SIRE516

Data visualization

Basic R plots

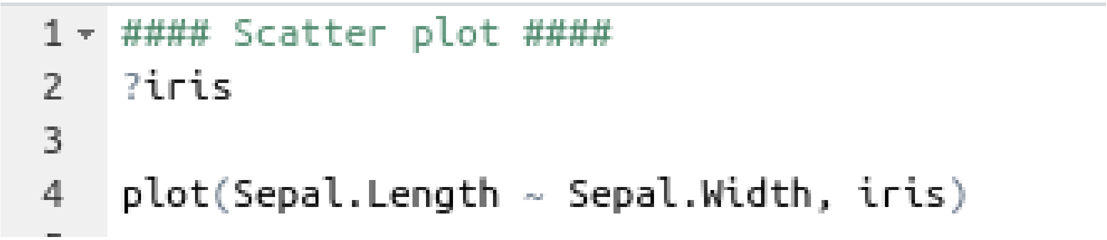
Basic R plots

- Pro: Fast and simple
- Con: More difficult customization

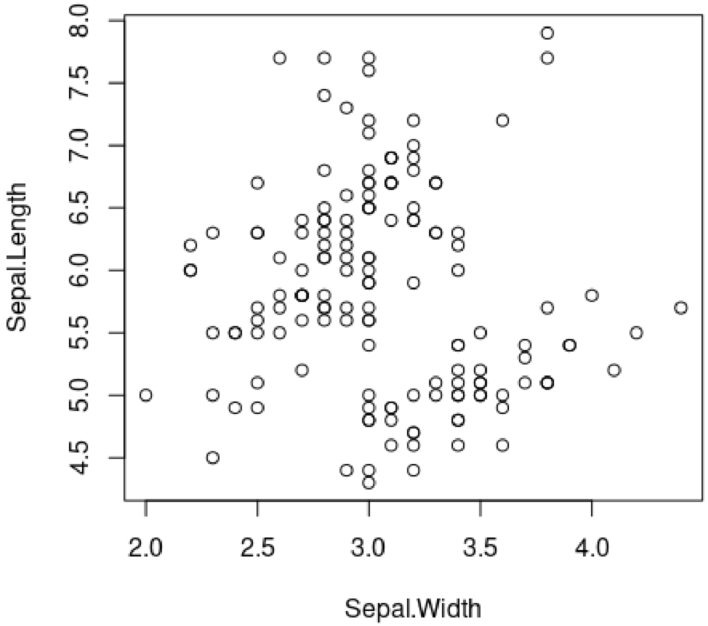
- In this class, we will use these plots for quick visualization.
- I will not cover customization of these plots.
- Plot customizations will be done in ‘ggplot.’

Scatter plot

- Showing two dimensional data
- Two numerical values

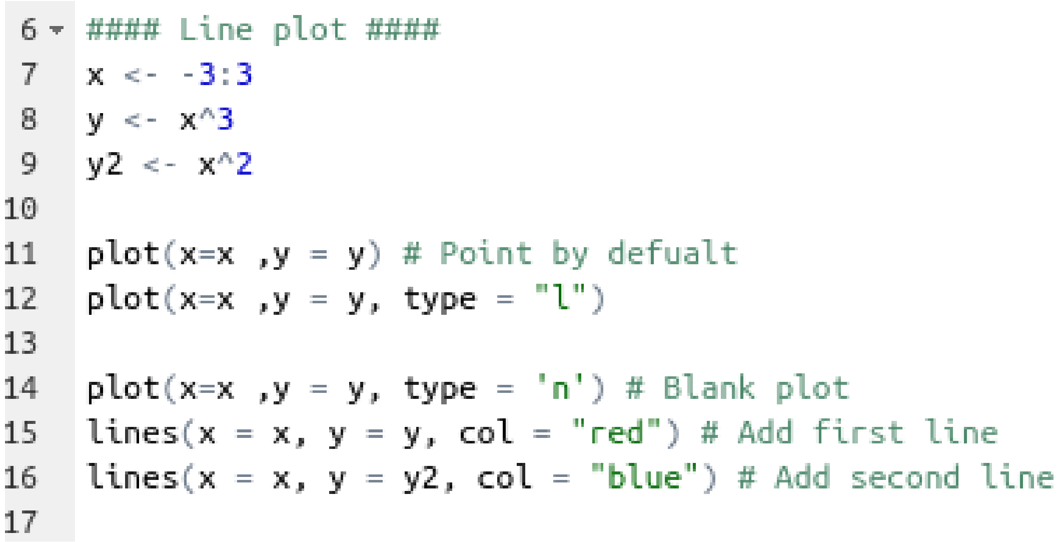


```
#### Scatter plot ####
?iris

plot(Sepal.Length ~ Sepal.Width, iris)
```



Line plot

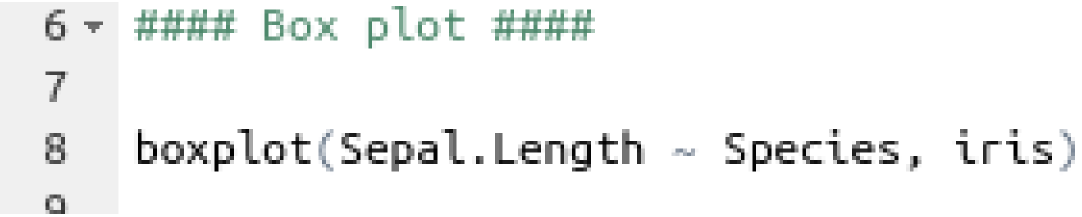
- Showing two dimensional data
- Two numerical values
- The line can show:
  - Repeated measurements over time (longitudinal data with X variable = time)
  - Trend of correlation between variables



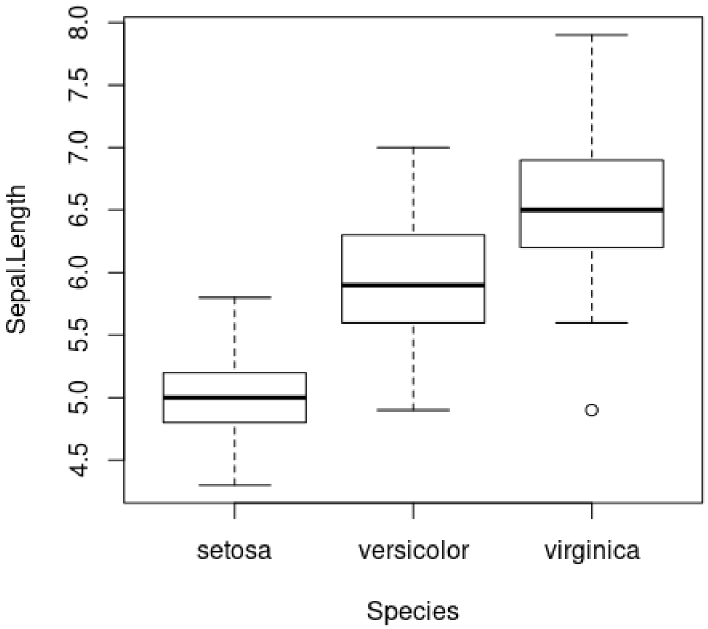
```
#### Line plot ####
x <- -3:3
y <- x^3
y2 <- x^2

plot(x=x ,y = y) # Point by defualt
plot(x=x ,y = y, type = "l")

plot(x=x ,y = y, type = 'n') # Blank plot
lines(x = x, y = y, col = "red") # Add first line
lines(x = x, y = y2, col = "blue") # Add second line

```

Box plot

- Showing two dimensional data
- One numerical value and one categorical value

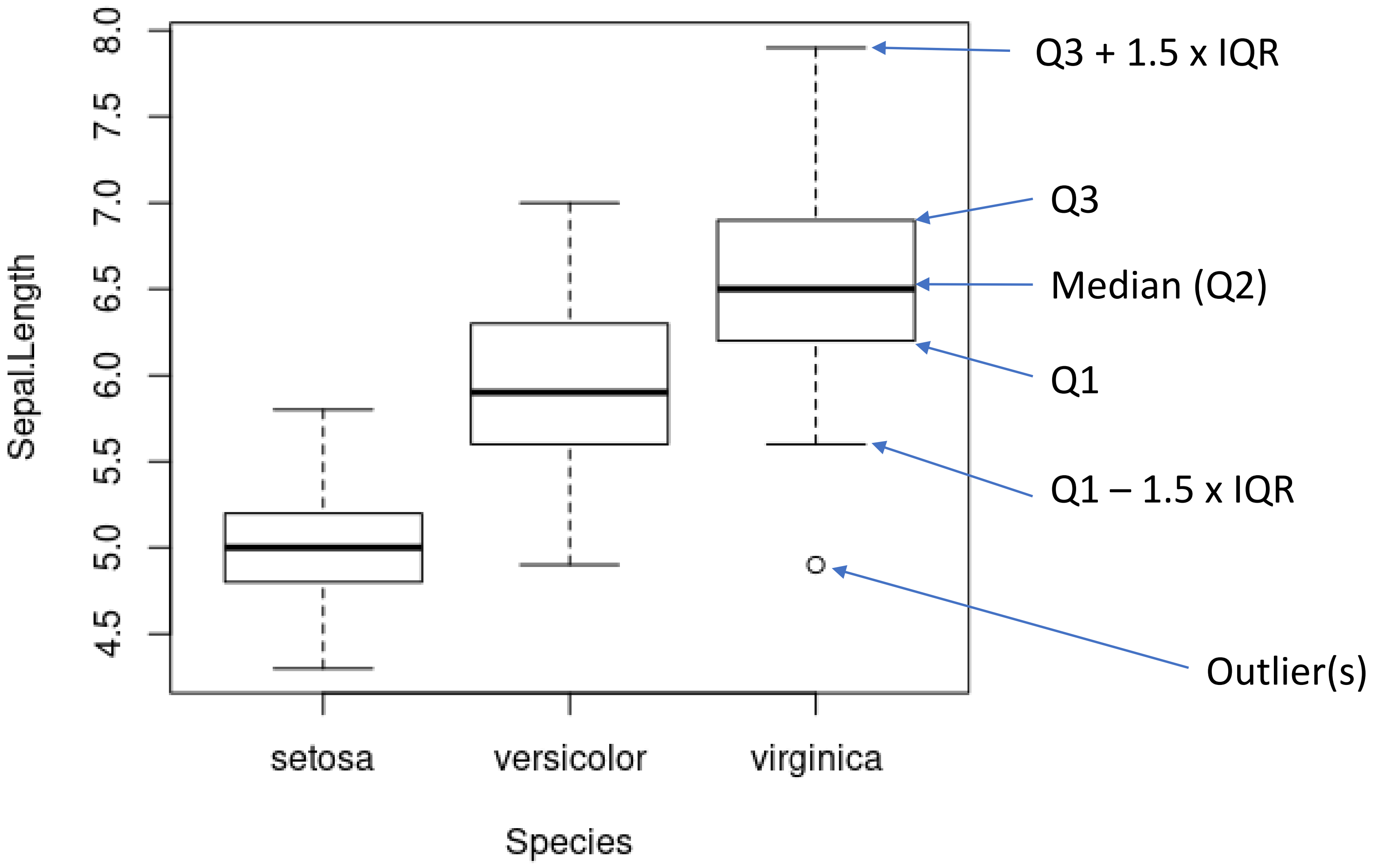


```
#### Box plot ####

boxplot(Sepal.Length ~ Species, iris)
```



Box plot

- Interpretation
- IQR = Q3-Q1



Q3 + 1.5 x IQR

Q3

Median (Q2)

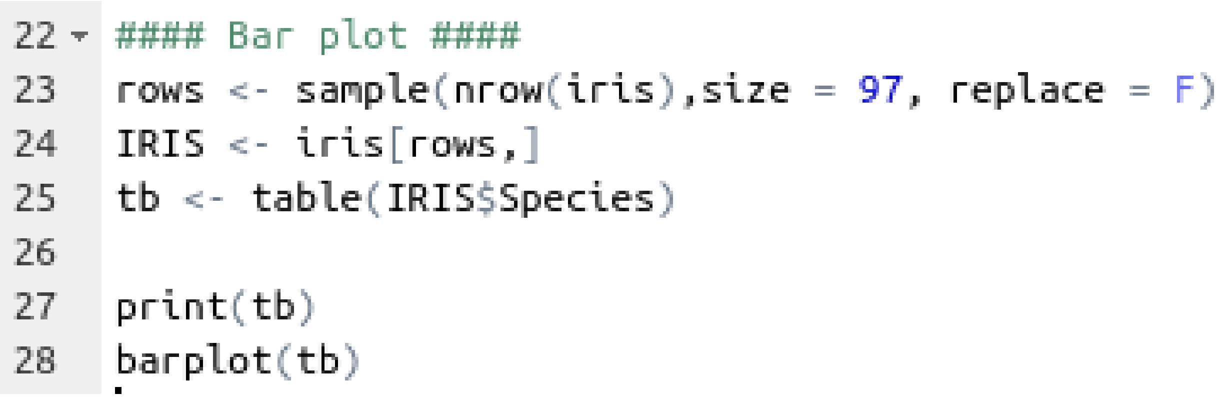
Q1

Q1 – 1.5 x IQR

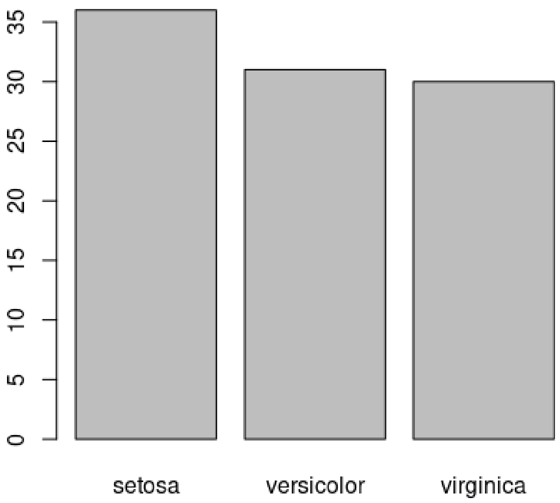
Outlier(s)

Bar plot

- One-dimension data → Counts of categorical variables
- Two-dimension data → One categorical and one central value (e.g. mean) of numerical variables
  - Mean + Standard error of the mean: If data are not normally distributed, the shape of distribution is lost.
  - Boxplot for showing how data are distributed

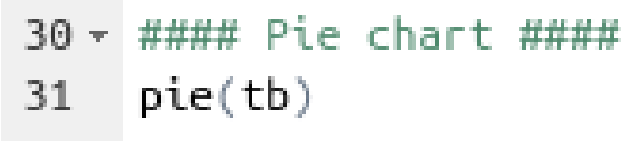


```r
#### Bar plot ####
rows <- sample(nrow(iris),size = 97, replace = F)
IRIS <- iris[rows,]
tb <- table(IRIS$Species)

print(tb)
barplot(tb)
```

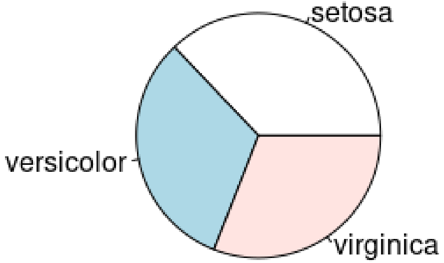


Pie chart

- One-dimension data → Counts of categorical variables
- Caution: Pie chart poorly presents data (angle vs height in bar plot). Avoid using it

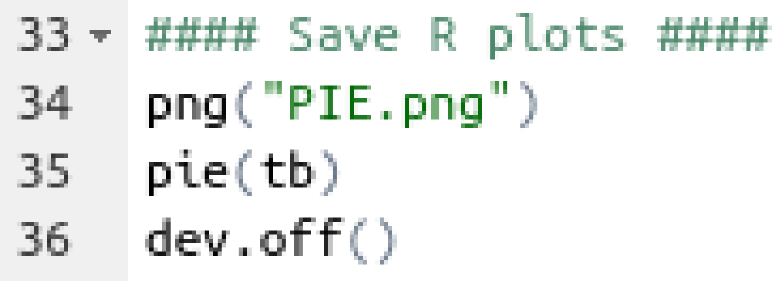


```
#### Pie chart ####
pie(tb)
```



How to save R plots

- Specify where (what file format) you want to direct the plot from the screen/terminal to
- Plot the plot
- Turn off the direction

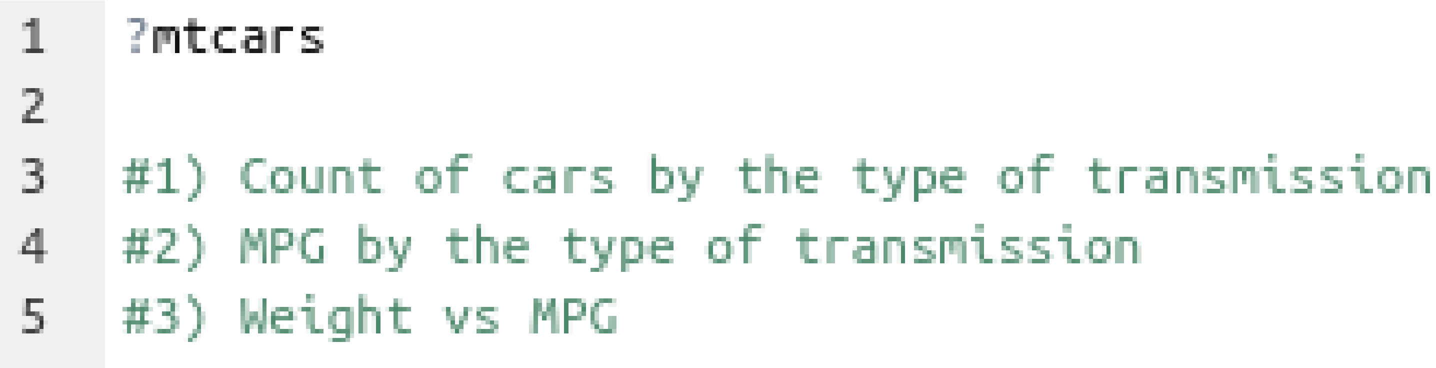


```r
#### Save R plots ####
png("PIE.png")
pie(tb)
dev.off()
```

Practical 1

Practical 1

- We will use basic R plots to explore ‘mtcars’ dataset
- Try to select appropriate plot types



```
?mtcars

#1) Count of cars by the type of transmission
#2) MPG by the type of transmission
#3) Weight vs MPG
```

ggplot2

Why ggplot2?

- Coding is more intuitive?
- Customizations!
- Plots with publication quality
  - Resolution
  - Format (fonts and special characters e.g. the Lancet’s decimal: 23·4, not 23.4)
- Packages for advanced plots built from ggplot2
  - Survival analysis
  - Spatial statistics
- Require some data preparations

Basic structure of ggplot2

```
graph <- ggplot(data = data1, aes(x = x1, y = y1, col = z1)) +
                geom_point() +
                geom_line(data = data2, aes_string(x = “x2”, y = “y2”, col = “z2))+
                scale_color_discrete(…)+
                scale_x_continuous(…)+
                theme(panel.grid = …)
```

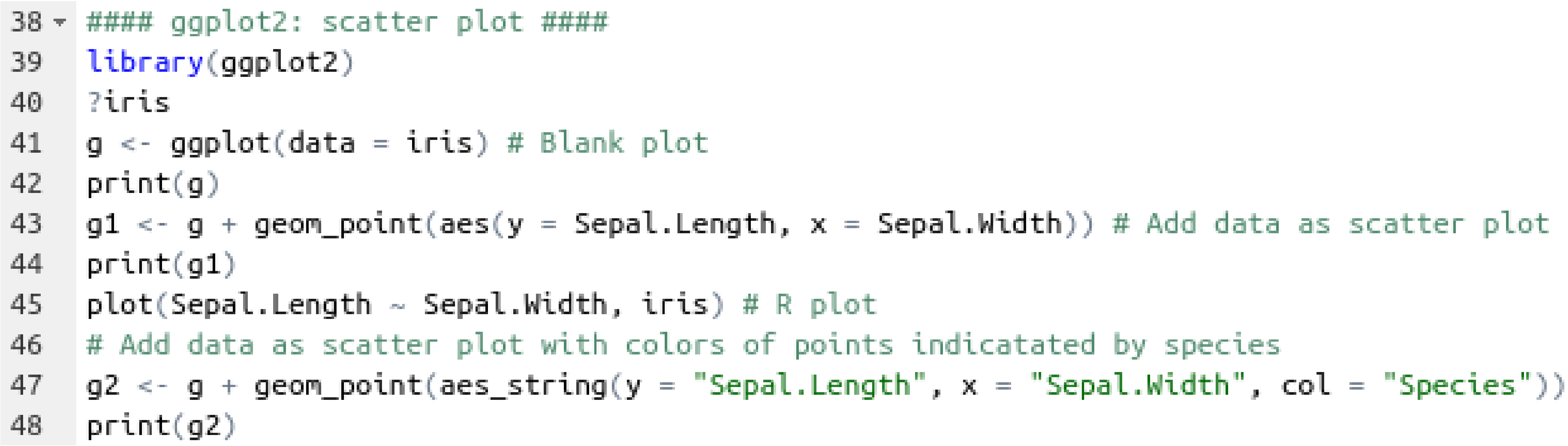
- Plot elements are added step-by-step
- Aesthetic mappings: aes (no “_”) and aes_string (with “_”) telling what plot element connect to what variable
- Assign plot elements outside of aes() → Constant value (x = 1, col =“red”)

Basic structure of ggplot2

```
graph <- ggplot(data = data1, aes(x = x1, y = y1, col = z1)) +
              geom_point() +
              geom_line(data = data2, aes_string(x = “x2”, y = “y2”, col = “z2))+
              scale_color_discrete(…)+
              scale_x_continuous(…)+
              theme(panel.grid = …)
```

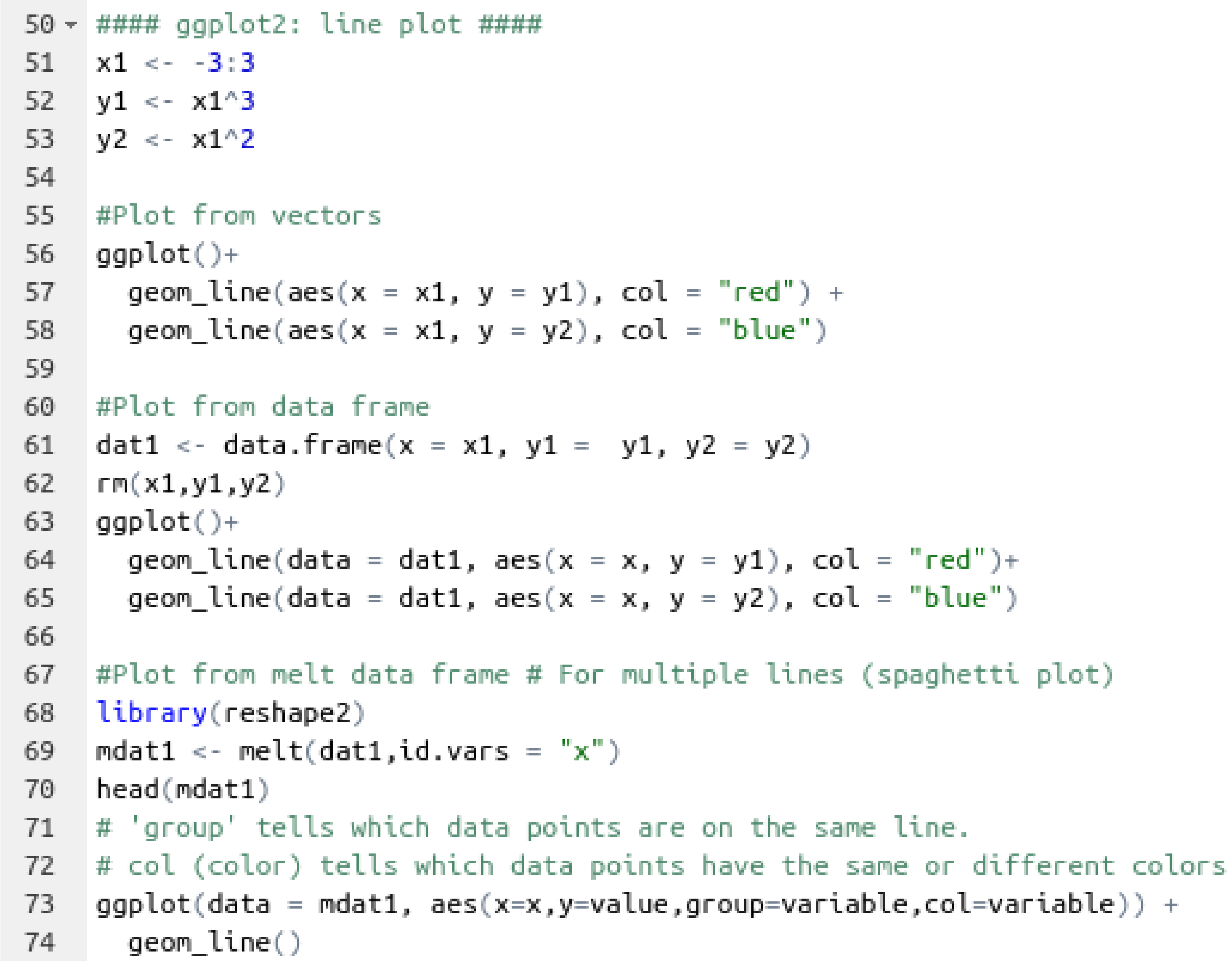
- geom_[plot type] → add data points to the plot as indicated by “plot_type”
- scale_[something]_[data_type] → scale plot elements (usually) mapped by aes in a certain manner (e.g. log10-scale, customized color)
- theme(…) → customize ‘static’ plot elements (e.g. fonts, grid)

Scatter plot



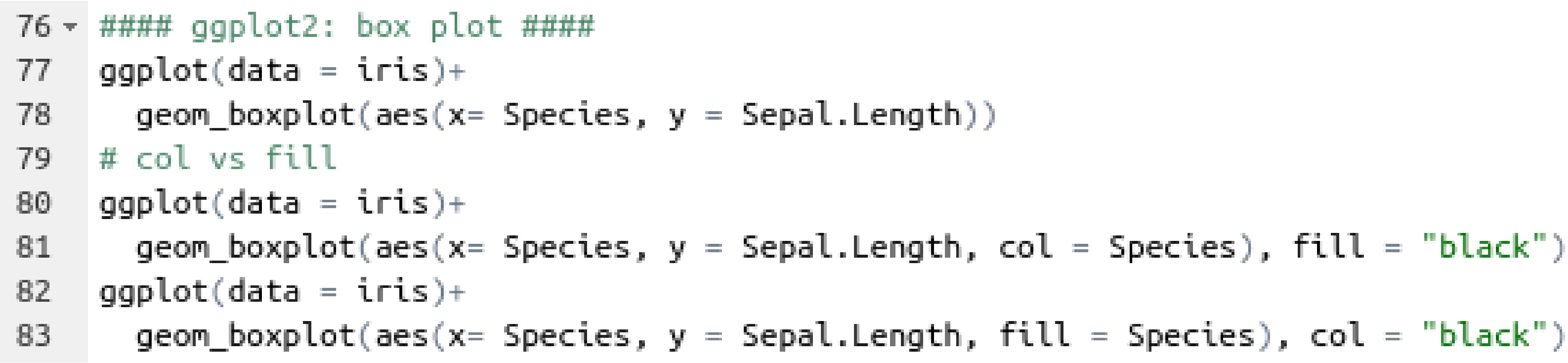
```r
#### ggplot2: scatter plot ####
library(ggplot2)
?iris
g <- ggplot(data = iris) # Blank plot
print(g)
g1 <- g + geom_point(aes(y = Sepal.Length, x = Sepal.Width)) # Add data as scatter plot
print(g1)
plot(Sepal.Length ~ Sepal.Width, iris) # R plot
# Add data as scatter plot with colors of points indicatated by species
g2 <- g + geom_point(aes_string(y = "Sepal.Length", x = "Sepal.Width", col = "Species"))
print(g2)
```

Line plot



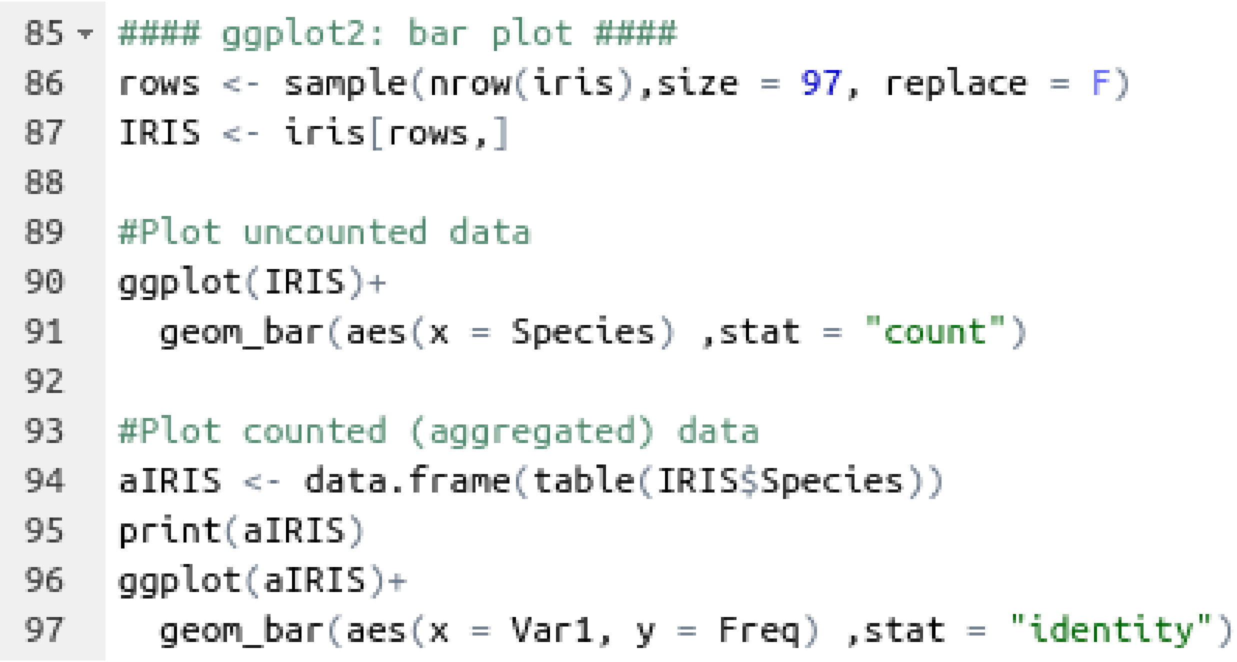
```r
#### ggplot2: line plot ####
x1 <- -3:3
y1 <- x1^3
y2 <- x1^2

#Plot from vectors
ggplot()+
  geom_line(aes(x = x1, y = y1), col = "red") +
  geom_line(aes(x = x1, y = y2), col = "blue")

#Plot from data frame
dat1 <- data.frame(x = x1, y1 =  y1, y2 = y2)
rm(x1,y1,y2)
ggplot()+
  geom_line(data = dat1, aes(x = x, y = y1), col = "red")+
  geom_line(data = dat1, aes(x = x, y = y2), col = "blue")

#Plot from melt data frame # For multiple lines (spaghetti plot)
library(reshape2)
mdat1 <- melt(dat1,id.vars = "x")
head(mdat1)
# 'group' tells which data points are on the same line.
# col (color) tells which data points have the same or different colors
ggplot(data = mdat1, aes(x=x,y=value,group=variable,col=variable)) +
  geom_line()
```

Box plot



```r
#### ggplot2: box plot ####
ggplot(data = iris)+
  geom_boxplot(aes(x= Species, y = Sepal.Length))
# col vs fill
ggplot(data = iris)+
  geom_boxplot(aes(x= Species, y = Sepal.Length, col = Species), fill = "black")
ggplot(data = iris)+
  geom_boxplot(aes(x= Species, y = Sepal.Length, fill = Species), col = "black")
```

Bar plot

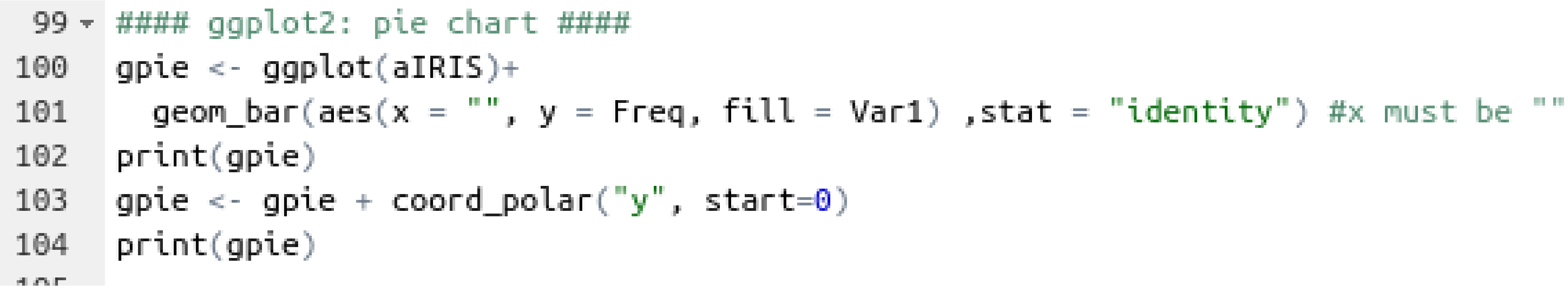


```r
#### ggplot2: bar plot ####
rows <- sample(nrow(iris),size = 97, replace = F)
IRIS <- iris[rows,]

#Plot uncounted data
ggplot(IRIS)+
  geom_bar(aes(x = Species) ,stat = "count")

#Plot counted (aggregated) data
aIRIS <- data.frame(table(IRIS$Species))
print(aIRIS)
ggplot(aIRIS)+
  geom_bar(aes(x = Var1, y = Freq) ,stat = "identity")
```

- stat = “identity” can be used to plot “center” values of aggregated data (e.g. mean of group)

Pie chart

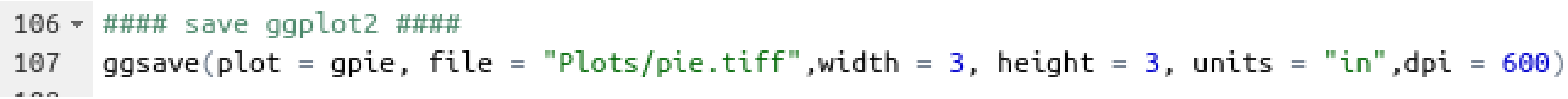
- No pie chart function in ggplot2
- Geom_bar(…)+ coord_polar(…)



```
#### ggplot2: pie chart ####
gpie <- ggplot(aIRIS)+
  geom_bar(aes(x = "", y = Freq, fill = Var1) ,stat = "identity") #x must be ""
print(gpie)
gpie <- gpie + coord_polar("y", start=0)
print(gpie)
```

How to save ggplot2

- Simply use “ggsave()” function
- Name the type of the picture you want to save (e.g. tiff, jpeg, png)
- Specify dimension and units (e.g. width = 3, height = 4, unit = “in”)
- Specify resolution (For publication dpi > 300)

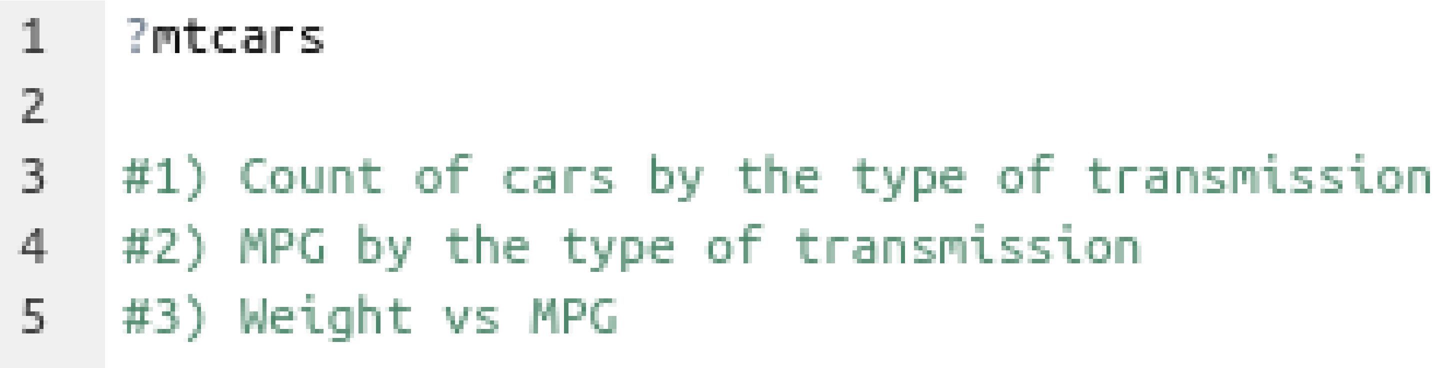


```
#### save ggplot2 ####
ggsave(plot = gpie, file = "Plots/pie.tiff",width = 3, height = 3, units = "in",dpi = 600)
```

Practical 2

Practical 2

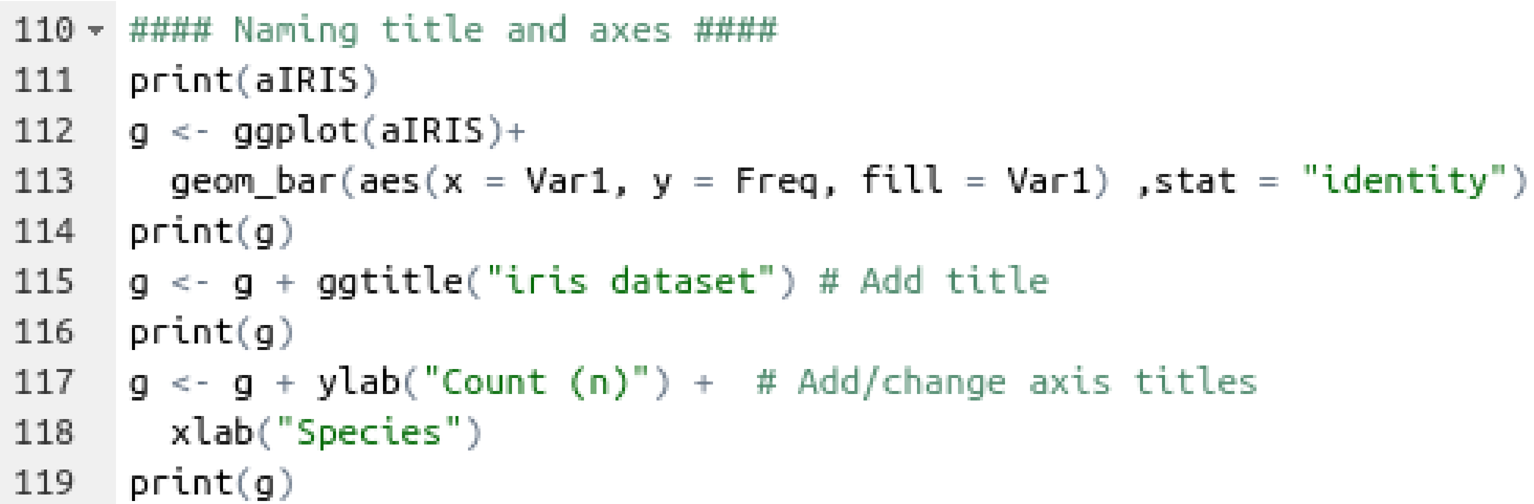
- As in practical 1, we will use ggplot2 to explore ‘mtcars’ dataset
- Try to select appropriate plots plot types



```
?mtcars

#1) Count of cars by the type of transmission
#2) MPG by the type of transmission
#3) Weight vs MPG
```

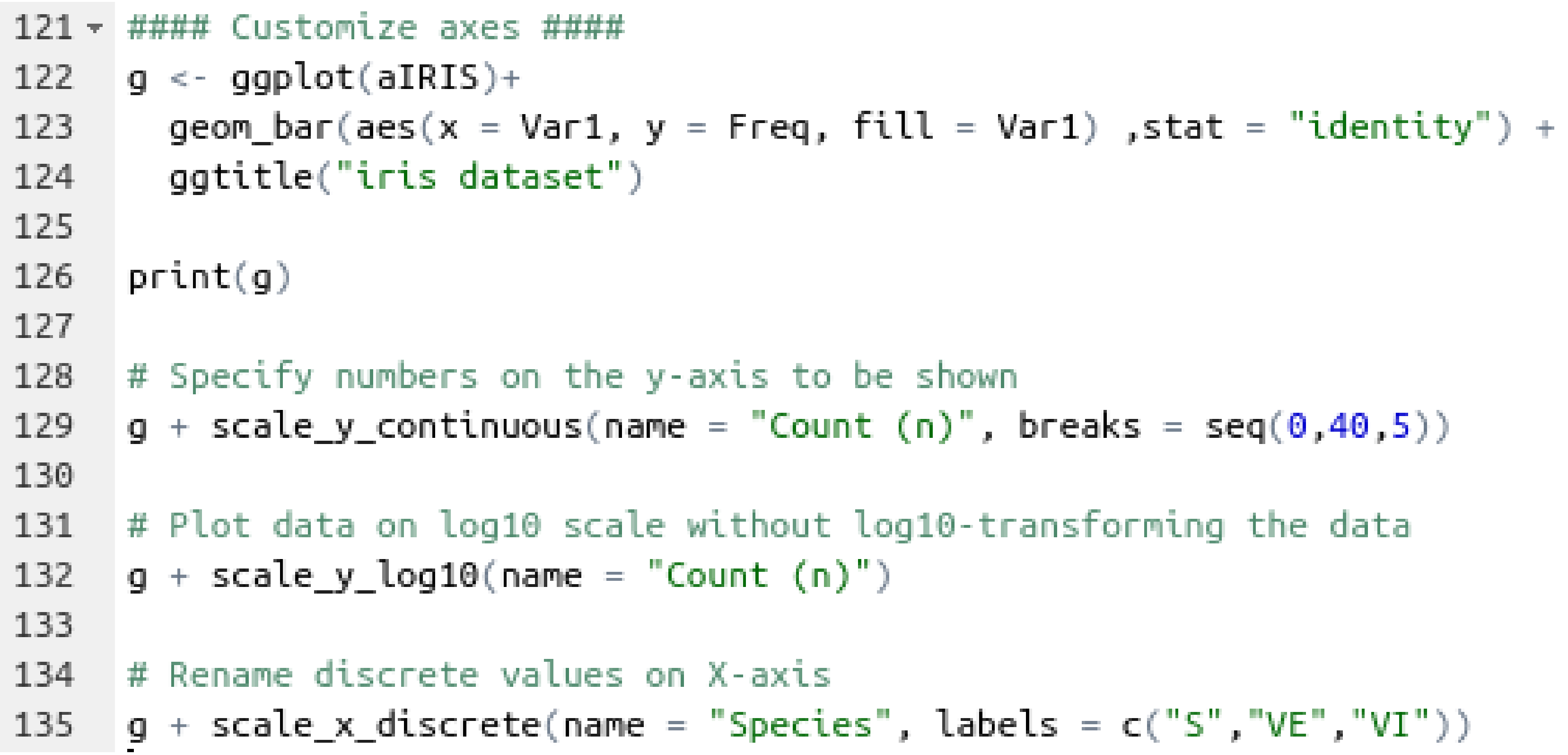
ggplot2 customization

Naming title and axes



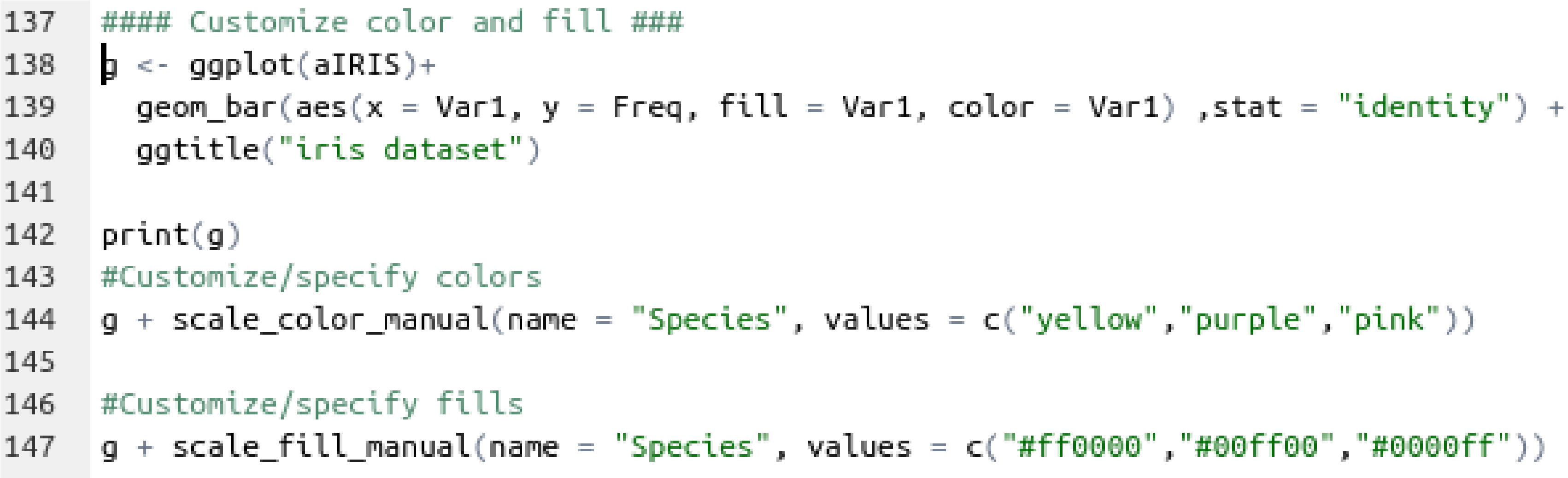
```r
#### Naming title and axes ####
print(aIRIS)
g <- ggplot(aIRIS)+
  geom_bar(aes(x = Var1, y = Freq, fill = Var1) ,stat = "identity")
print(g)
g <- g + ggtitle("iris dataset") # Add title
print(g)
g <- g + ylab("Count (n)") +  # Add/change axis titles
  xlab("Species")
print(g)
```

Customize axes



```r
#### Customize axes ####
g <- ggplot(aIRIS)+
  geom_bar(aes(x = Var1, y = Freq, fill = Var1) ,stat = "identity") +
  ggtitle("iris dataset")

print(g)

# Specify numbers on the y-axis to be shown
g + scale_y_continuous(name = "Count (n)", breaks = seq(0,40,5))

# Plot data on log10 scale without log10-transforming the data
g + scale_y_log10(name = "Count (n)")

# Rename discrete values on X-axis
g + scale_x_discrete(name = "Species", labels = c("S","VE","VI"))
```

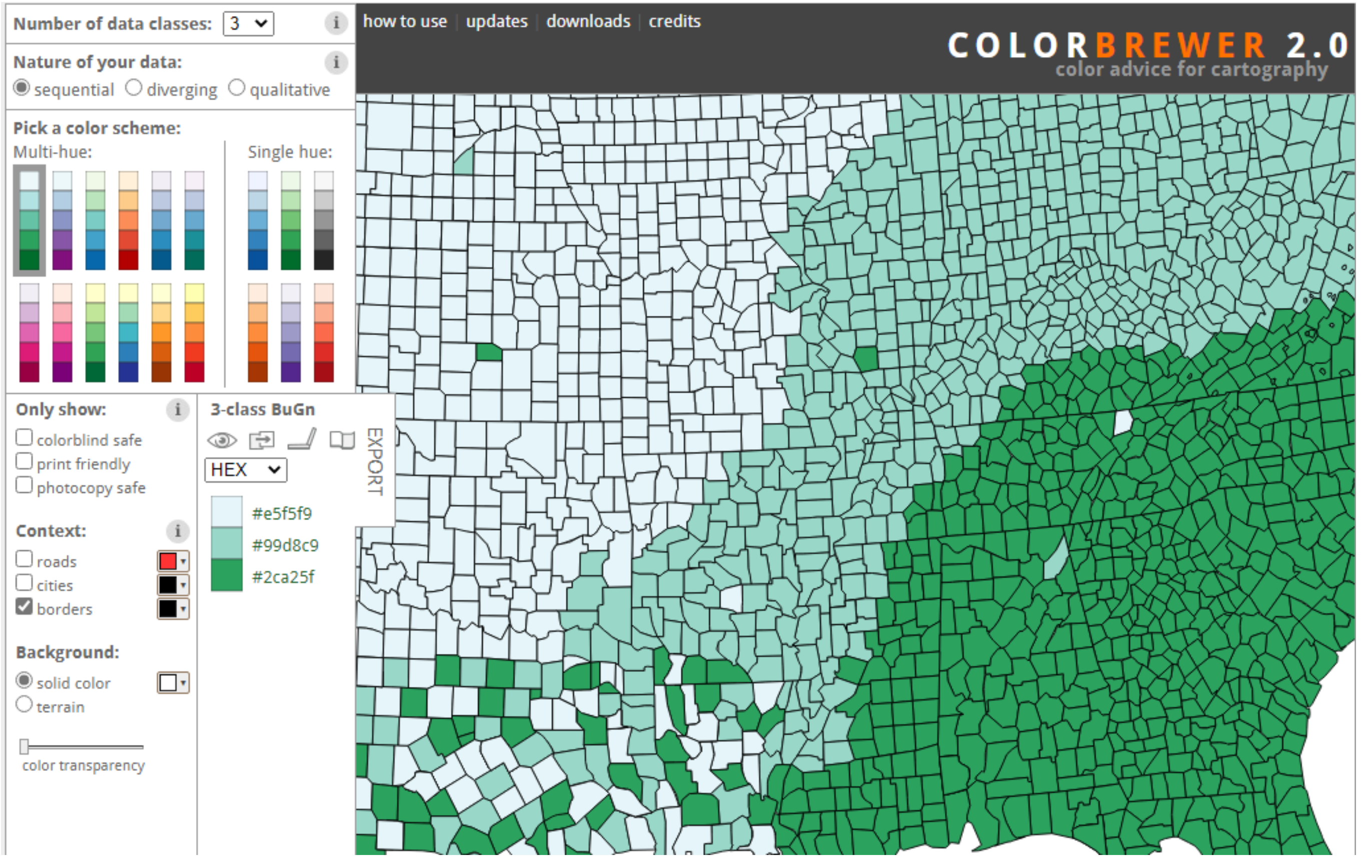
Customize color and fill



```r
#### Customize color and fill ###
g <- ggplot(aIRIS)+
  geom_bar(aes(x = Var1, y = Freq, fill = Var1, color = Var1) ,stat = "identity") +
  ggtitle("iris dataset")

print(g)
#Customize/specify colors
g + scale_color_manual(name = "Species", values = c("yellow","purple","pink"))

#Customize/specify fills
g + scale_fill_manual(name = "Species", values = c("#ff0000","#00ff00","#0000ff"))
```

Color brewer

- https://colorbrewer2.org/



Number of data classes: 3

Nature of your data:
● sequential ○ diverging ○ qualitative

Pick a color scheme:
Multi-hue: Single hue:

3-class BuGn

HEX

🟦 #e5f5f9
🟩 #99d8c9
🟩 #2ca25f

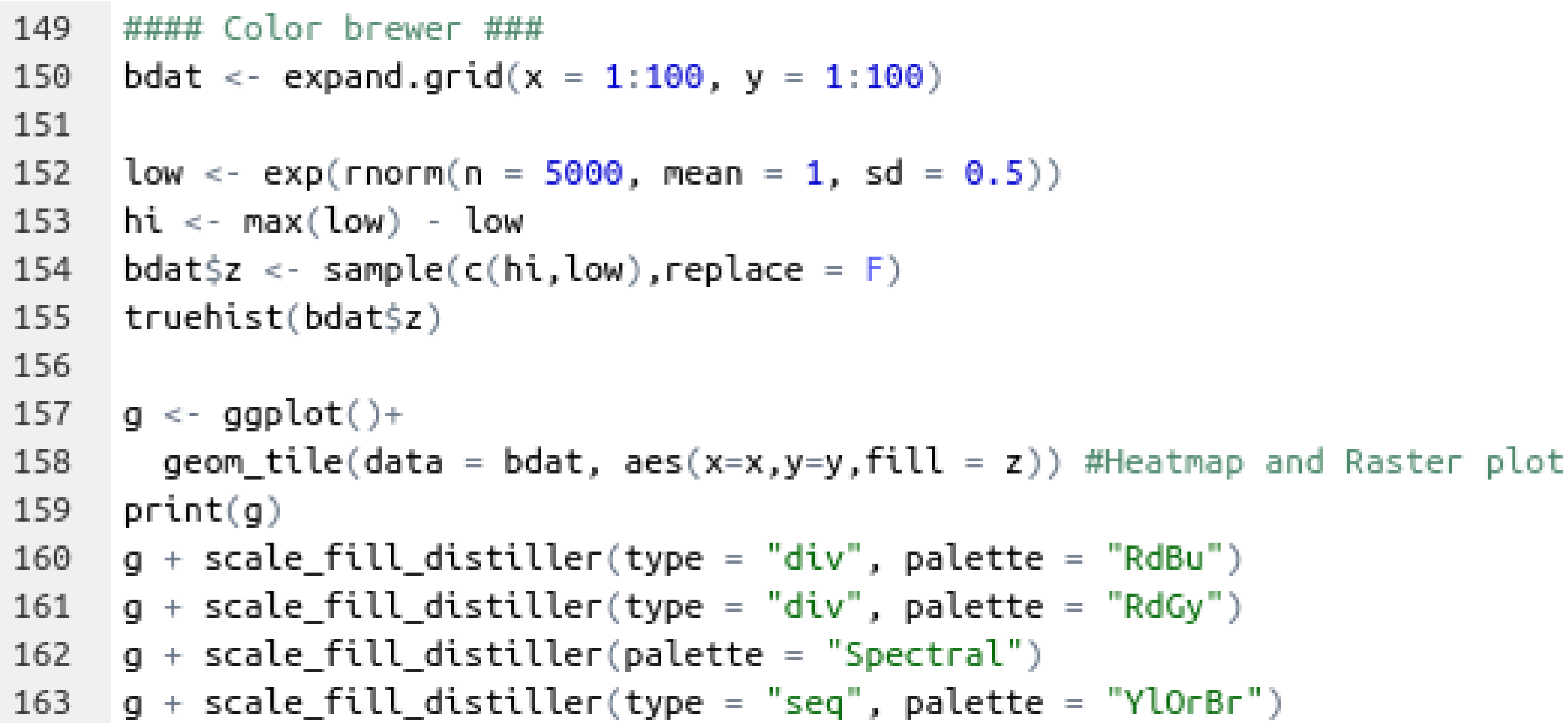
Only show:
☐ colorblind safe
☐ print friendly
☐ photocopy safe

Context:
☐ roads
☐ cities
☑ borders

Background:
● solid color
○ terrain

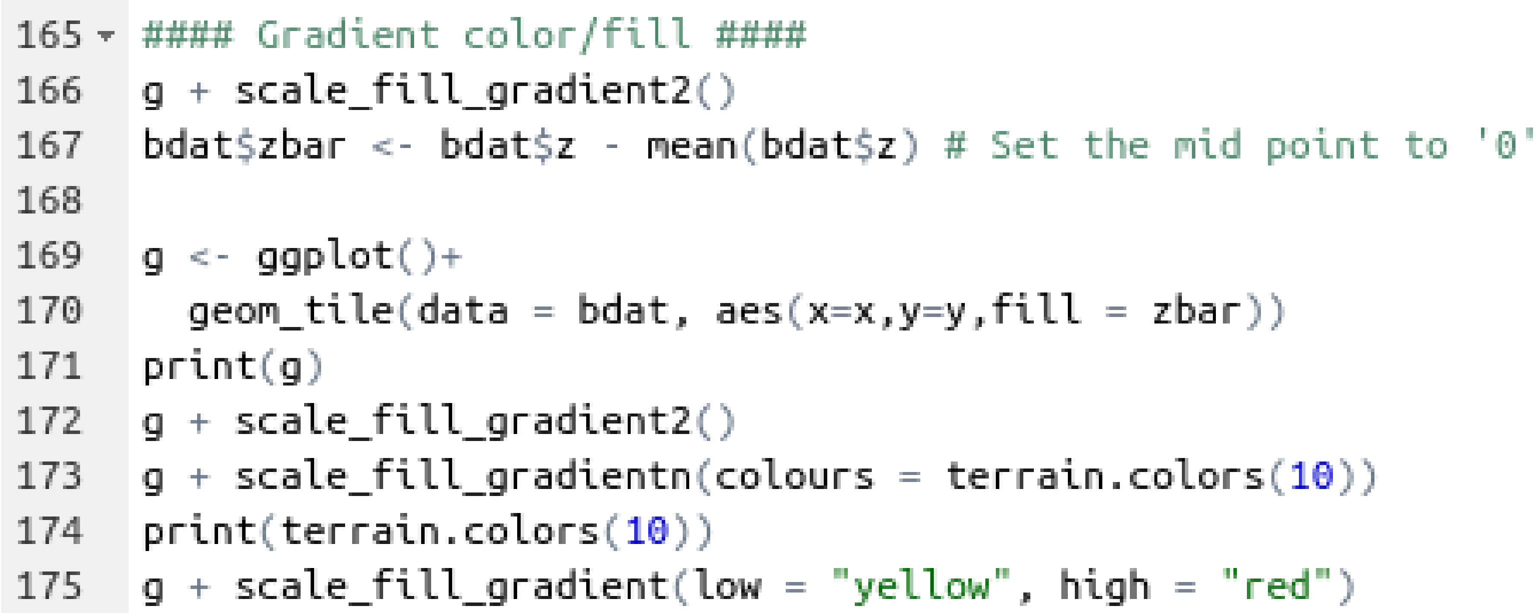
color transparency

Color brewer



```r
#### Color brewer ###
bdat <- expand.grid(x = 1:100, y = 1:100)

low <- exp(rnorm(n = 5000, mean = 1, sd = 0.5))
hi <- max(low) - low
bdat$z <- sample(c(hi,low),replace = F)
truehist(bdat$z)

g <- ggplot()+
  geom_tile(data = bdat, aes(x=x,y=y,fill = z)) #Heatmap and Raster plot
print(g)
g + scale_fill_distiller(type = "div", palette = "RdBu")
g + scale_fill_distiller(type = "div", palette = "RdGy")
g + scale_fill_distiller(palette = "Spectral")
g + scale_fill_distiller(type = "seq", palette = "YlOrBr")
```

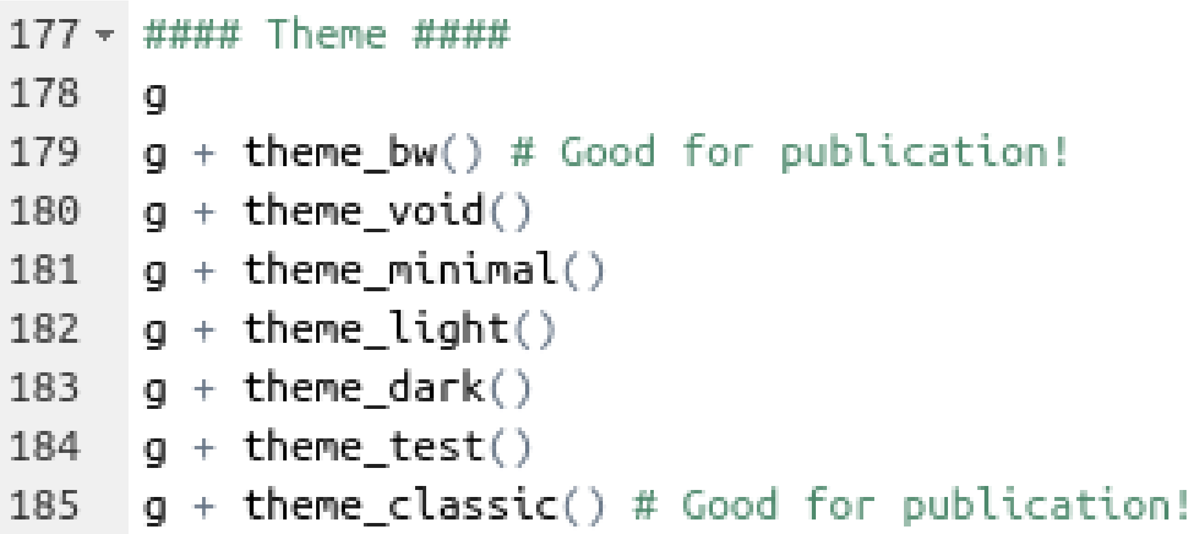
Gradient color/fill



```
#### Gradient color/fill ####
g + scale_fill_gradient2()
bdat$zbar <- bdat$z - mean(bdat$z) # Set the mid point to '0'

g <- ggplot()+
  geom_tile(data = bdat, aes(x=x,y=y,fill = zbar))
print(g)
g + scale_fill_gradient2()
g + scale_fill_gradientn(colours = terrain.colors(10))
print(terrain.colors(10))
g + scale_fill_gradient(low = "yellow", high = "red")
```

Theme!!!

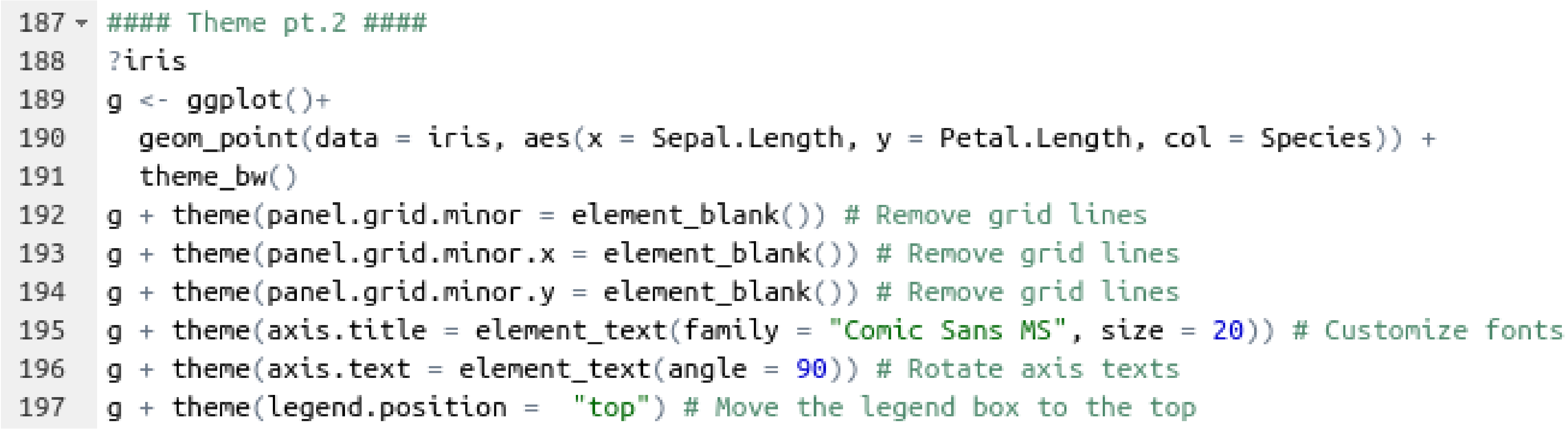
- Customize ‘static’ elements of the plot



```
#### Theme ####
g
g + theme_bw() # Good for publication!
g + theme_void()
g + theme_minimal()
g + theme_light()
g + theme_dark()
g + theme_test()
g + theme_classic() # Good for publication!
```

Theme!!!

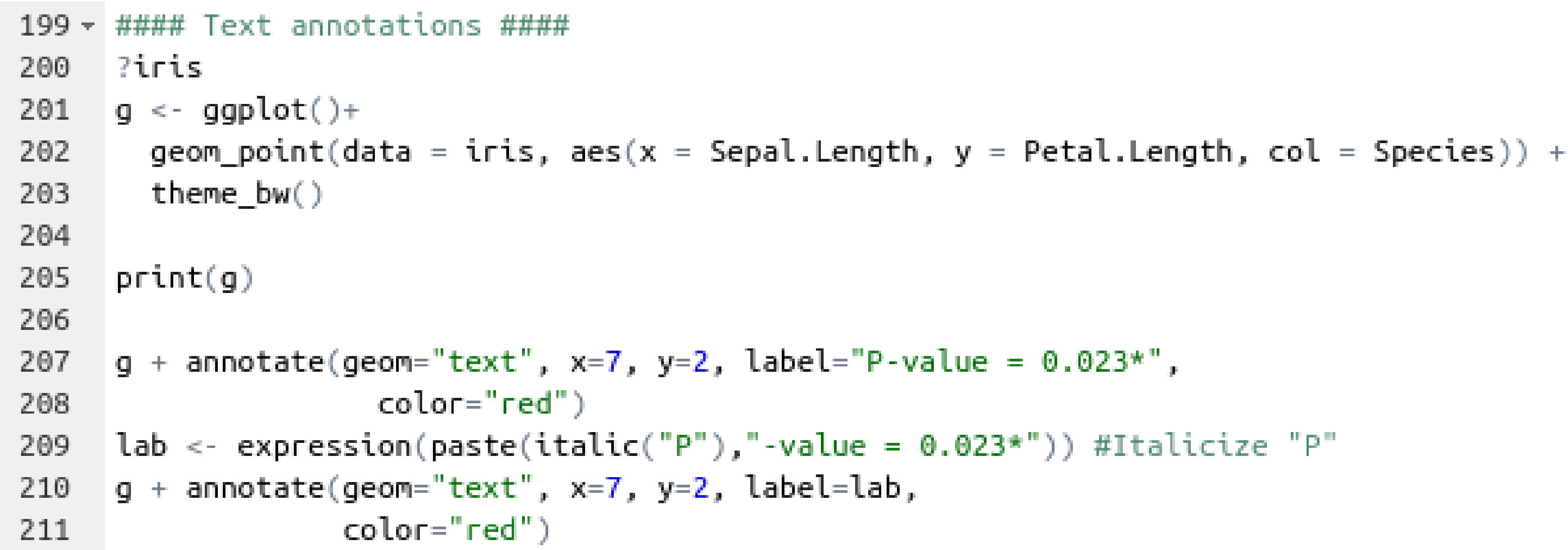
- Customize ‘static’ elements of the plot
- There are so many “elements” in the theme so please refer to “https://ggplot2.tidyverse.org/reference/theme.html”



```r
#### Theme pt.2 ####
?iris
g <- ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  theme_bw()
g + theme(panel.grid.minor = element_blank()) # Remove grid lines
g + theme(panel.grid.minor.x = element_blank()) # Remove grid lines
g + theme(panel.grid.minor.y = element_blank()) # Remove grid lines
g + theme(axis.title = element_text(family = "Comic Sans MS", size = 20)) # Customize fonts
g + theme(axis.text = element_text(angle = 90)) # Rotate axis texts
g + theme(legend.position =  "top") # Move the legend box to the top
```

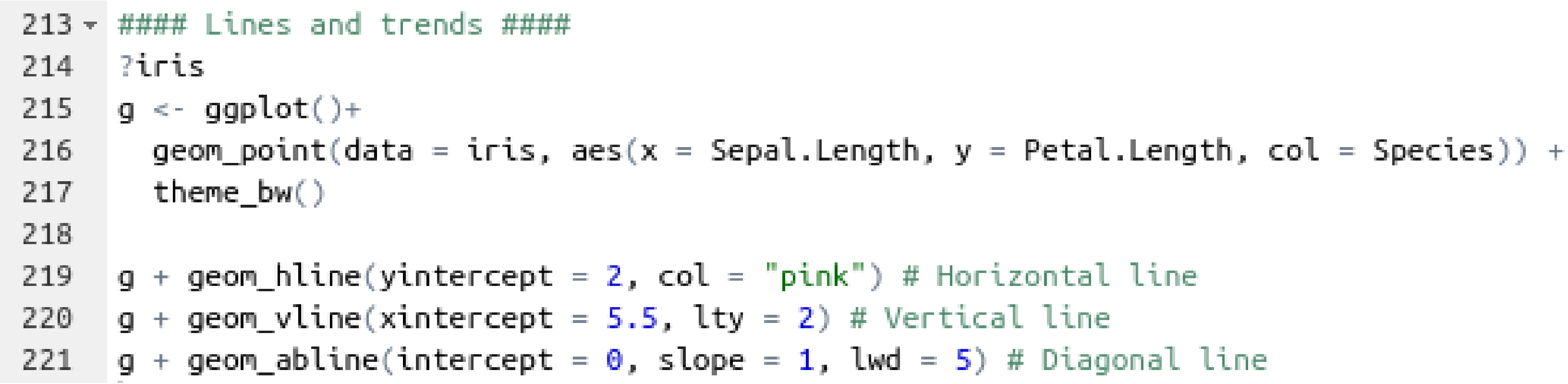
Add other elements to ggplot2

Text annotations



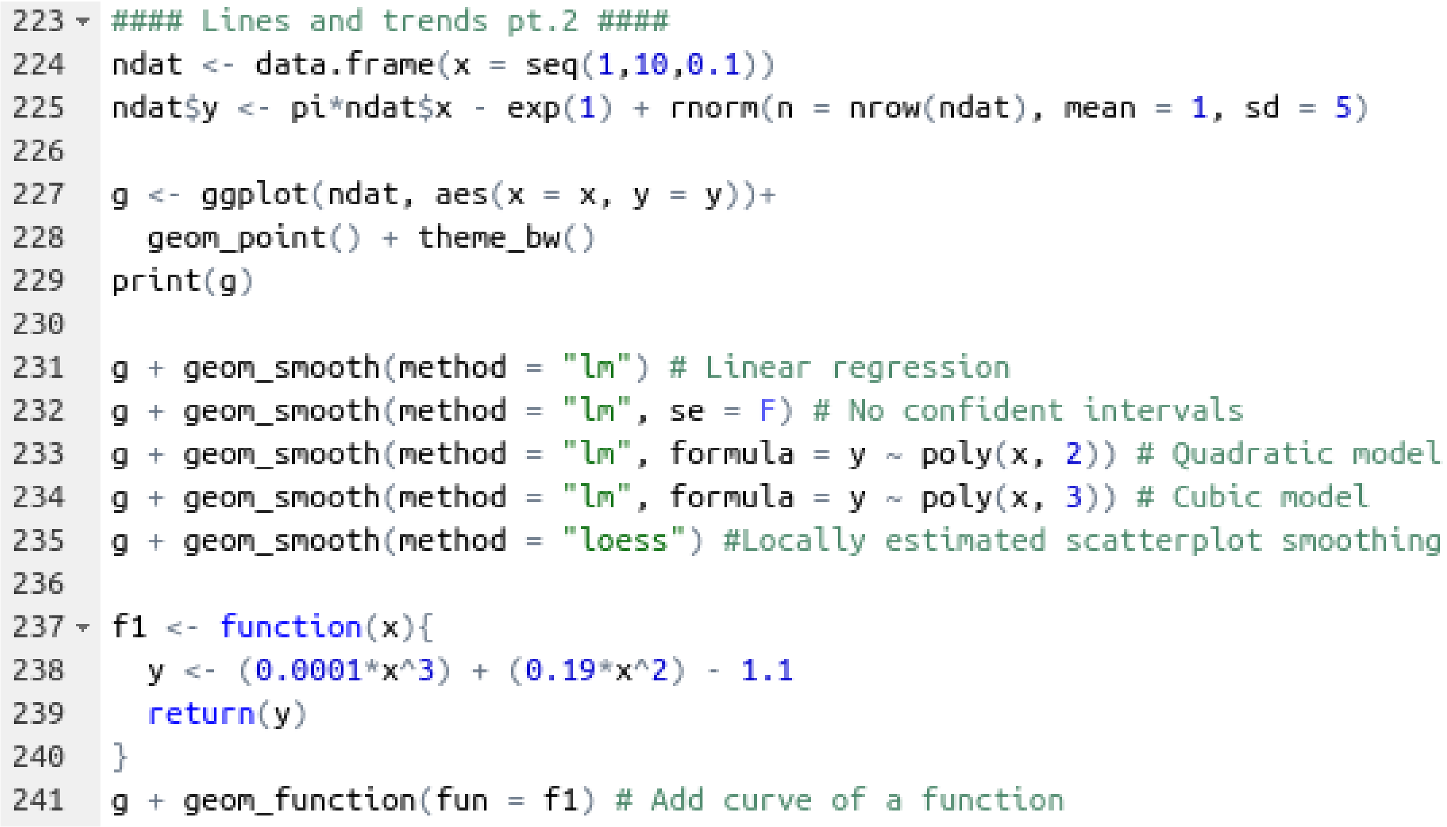
```r
#### Text annotations ####
?iris
g <- ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  theme_bw()

print(g)

g + annotate(geom="text", x=7, y=2, label="P-value = 0.023*",
               color="red")
lab <- expression(paste(italic("P"),"-value = 0.023*")) #Italicize "P"
g + annotate(geom="text", x=7, y=2, label=lab,
             color="red")
```

Lines and trends



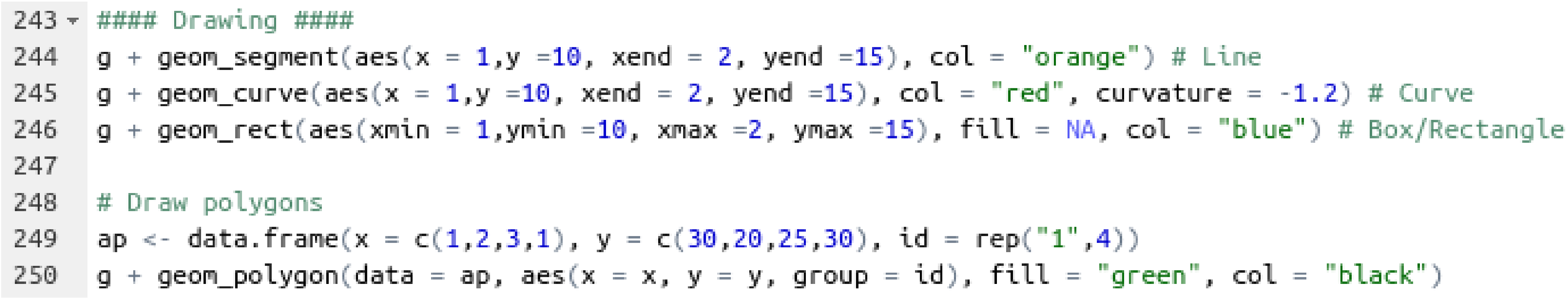
```
#### Lines and trends ####
?iris
g <- ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  theme_bw()

g + geom_hline(yintercept = 2, col = "pink") # Horizontal line
g + geom_vline(xintercept = 5.5, lty = 2) # Vertical line
g + geom_abline(intercept = 0, slope = 1, lwd = 5) # Diagonal line
```

Lines and trends



```r
#### Lines and trends pt.2 ####
ndat <- data.frame(x = seq(1,10,0.1))
ndat$y <- pi*ndat$x - exp(1) + rnorm(n = nrow(ndat), mean = 1, sd = 5)

g <- ggplot(ndat, aes(x = x, y = y))+
  geom_point() + theme_bw()
print(g)

g + geom_smooth(method = "lm") # Linear regression
g + geom_smooth(method = "lm", se = F) # No confident intervals
g + geom_smooth(method = "lm", formula = y ~ poly(x, 2)) # Quadratic model
g + geom_smooth(method = "lm", formula = y ~ poly(x, 3)) # Cubic model
g + geom_smooth(method = "loess") #Locally estimated scatterplot smoothing

f1 <- function(x){
  y <- (0.0001*x^3) + (0.19*x^2) - 1.1
  return(y)
}
g + geom_function(fun = f1) # Add curve of a function
```

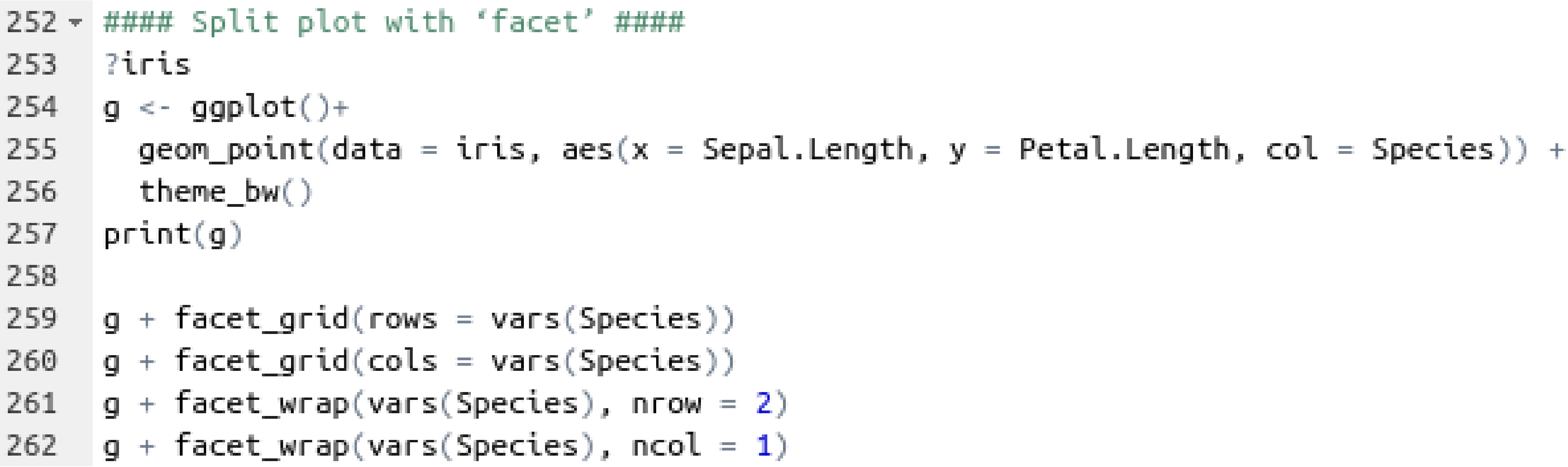
Drawing



```r
#### Drawing ####
g + geom_segment(aes(x = 1,y =10, xend = 2, yend =15), col = "orange") # Line
g + geom_curve(aes(x = 1,y =10, xend = 2, yend =15), col = "red", curvature = -1.2) # Curve
g + geom_rect(aes(xmin = 1,ymin =10, xmax =2, ymax =15), fill = NA, col = "blue") # Box/Rectangle

# Draw polygons
ap <- data.frame(x = c(1,2,3,1), y = c(30,20,25,30), id = rep("1",4))
g + geom_polygon(data = ap, aes(x = x, y = y, group = id), fill = "green", col = "black")
```

Multiple plots

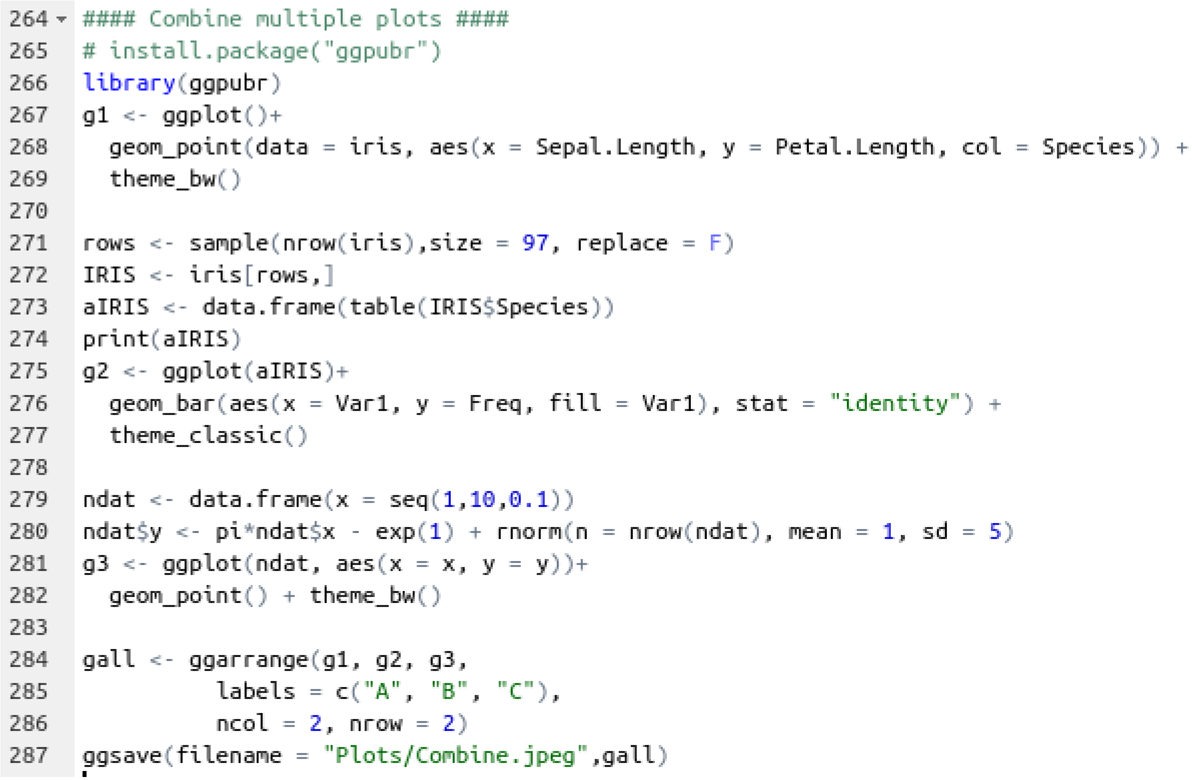
Split plot with ‘facet’



```
#### Split plot with ‘facet’ ####
?iris
g <- ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  theme_bw()
print(g)

g + facet_grid(rows = vars(Species))
g + facet_grid(cols = vars(Species))
g + facet_wrap(vars(Species), nrow = 2)
g + facet_wrap(vars(Species), ncol = 1)
```

Combine multiple plots

- ggarrange() in “ggpubr”



```r
#### Combine multiple plots ####
# install.package("ggpubr")
library(ggpubr)
g1 <- ggplot()+
  geom_point(data = iris, aes(x = Sepal.Length, y = Petal.Length, col = Species)) +
  theme_bw()

rows <- sample(nrow(iris),size = 97, replace = F)
IRIS <- iris[rows,]
aIRIS <- data.frame(table(IRIS$Species))
print(aIRIS)
g2 <- ggplot(aIRIS)+
  geom_bar(aes(x = Var1, y = Freq, fill = Var1), stat = "identity") +
  theme_classic()

ndat <- data.frame(x = seq(1,10,0.1))
ndat$y <- pi*ndat$x - exp(1) + rnorm(n = nrow(ndat), mean = 1, sd = 5)
g3 <- ggplot(ndat, aes(x = x, y = y))+
  geom_point() + theme_bw()

gall <- ggarrange(g1, g2, g3,
          labels = c("A", "B", "C"),
          ncol = 2, nrow = 2)
ggsave(filename = "Plots/Combine.jpeg",gall)
```

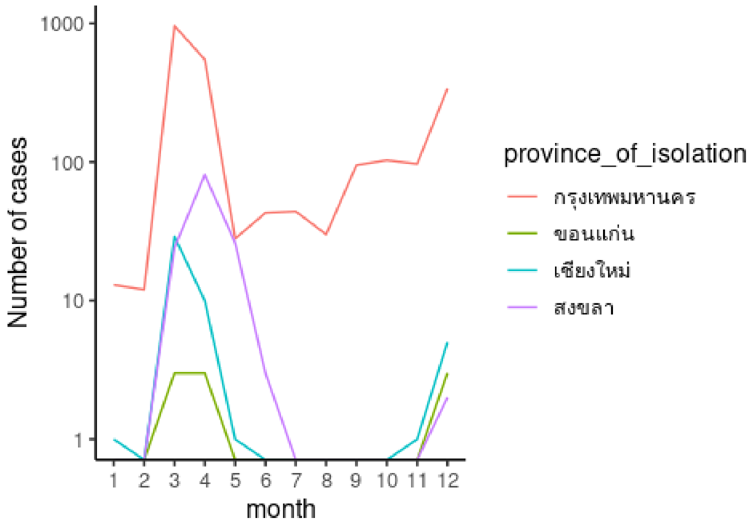
Resources

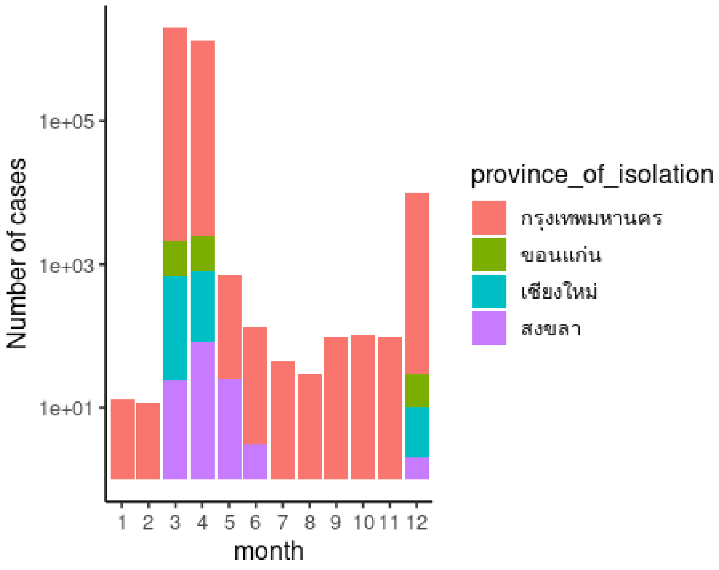
- Cheat sheets: https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf
- Tutorial: https://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html
- ggpubr (improve ggplot2 for publication): https://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/

In-class assignment

COVID-19 Data

- Download COVID-19 cases data from: https://data.go.th/dataset/8a956917-436d-4afd-a2d4-59e4dd8e906e/resource/be19a8ad-ab48-4081-b04a-8035b5b2b8d6/download/confirmed-cases.csv
- Write an R script that plots monthly cases from 1/1/2020 – 31/12/2020 (announce_date) showing data from Bangkok, Chiang Mai, Khon Kaen, Songkhla (province_of_isolation).
- Plot either a line plot or a bar graph is OK (i.e., pick one).
- Log10 scaling for the Y-axis is optional


Number of cases
month
province_of_isolation
กรุงเทพมหานคร
ขอนแก่น
เชียงใหม่
สงขลา


Number of cases
1e+05
1e+03
1e+01
month
province_of_isolation
กรุงเทพมหานคร
ขอนแก่น
เชียงใหม่
สงขลา