_boxplot(notch = T, show.legend = F)
p2 = ggplot(iris, aes(x=Species, y=Petal.Length, color=SepalFlag))
+ geom_boxplot()
grid.arrange(p1, p2, ncol=2)
```



Bar Chart

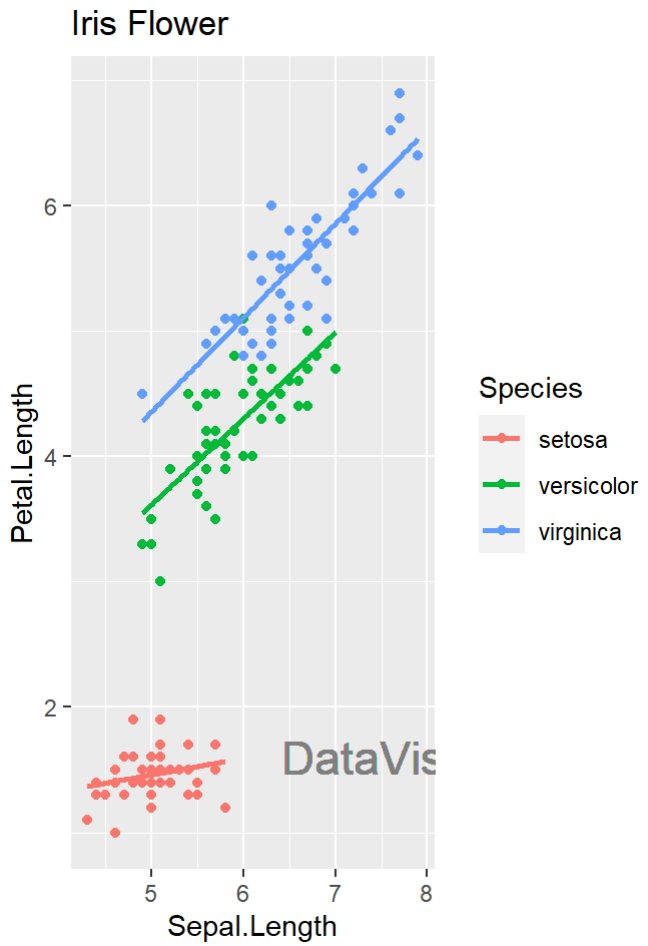
```
p1 = ggplot(iris, aes(Species, fill=SepalFlag)) + geom_bar(position = "stack")
p2 = ggplot(iris, aes(Species, fill=Sepal.Length, group=Sepal.Length)) + geom_bar()
grid.arrange(p1, p2, ncol=2)
```

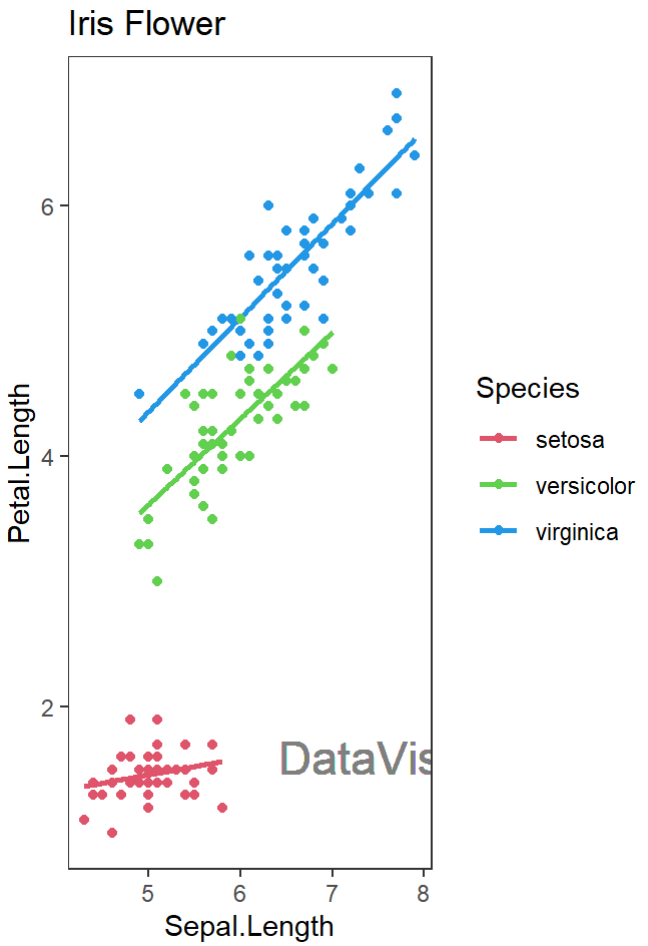


3.4 Options and themes

```
p1 = ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, colour=Species)) +
  geom_point() + stat_smooth(method="lm", se=F) +
  labs(title="Iris Flower") +
  geom_text(aes(7.3, 1.6), label="DataVis", size=6, color="gray50")
p2 = p1 + geom_point() + theme_bw() +
  theme(panel.grid= element_blank()) + scale_color_manual(values = c(2,3,4))
grid.arrange(p1, p2, ncol=2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



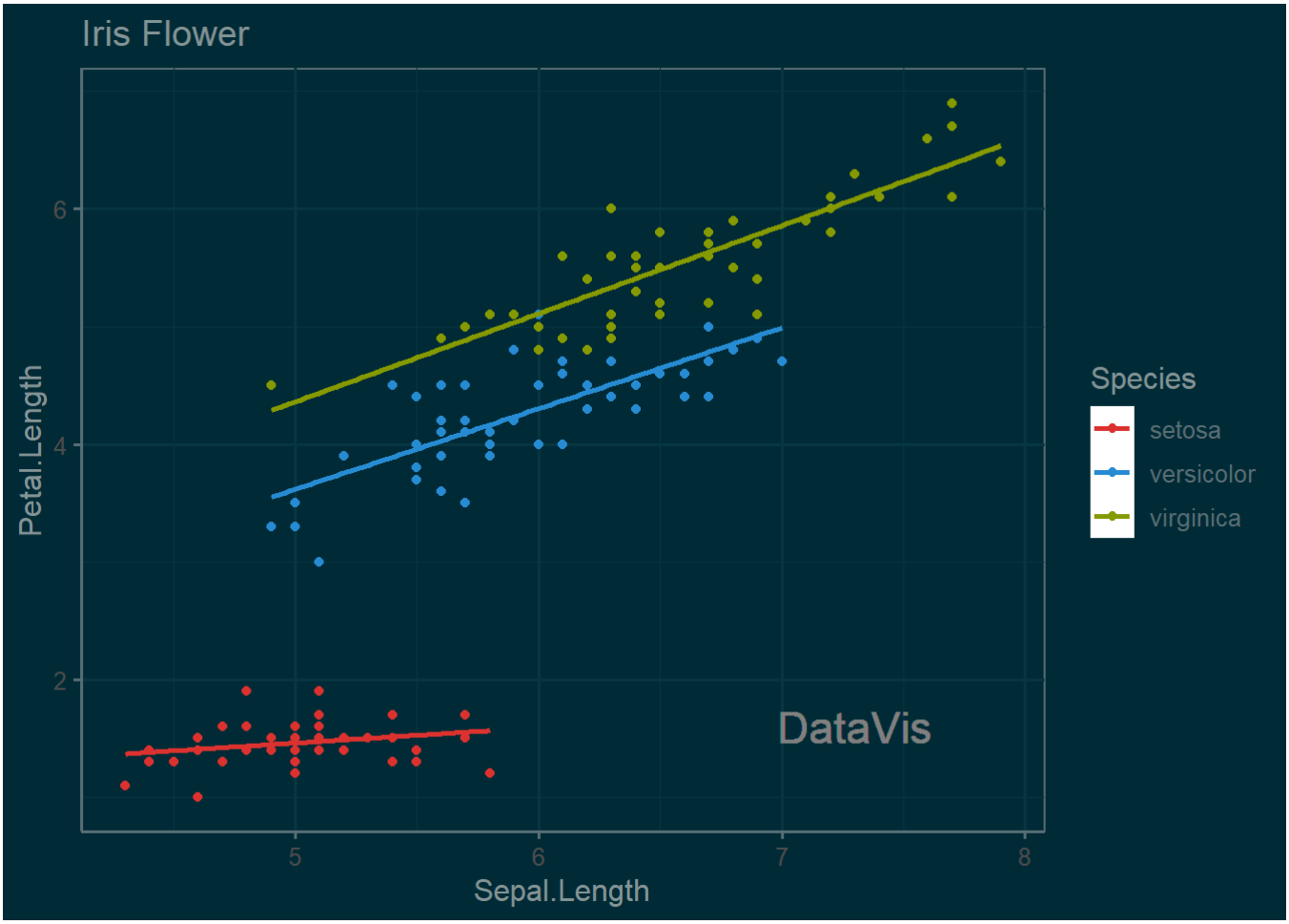



```
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.0.5
```

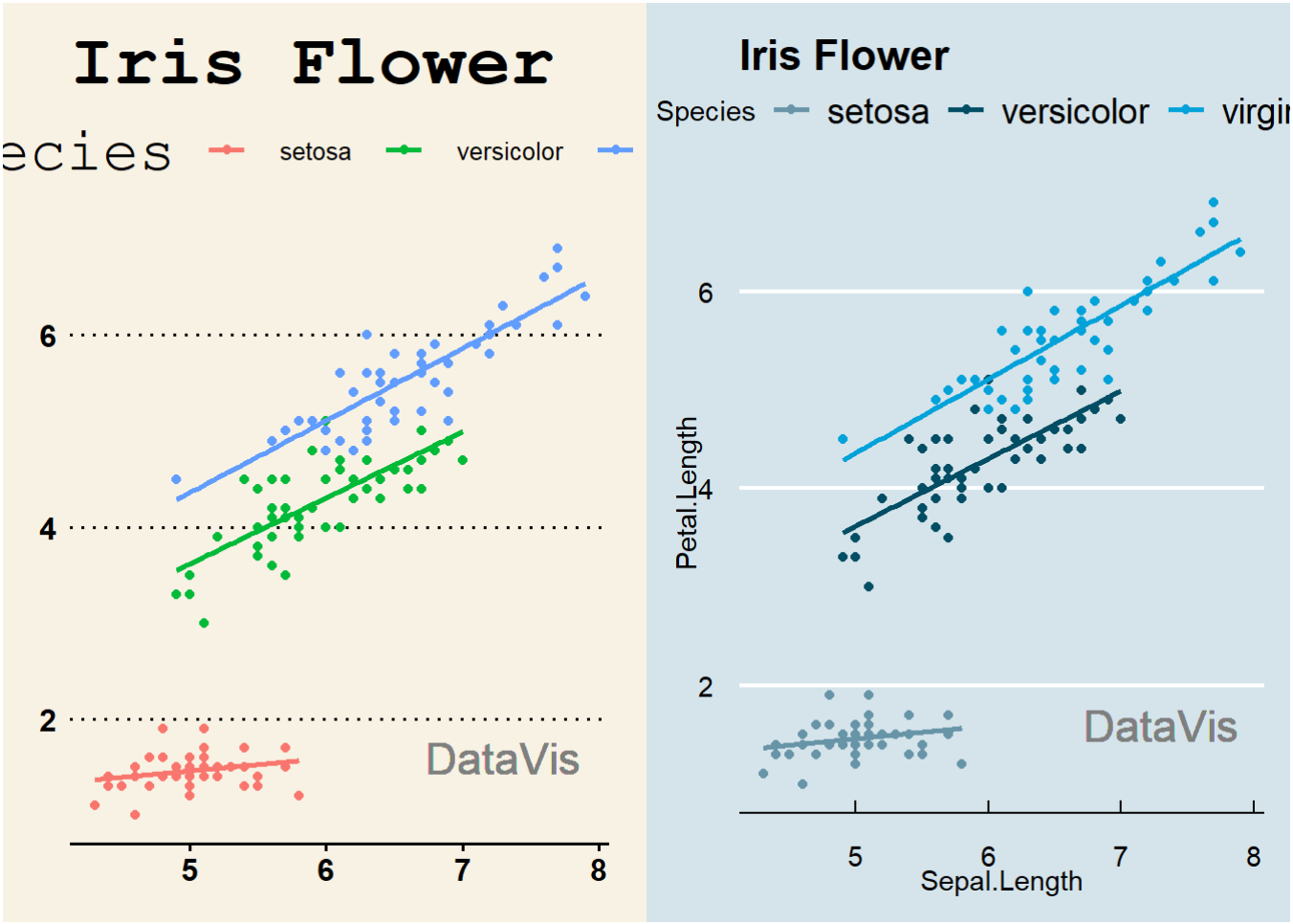
```
p1 +  theme_solarized(light = F) + scale_colour_solarized("red")
# "Solarized Theme"
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
p2 = p1 + theme_wsj()  #  "WSJ Theme"
p3 = p1 + theme_economist() + scale_colour_economist()  # "Economi
st Theme"
grid.arrange(p2, p3, ncol=2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



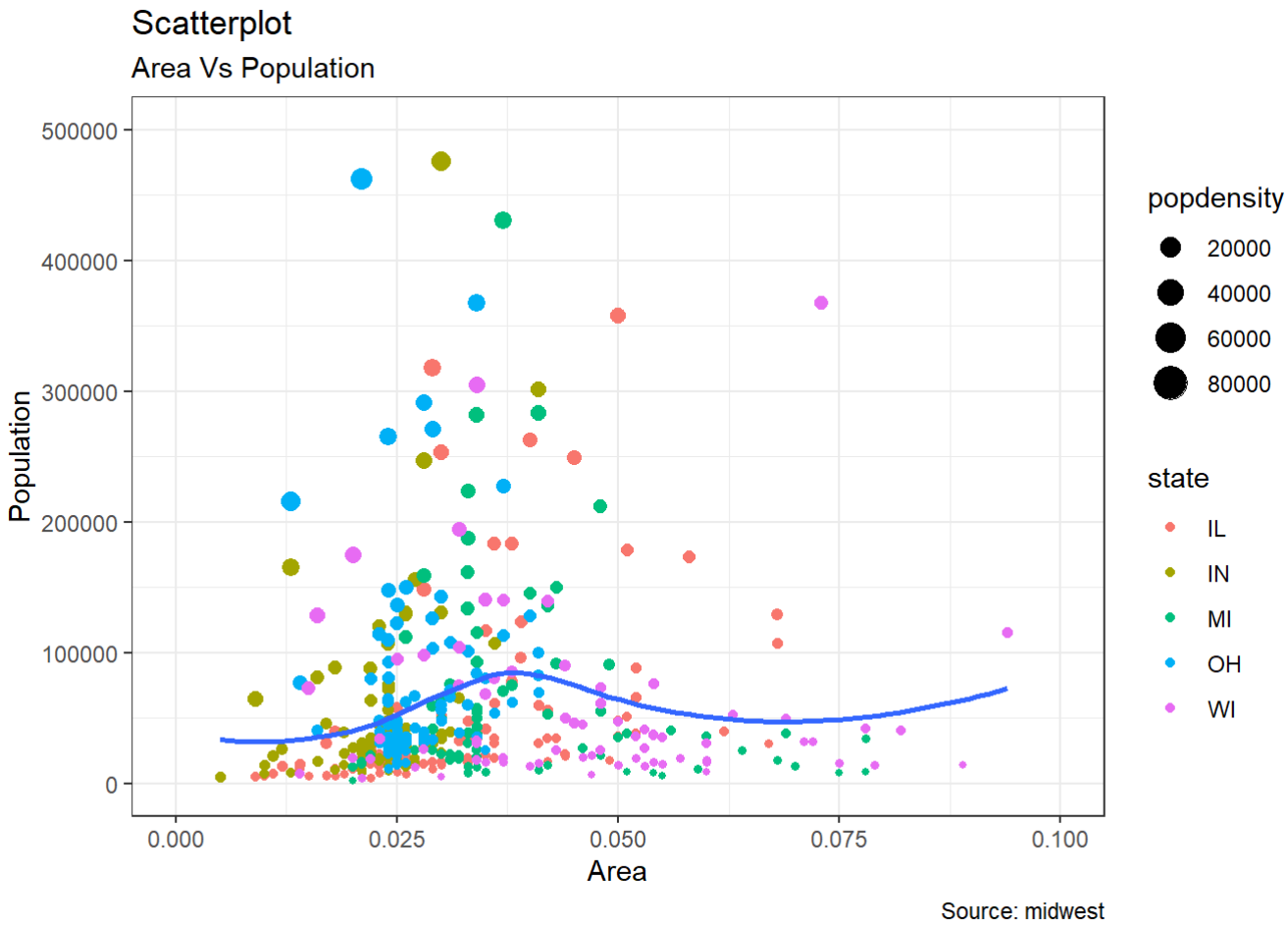
More examples

Selective examples from Top 50 ggplot2 Visualizations (http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html)

Scatter plot

```
# install.packages("ggplot2")
# load package and data
options(scipen=999)  # turn-off scientific notation like 1e+48
library(ggplot2)
theme_set(theme_bw())  # pre-set the bw theme.
data("midwest", package = "ggplot2")
# midwest <- read.csv("http://goo.gl/G1K41K")  # bkup data source

# Scatterplot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) +
  xlim(c(0, 0.1)) +
  ylim(c(0, 500000)) +
  labs(subtitle="Area Vs Population",
       y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")

plot(gg)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Marginal Histogram / Boxplot

```
library(ggplot2)
library(ggExtra)
```

```
## Warning: package 'ggExtra' was built under R version 4.0.5
```

```
data(mpg, package="ggplot2")
# mpg <- read.csv("http://goo.gl/uEeRGu")

# Scatterplot
theme_set(theme_bw())  # pre-set the bw theme.
mpg_select <- mpg[mpg$hwy >= 35 & mpg$cty > 27, ]
g <- ggplot(mpg, aes(cty, hwy)) +
  geom_count() +
  geom_smooth(method="lm", se=F)

ggMarginal(g, type = "histogram", fill="transparent")
```

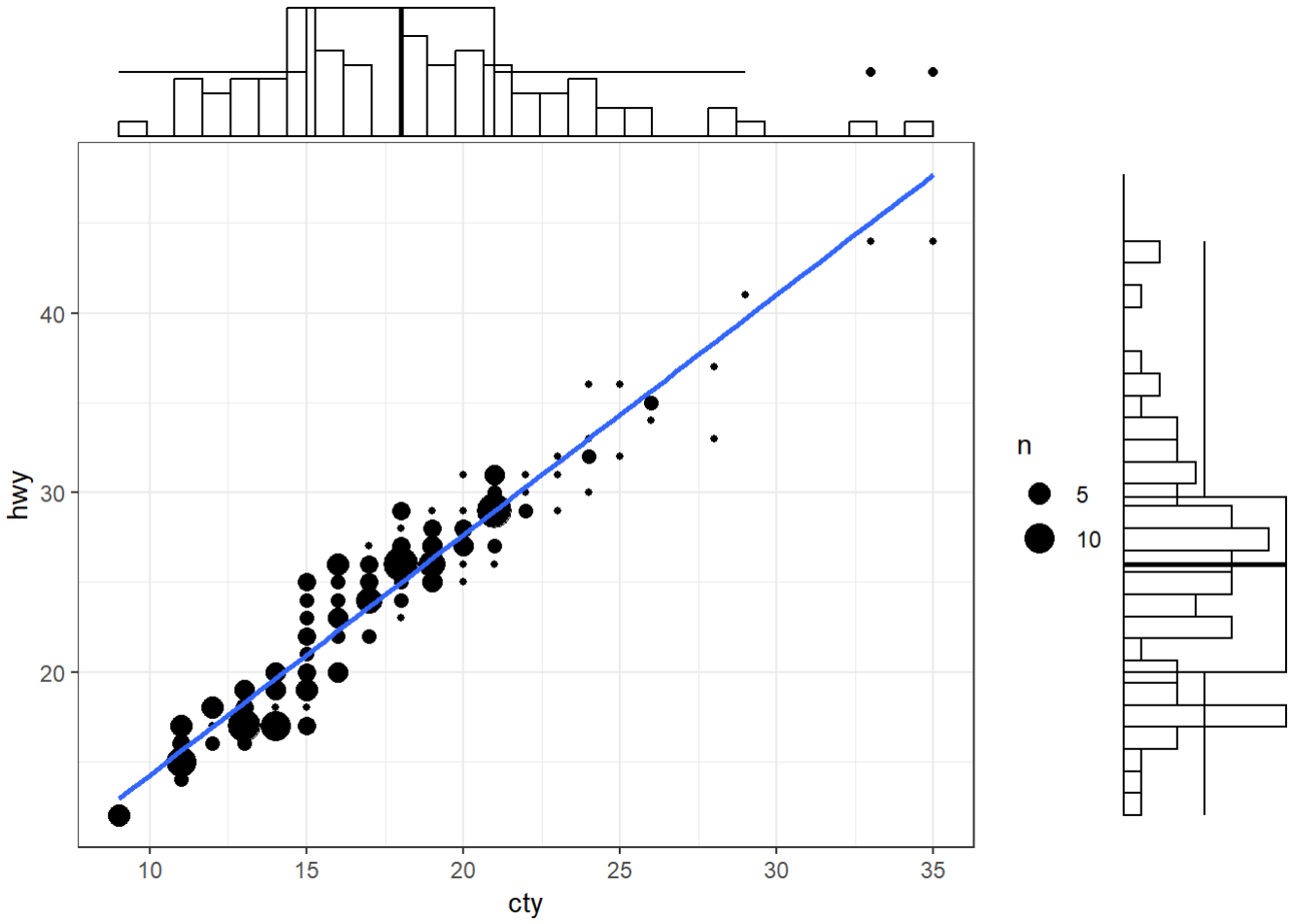
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```

```
ggMarginal(g, type = "boxplot", fill="transparent")
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Continuous x aesthetic
## i did you forget `aes(group = ...)`?
```

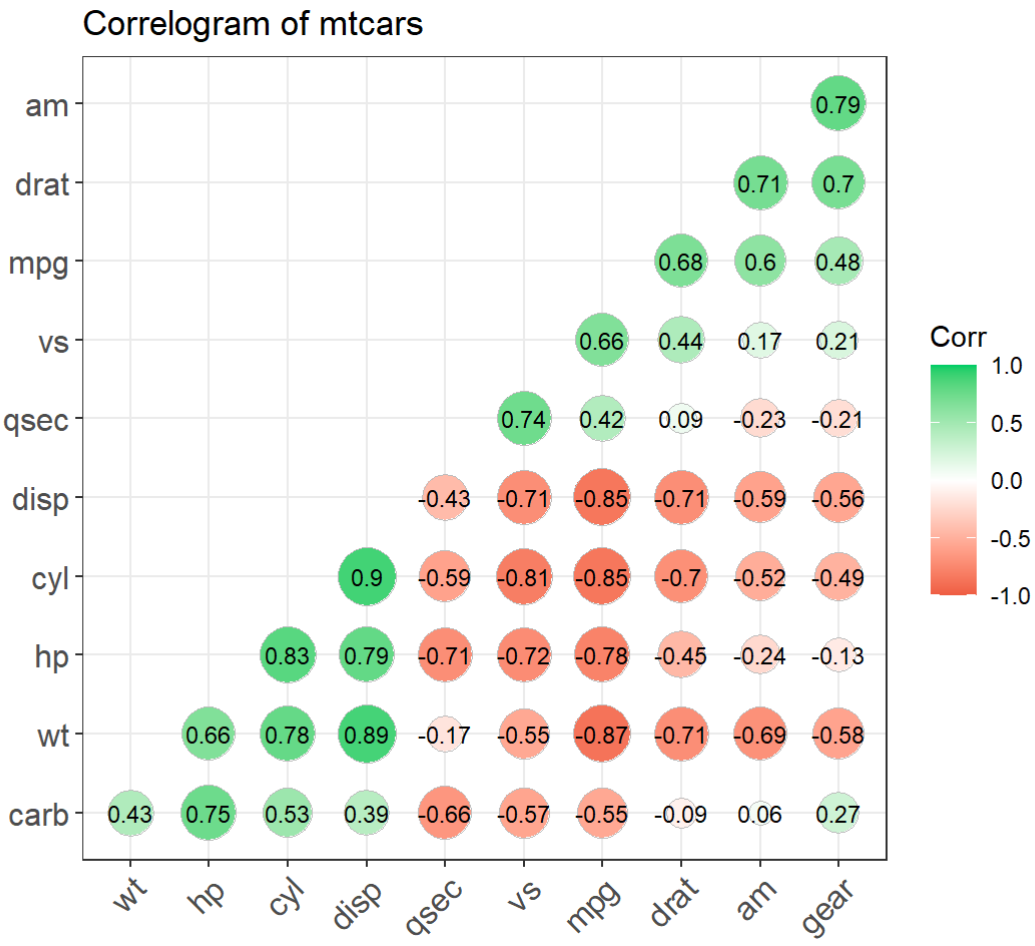
```
## Warning: Continuous x aesthetic
## i did you forget `aes(group = ...)`?
```




```
# ggMarginal(g, type = "density", fill="transparent")
```

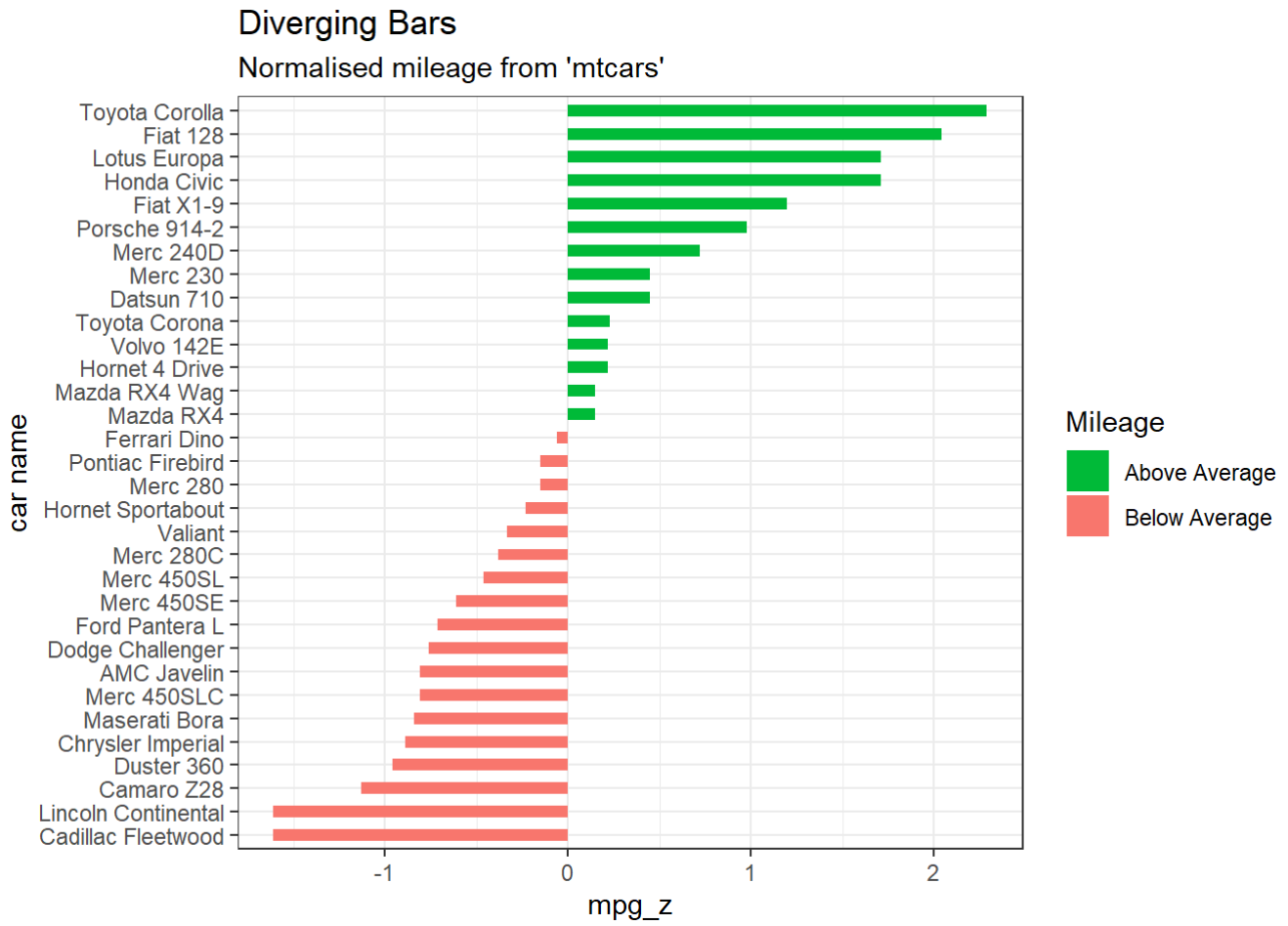
Correlogram

```
library(ggplot2)
library(ggcorrplot)

# Correlation matrix
data(mtcars)
corr <- round(cor(mtcars), 2)

# Plot
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of mtcars",
           ggtheme=theme_bw)
```



Diverging bars

```
library(ggplot2)
theme_set(theme_bw())

# Data Prep
data("mtcars")  # load data
mtcars$`car name` <- rownames(mtcars)  # create new column for car names
mtcars$mpg_z <- round((mtcars$mpg - mean(mtcars$mpg))/sd(mtcars$mpg), 2)  # compute normalized mpg
mtcars$mpg_type <- ifelse(mtcars$mpg_z < 0, "below", "above")  # above / below avg flag
mtcars <- mtcars[order(mtcars$mpg_z), ]  # sort
mtcars$`car name` <- factor(mtcars$`car name`, levels = mtcars$`car name`)  # convert to factor to retain sorted order in plot.

# Diverging Barcharts
ggplot(mtcars, aes(x=`car name`, y=mpg_z, label=mpg_z)) +
  geom_bar(stat='identity', aes(fill=mpg_type), width=.5)  +
  scale_fill_manual(name="Mileage",
                    labels = c("Above Average", "Below Average"),
                    values = c("above"="#00ba38", "below"="#f8766d")) +
  labs(subtitle="Normalised mileage from 'mtcars'",
       title= "Diverging Bars") +
  coord_flip()
```

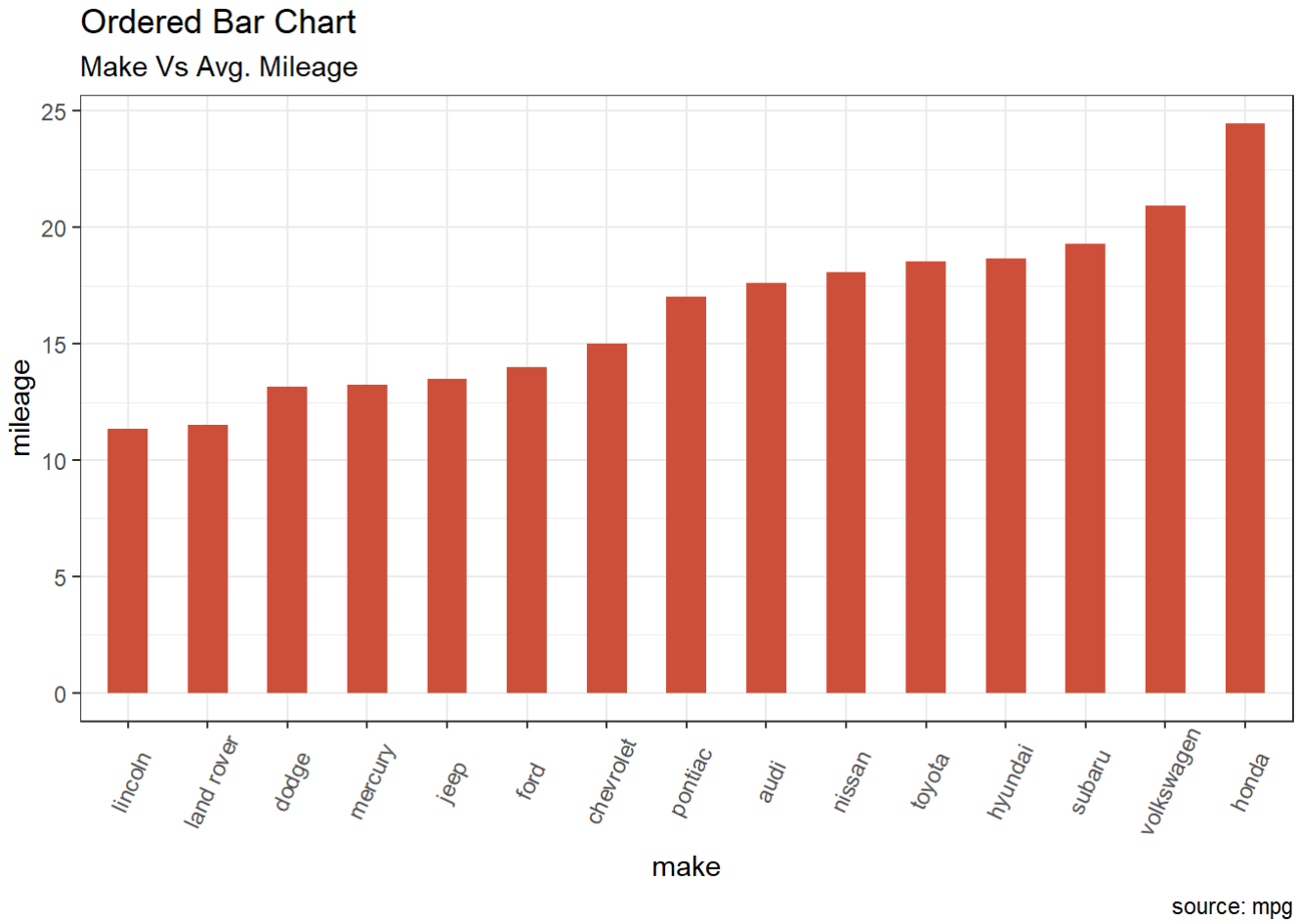


Ordered Bar Chart

```
cty_mpg <- aggregate(mpg$cty, by=list(mpg$manufacturer), FUN=mean)
# aggregate
colnames(cty_mpg) <- c("make", "mileage")  # change column names
cty_mpg <- cty_mpg[order(cty_mpg$mileage), ]  # sort
cty_mpg$make <- factor(cty_mpg$make, levels = cty_mpg$make)  # to
 retain the order in plot.
head(cty_mpg, 4)
```

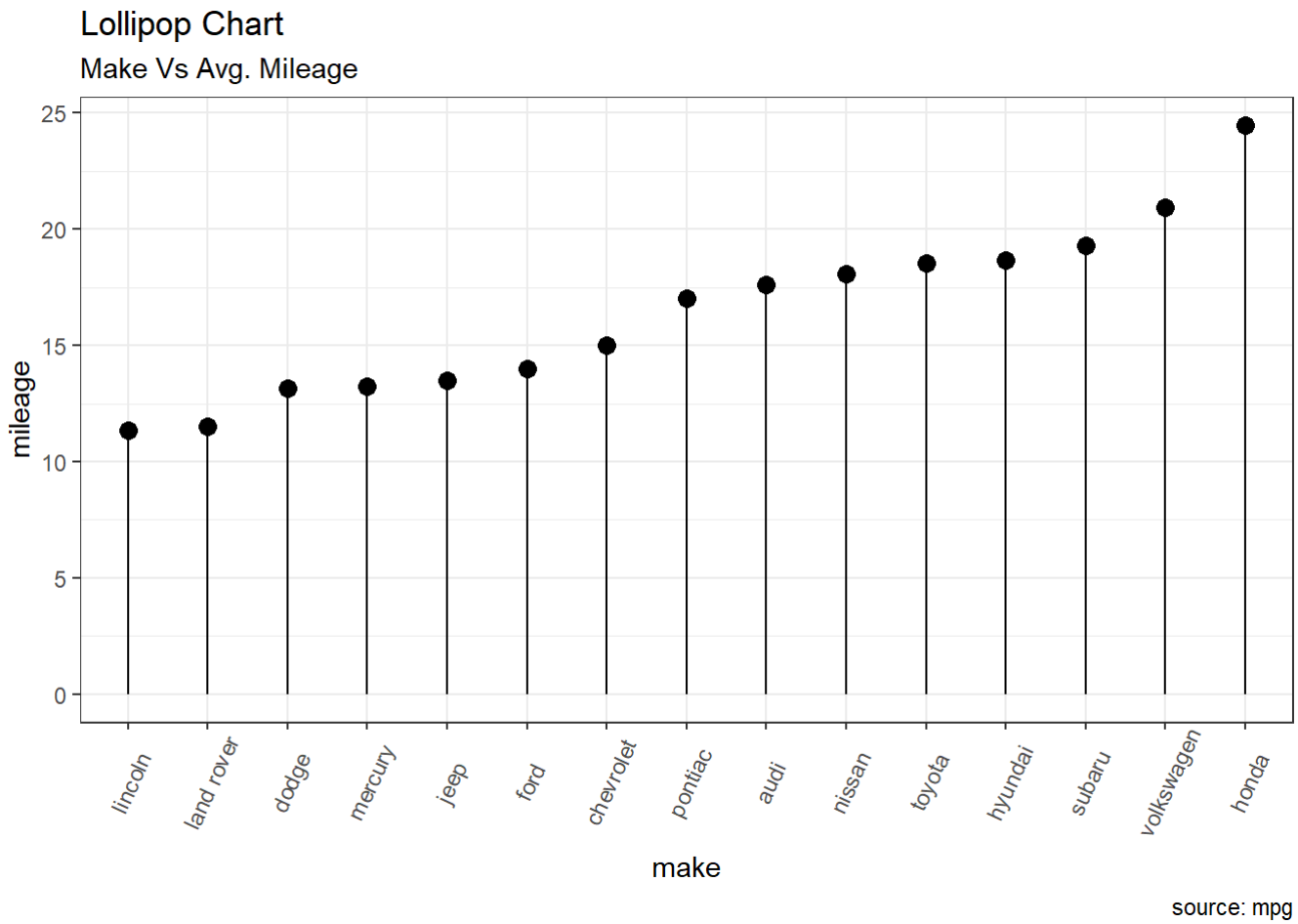
```
##          make  mileage
## 9     lincoln 11.33333
## 8  land rover 11.50000
## 3       dodge 13.13514
## 10    mercury 13.25000
```

```
library(ggplot2)
theme_set(theme_bw())

# Draw plot
ggplot(cty_mpg, aes(x=make, y=mileage)) +
  geom_bar(stat="identity", width=.5, fill="tomato3") +
  labs(title="Ordered Bar Chart",
       subtitle="Make Vs Avg. Mileage",
       caption="source: mpg") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



Lollipop Chart

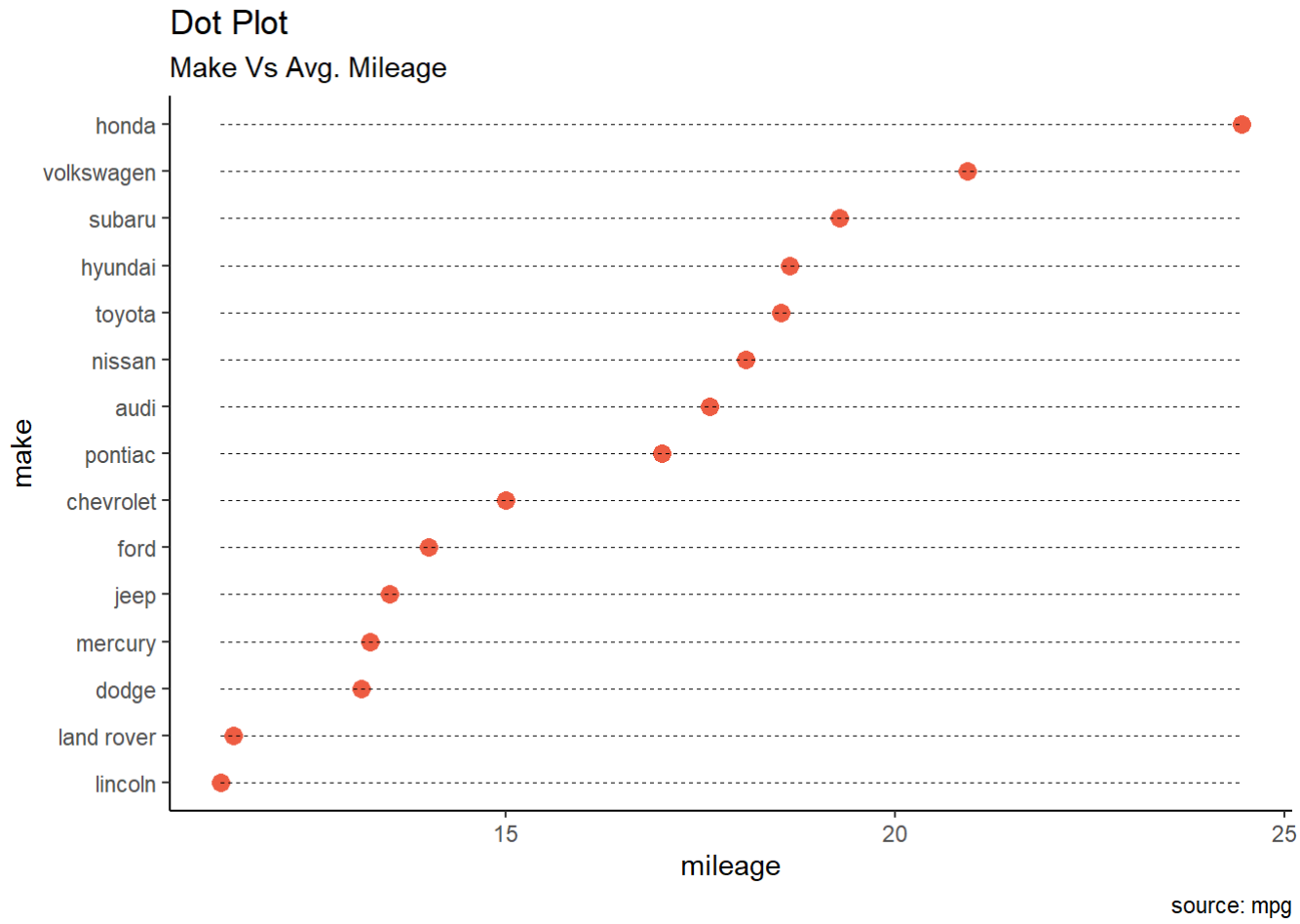
```
library(ggplot2)
theme_set(theme_bw())

# Plot
ggplot(cty_mpg, aes(x=make, y=mileage)) +
  geom_point(size=3) +
  geom_segment(aes(x=make,
                   xend=make,
                   y=0,
                   yend=mileage)) +
  labs(title="Lollipop Chart",
       subtitle="Make Vs Avg. Mileage",
       caption="source: mpg") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



Dot Plot

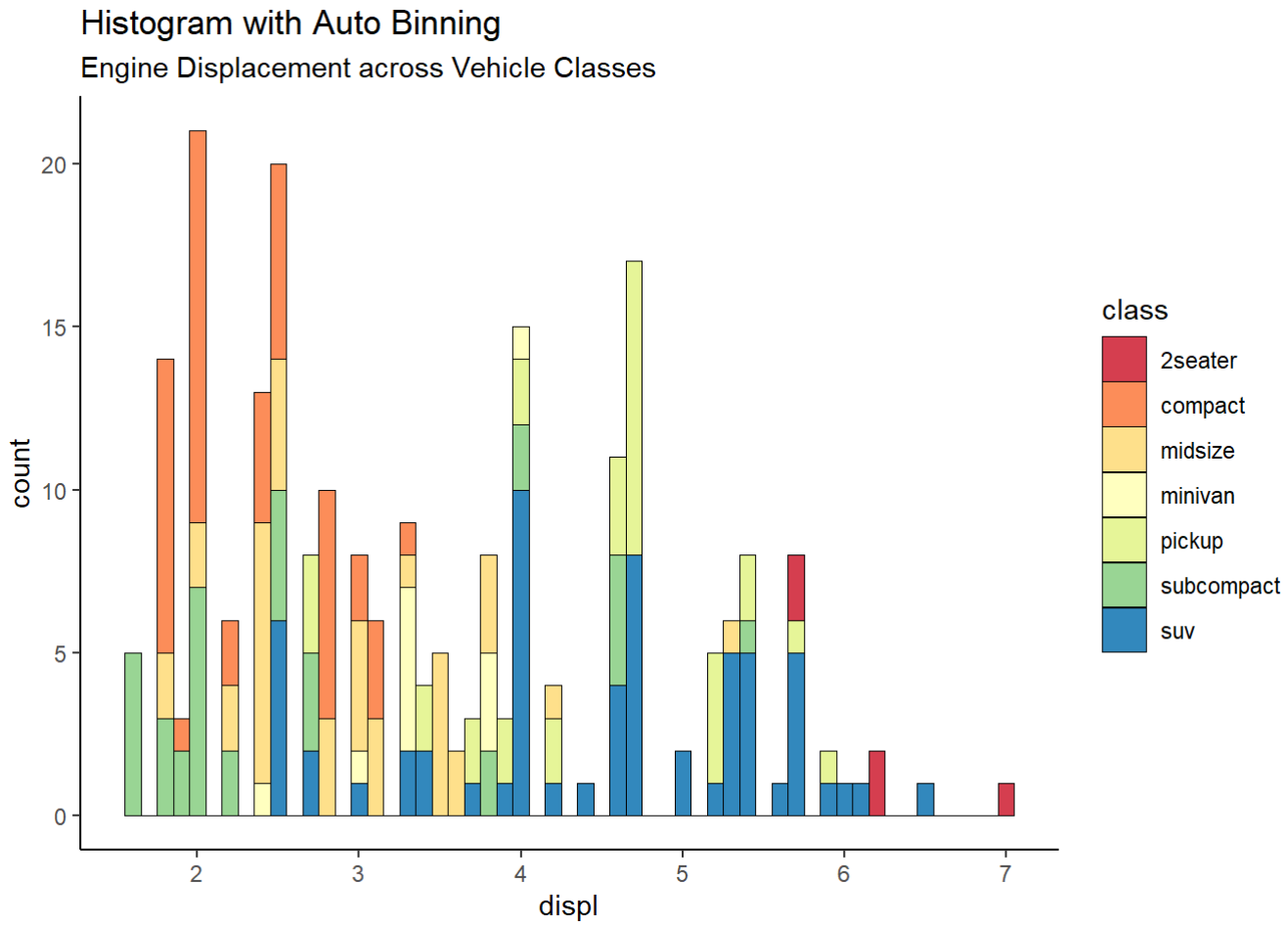
```
library(ggplot2)
library(scales)
theme_set(theme_classic())

# Plot
ggplot(cty_mpg, aes(x=make, y=mileage)) +
  geom_point(col="tomato2", size=3) +   # Draw points
  geom_segment(aes(x=make,
                   xend=make,
                   y=min(mileage),
                   yend=max(mileage)),
               linetype="dashed",
               size=0.1) +   # Draw dashed lines
  labs(title="Dot Plot",
       subtitle="Make Vs Avg. Mileage",
       caption="source: mpg") +
  coord_flip()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggp
lot2 3.4.0.
## i Please use `linewidth` instead.
```



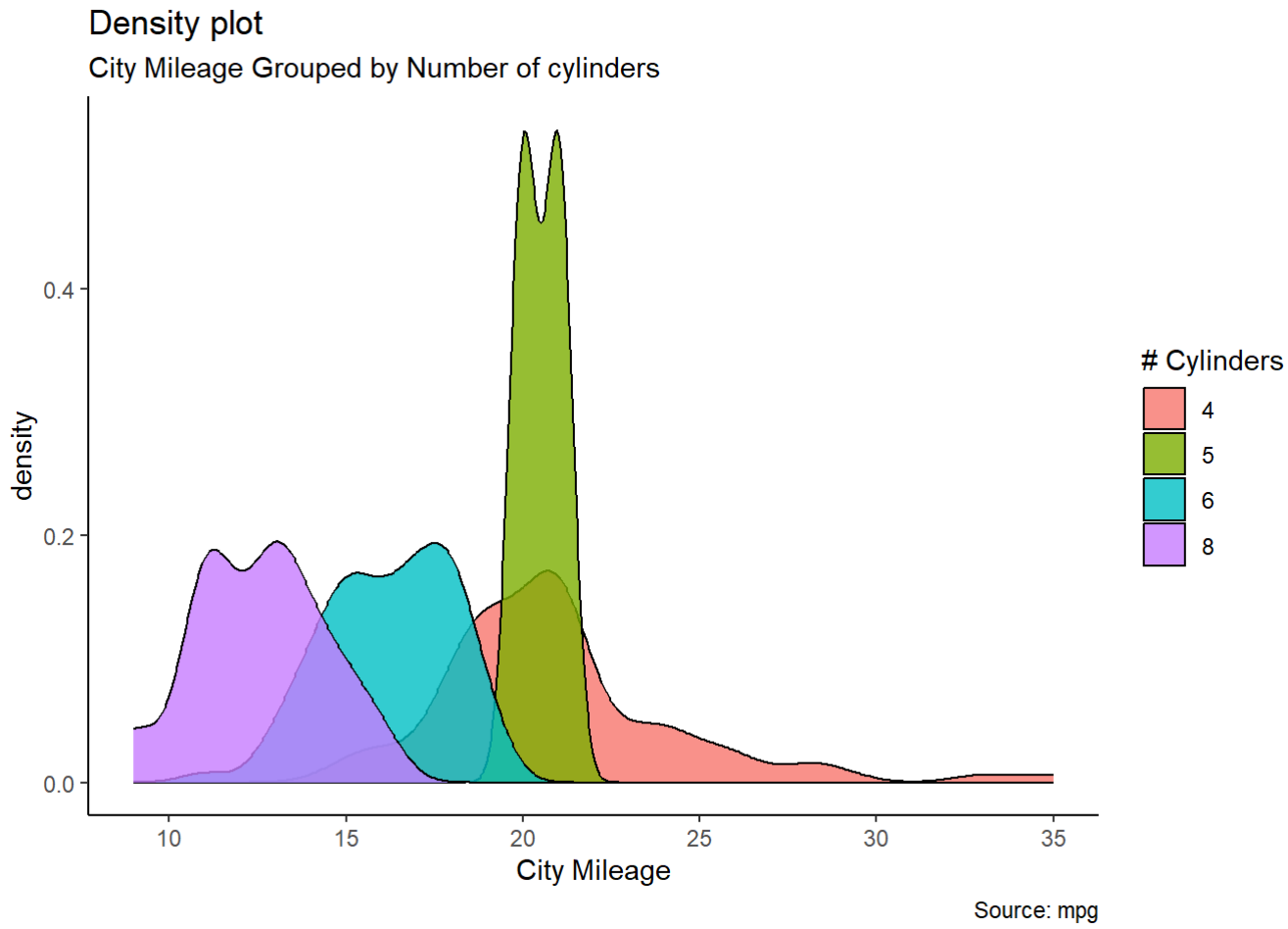
Histogram

```
library(ggplot2)
theme_set(theme_classic())

# Histogram on a Continuous (Numeric) Variable
g <- ggplot(mpg, aes(displ))+ scale_fill_brewer(palette = "Spectra
l")

g + geom_histogram(aes(fill=class),
                   binwidth = .1,
                   col="black",
                   size=.1) +  # change binwidth
  labs(title="Histogram with Auto Binning",
       subtitle="Engine Displacement across Vehicle Classes")
```



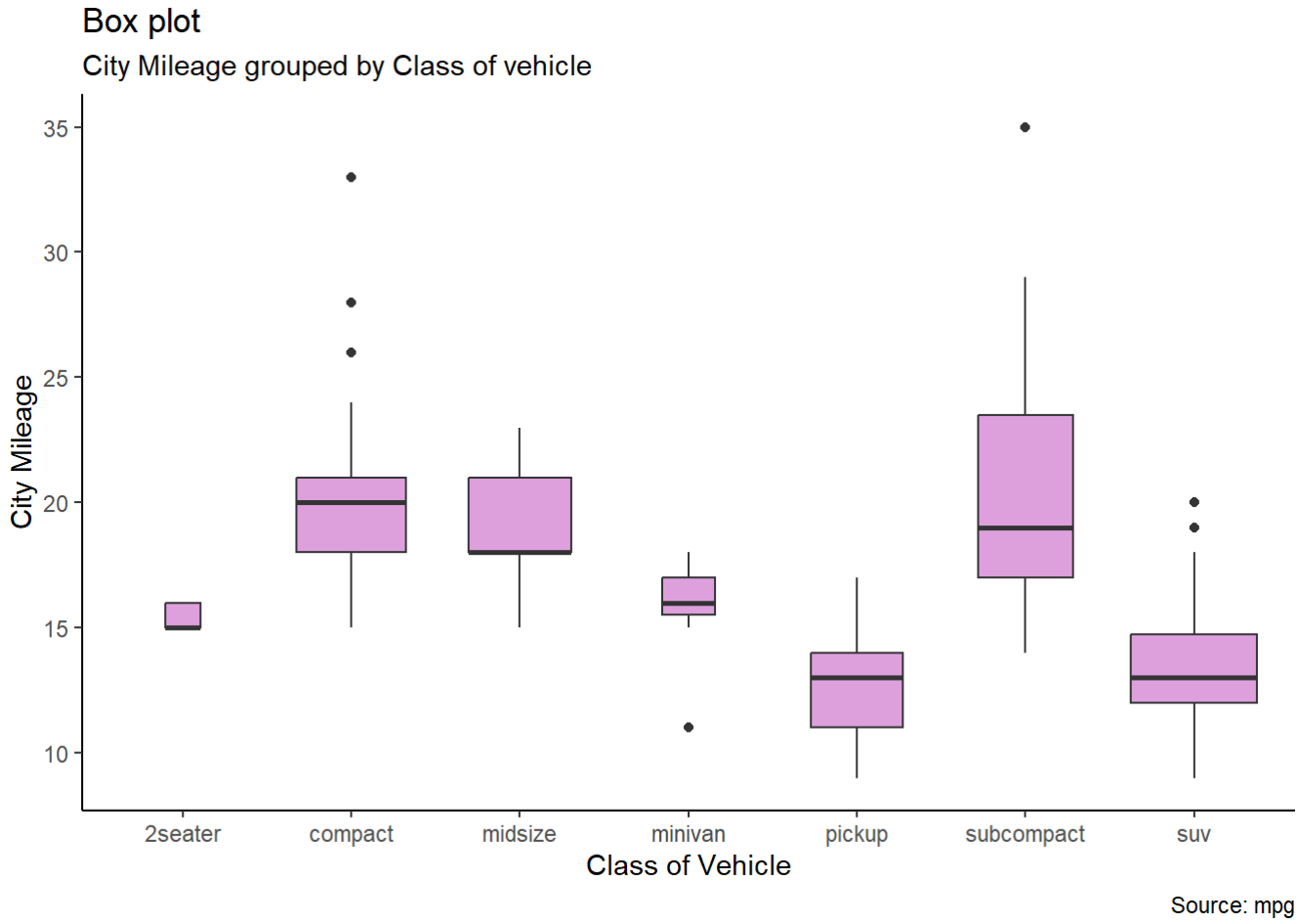
Density plot

```
library(ggplot2)
theme_set(theme_classic())

# Plot
g <- ggplot(mpg, aes(cty))
g + geom_density(aes(fill=factor(cyl)), alpha=0.8) +
    labs(title="Density plot",
         subtitle="City Mileage Grouped by Number of cylinders",
         caption="Source: mpg",
         x="City Mileage",
         fill="# Cylinders")
```



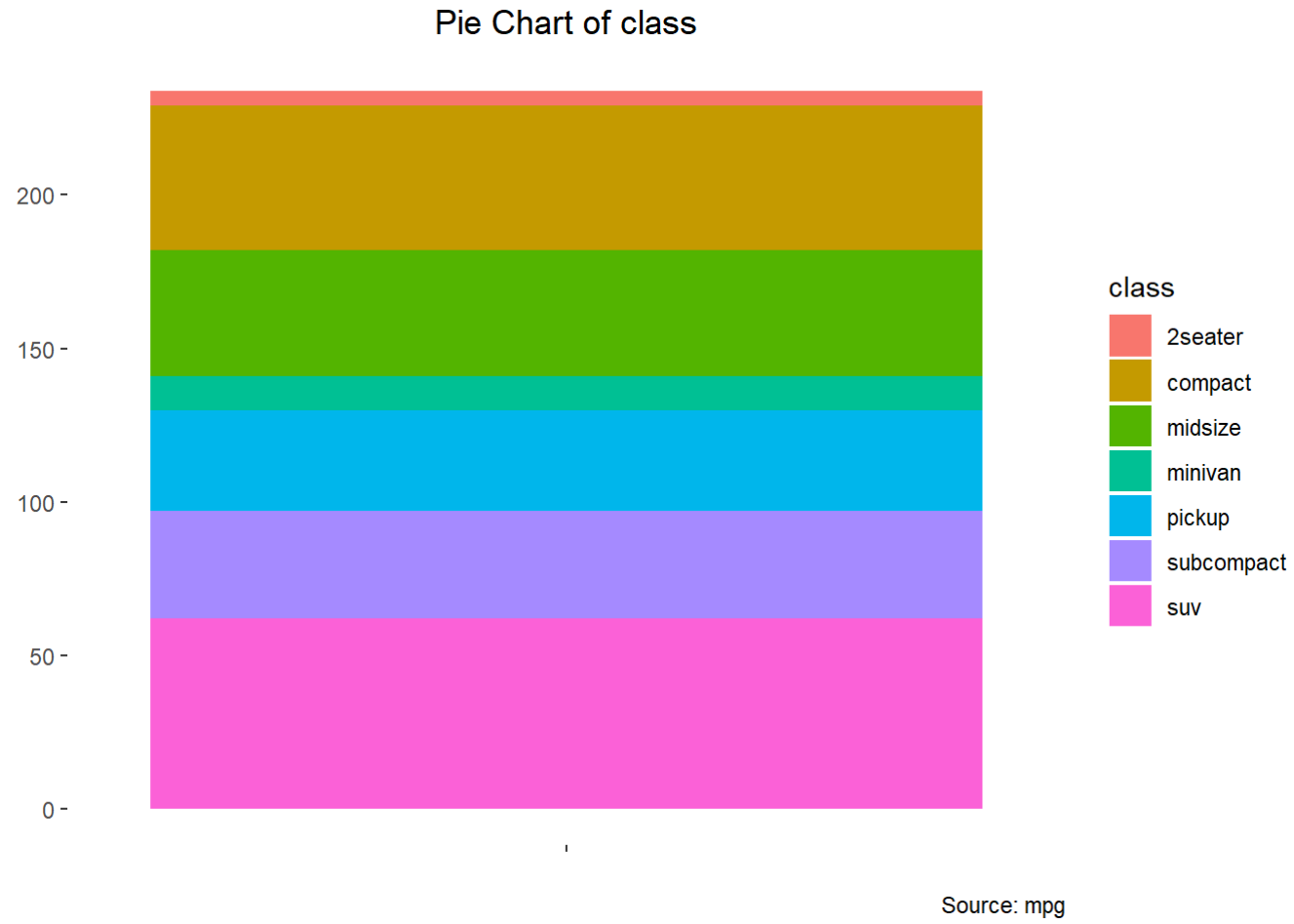
Box Plot

```
library(ggplot2)
theme_set(theme_classic())

# Plot
g <- ggplot(mpg, aes(class, cty))
g + geom_boxplot(varwidth=T, fill="plum") +
    labs(title="Box plot",
         subtitle="City Mileage grouped by Class of vehicle",
         caption="Source: mpg",
         x="Class of Vehicle",
         y="City Mileage")
```

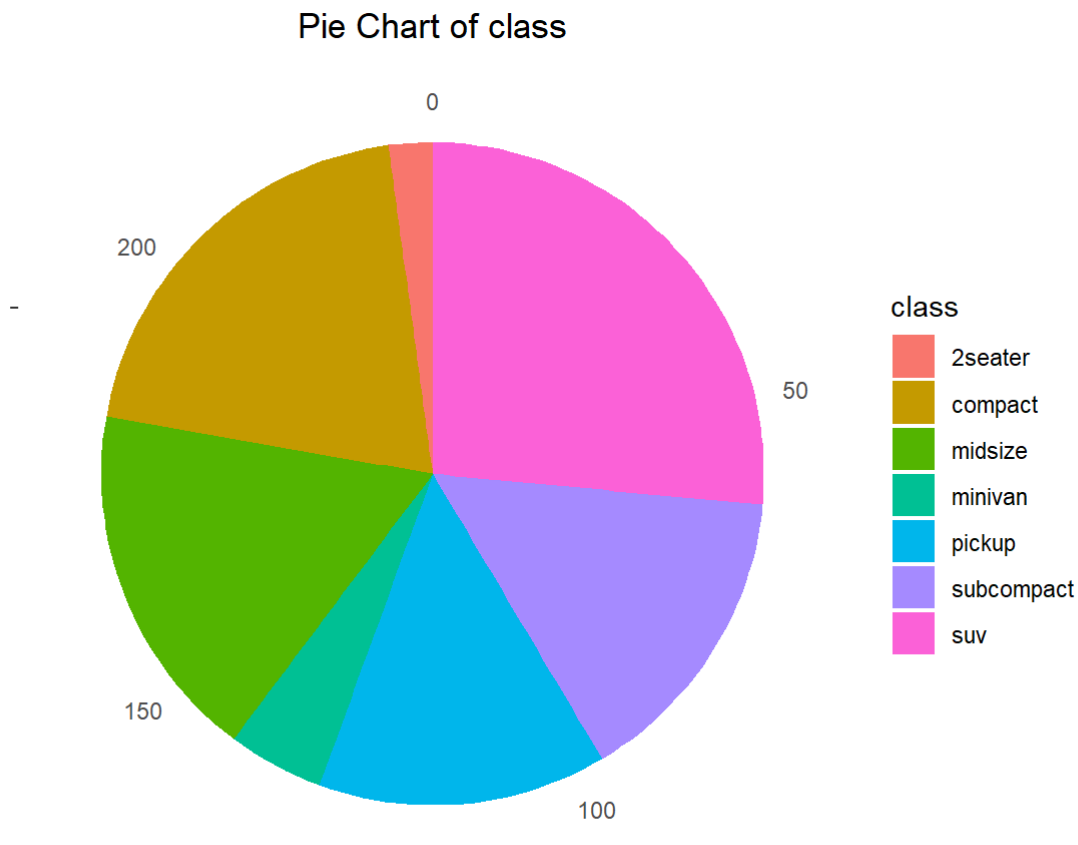


Pie Chart

```
library(ggplot2)
theme_set(theme_classic())

# Source: Frequency table
df <- as.data.frame(table(mpg$class))
colnames(df) <- c("class", "freq")
pie <- ggplot(df, aes(x = "", y=freq, fill = factor(class))) +
  geom_bar(width = 1, stat = "identity") +
  theme(axis.line = element_blank(),
        plot.title = element_text(hjust=0.5)) +
  labs(fill="class",
       x=NULL,
       y=NULL,
       title="Pie Chart of class",
       caption="Source: mpg")
pie
```




```
pie + coord_polar(theta = "y", start=0)
```



```
library(treemapify)
```

```
## Warning: package 'treemapify' was built under R version 4.0.5
```

```
# install.packages("ggplot2")
library(ggplot2)
group <- paste("Group", 1:9)
subgroup <- c("A", "C", "B", "A", "A",
              "C", "C", "B", "B")
value <- c(7, 25, 50, 5, 16,
           18, 30, 12, 41)

df <- data.frame(group, subgroup, value)
df
```

```
##     group subgroup value
## 1 Group 1        A     7
## 2 Group 2        C    25
## 3 Group 3        B    50
## 4 Group 4        A     5
## 5 Group 5        A    16
## 6 Group 6        C    18
## 7 Group 7        C    30
## 8 Group 8        B    12
## 9 Group 9        B    41
```

```
ggplot(df, aes(area = value, fill = value, label = group)) +
  geom_treemap() +
  geom_treemap_text(colour = c(rep("white", 2),
                               1, rep("white", 6)),
                    place = "centre", size = 15) +
  scale_fill_viridis_c()
```

Calendar Heatmap

```
library(ggplot2)
library(plyr)
library(scales)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

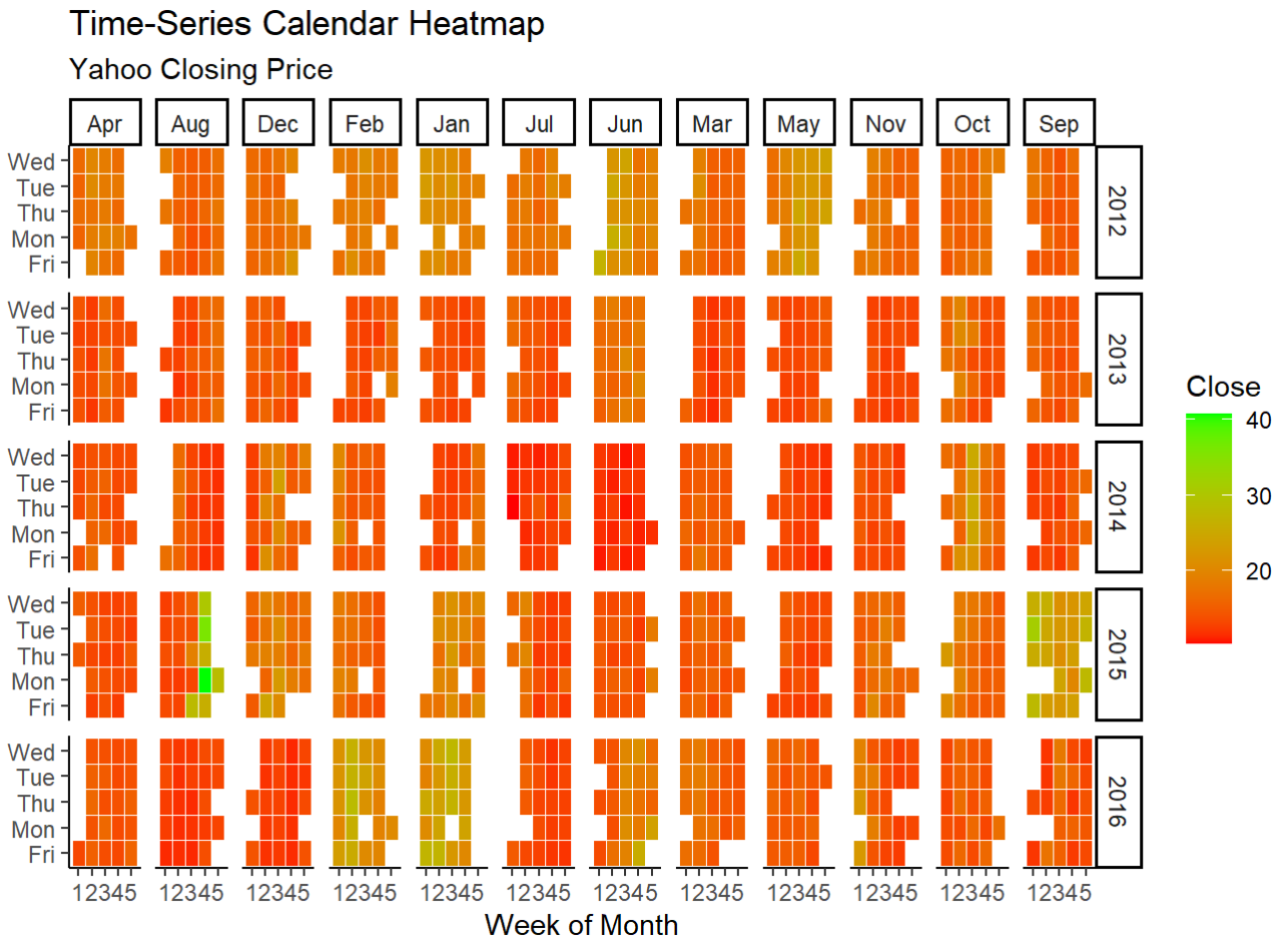
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
df <- read.csv("yahoo.csv")
df$date <- as.Date(df$date)  # format date
df <- df[df$year >= 2012, ]  # filter reqd years

# Create Month Week
df$yearmonth <- as.yearmon(df$date)
df$yearmonthf <- factor(df$yearmonth)
df <- ddply(df,.(yearmonthf), transform, monthweek=1+week-min(wee
k))  # compute week number of month
df <- df[, c("year", "yearmonthf", "monthf", "week", "monthweek",
"weekdayf", "VIX.Close")]
head(df)
```

```
##   year   yearmonthf monthf week monthweek weekdayf VIX.Close
## 1 2012 1\u6708 2012    Jan    1         1      Tue     22.97
## 2 2012 1\u6708 2012    Jan    1         1      Wed     22.22
## 3 2012 1\u6708 2012    Jan    1         1      Thu     21.48
## 4 2012 1\u6708 2012    Jan    1         1      Fri     20.63
## 5 2012 1\u6708 2012    Jan    2         2      Mon     21.07
## 6 2012 1\u6708 2012    Jan    2         2      Tue     20.69
```

```
#>   year yearmonthf monthf week monthweek weekdayf VIX.Close
#> 1 2012   Jan 2012    Jan    1         1      Tue     22.97
#> 2 2012   Jan 2012    Jan    1         1      Wed     22.22
#> 3 2012   Jan 2012    Jan    1         1      Thu     21.48
#> 4 2012   Jan 2012    Jan    1         1      Fri     20.63
#> 5 2012   Jan 2012    Jan    2         2      Mon     21.07
#> 6 2012   Jan 2012    Jan    2         2      Tue     20.69
```

```
# Plot
ggplot(df, aes(monthweek, weekdayf, fill = VIX.Close)) +
  geom_tile(colour = "white") +
  facet_grid(year~monthf) +
  scale_fill_gradient(low="red", high="green") +
  labs(x="Week of Month",
       y="",
       title = "Time-Series Calendar Heatmap",
       subtitle="Yahoo Closing Price",
       fill="Close")
```



Clusters

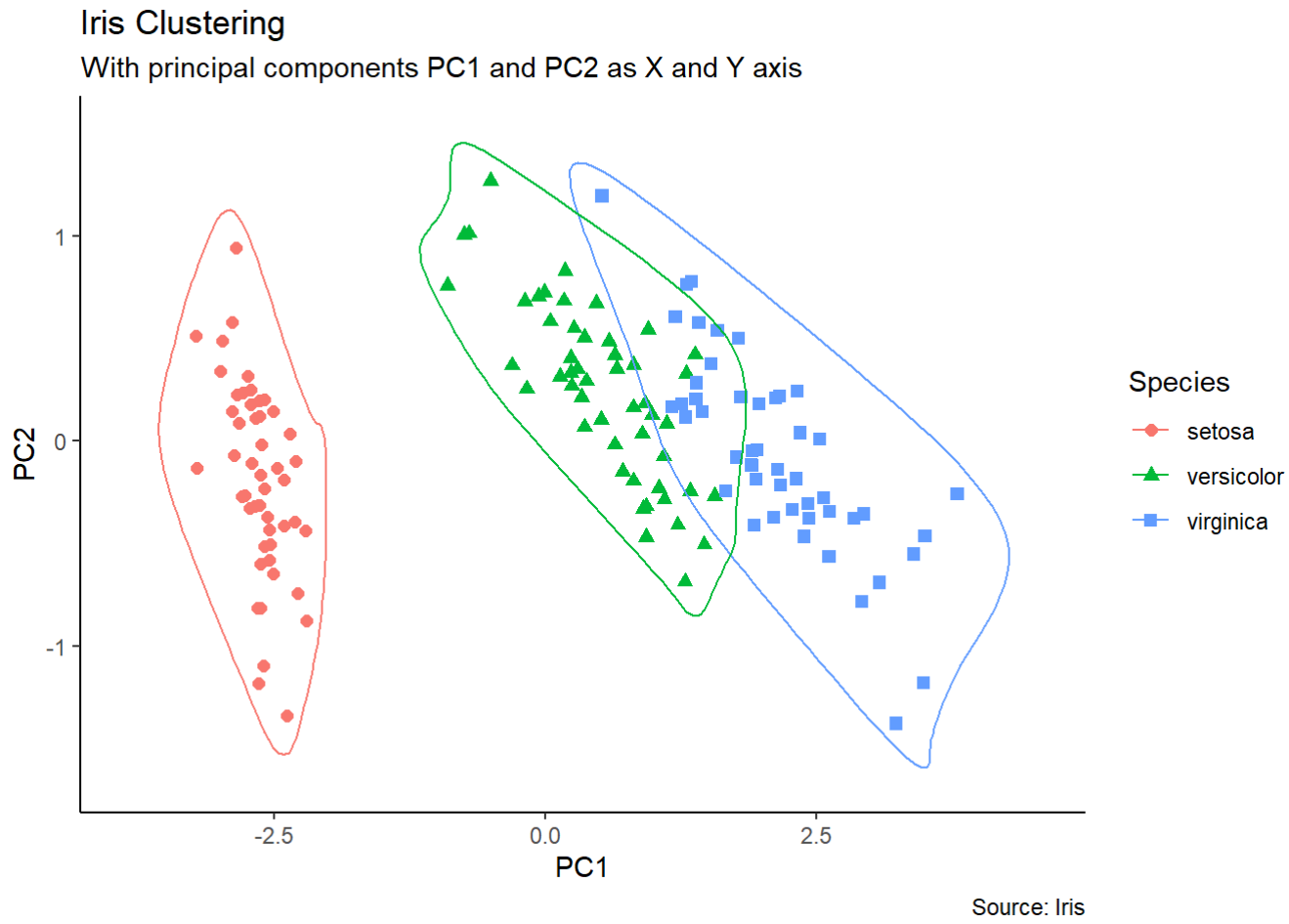
```
 #devtools::install_github("hrbrmstr/ggalt")
library(ggplot2)
library(ggalt)
```

```
## Registered S3 methods overwritten by 'ggalt':
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2
```

```
library(ggfortify)
```

```
## Registered S3 method overwritten by 'ggfortify':
##   method        from
##   fortify.table ggalt
```

```
theme_set(theme_classic())

# Compute data with principal components ------------------
df <- iris[c(1, 2, 3, 4)]
pca_mod <- prcomp(df)  # compute principal components

# Data frame of principal components ----------------------
df_pc <- data.frame(pca_mod$x, Species=iris$Species)  # dataframe
 of principal components
df_pc_vir <- df_pc[df_pc$Species == "virginica", ]  # df for 'virg
inica'
df_pc_set <- df_pc[df_pc$Species == "setosa", ]  # df for 'setosa'
df_pc_ver <- df_pc[df_pc$Species == "versicolor", ]  # df for 'ver
sicolor'

# Plot ----------------------------------------------------
ggplot(df_pc, aes(PC1, PC2, col=Species)) +
  geom_point(aes(shape=Species), size=2) +   # draw points
  labs(title="Iris Clustering",
       subtitle="With principal components PC1 and PC2 as X and Y
 axis",
       caption="Source: Iris") +
  coord_cartesian(xlim = 1.2 * c(min(df_pc$PC1), max(df_pc$PC1)),
                  ylim = 1.2 * c(min(df_pc$PC2), max(df_pc$PC2)))
 +   # change axis limits
  geom_encircle(data = df_pc_vir, aes(x=PC1, y=PC2)) +   # draw ci
rcles
  geom_encircle(data = df_pc_set, aes(x=PC1, y=PC2)) +
  geom_encircle(data = df_pc_ver, aes(x=PC1, y=PC2))
```



References

- HKU Stat3622 Data Visualization (https://ajzhanghk.github.io/Stat3622/)
- ggplot2. Elegant Graphics for Data Analysis (https://ggplot2-book.org/collective-geoms.html)