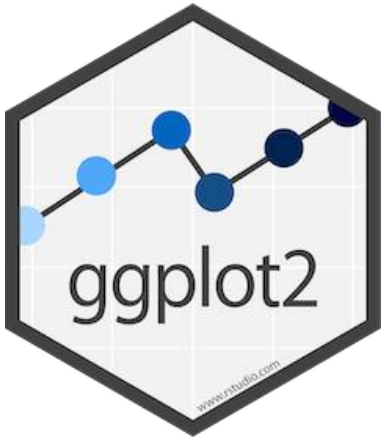


GGPLOT2 IN R

Presenter: Nguyen Le Duc Minh, MD

Sep 15th 2024

Content

- Part 1: Introduction to ggplot2
  - *Covers the basic knowledge about constructing simple ggplots and modifying the components and aesthetics.*
- Part 2: Customizing the look and feel
  - is *about more advanced customization like manipulating legend, annotations, multiplots with faceting and custom layouts*
- Part 3: Top 50 ggplot2 Visualizations
  - *applies what was learnt in part 1 and 2 to construct other types of ggplots such as bar charts, boxplots etc*
- Part 4: Principal component analysis (PCA)
- Part 5: Heatmap

Part 1: Introduction to ggplot2

Why do we use ggplot2 ?

- **ggplot2** is a plotting package that provides **helpful commands to create complex plots** from **data in a data frame.**
- It provides a **more programmatic interface** for specifying what variables to plot, how they are displayed, and general visual properties.
- Therefore, we **only need minimal changes** if the underlying data change or if we decide to change from a bar plot to a scatterplot.
- This helps in **creating publication quality plots** with minimal amounts of adjustments and tweaking.

How to install this packages ?

```
install.packages("ggplot2")
library(ggplot2)
```

Part 1: Introduction to data

Students Performance in Exams

```
> head(data)
  gender race.ethnicity parental.level.of.education       lunch test.preparation.course math.score
1 female        group B           bachelor's degree    standard                    none         72
2 female        group C                some college    standard                completed         69
3 female        group B             master's degree    standard                    none         90
4   male        group A          associate's degree free/reduced                    none         47
5   male        group C                some college    standard                    none         76
6 female        group B          associate's degree    standard                    none         71
  reading.score writing.score
1            72            74
2            90            88
3            95            93
4            57            44
5            78            75
6            83            78
```

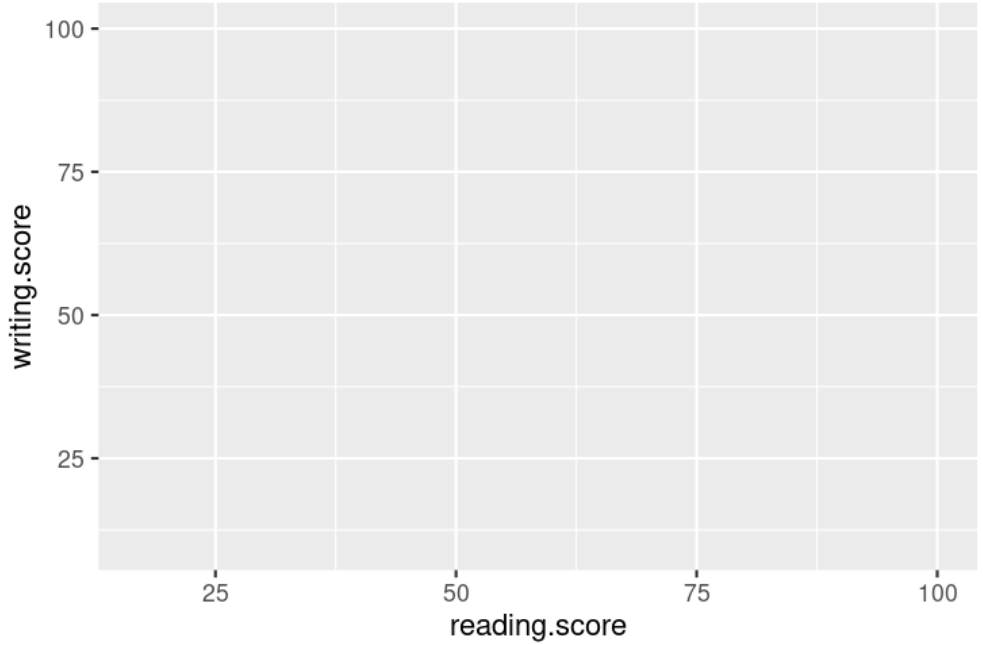
nrow=1000; ncol =8

Part 1: Understanding the Ggplot Syntax

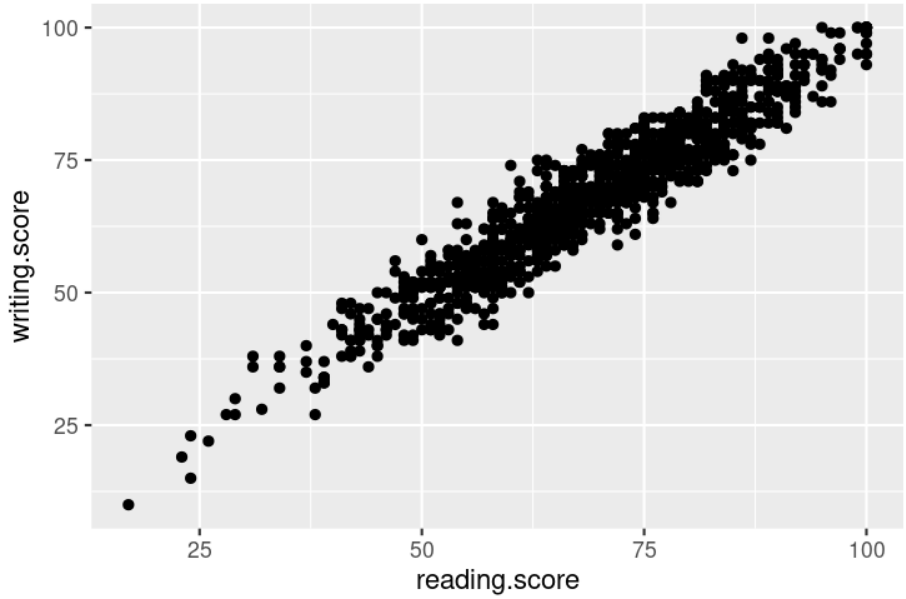
- Every ggplot2 plot has three key components:

1. Data
2. A set of aesthetic mappings between variables in the data and visual properties.
3. At least one layer which describes how to render each observation. Layers are usually created with a geom function.

```
ggplot(data, aes(x=reading.score, y=writing.score))
```

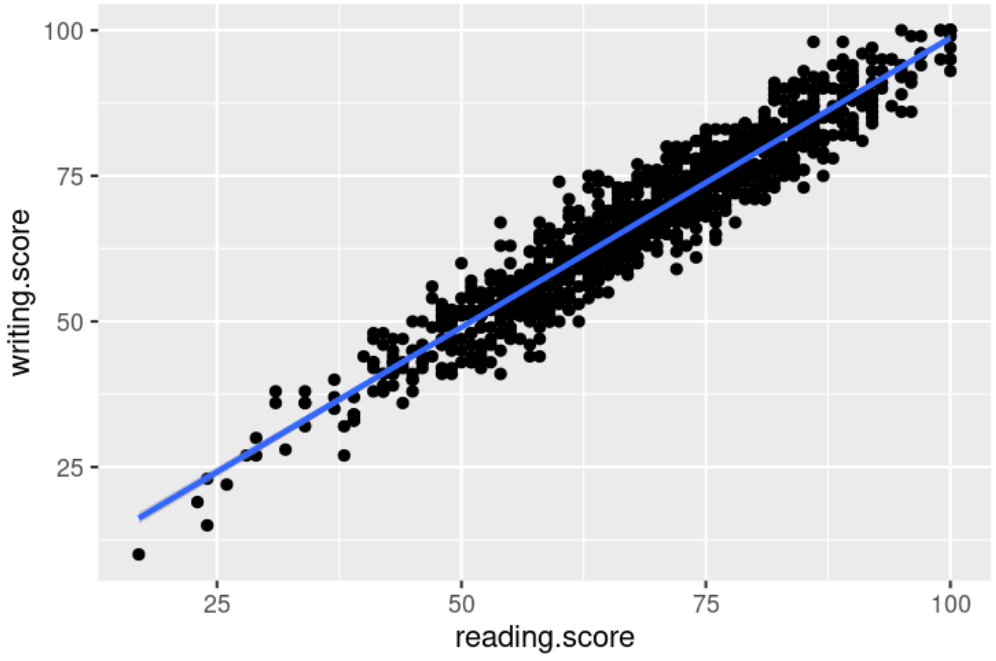



```
ggplot(data, aes(x=reading.score, y=writing.score)) +
geom_point()
```



Part 1: There are many such geom layers

```
ggplot(data, aes(x=reading.score, y=writing.score)) + geom_point() + geom_smooth(method="lm")
```




```
?geom_smooth
```

geom_smooth {ggplot2} R Documentation

Smoothed conditional means

Description

Aids the eye in seeing patterns in the presence of overplotting. `geom_smooth()` and `stat_smooth()` are effectively aliases: they both use the same arguments. Use `stat_smooth()` if you want to display the results with a non-standard geom.

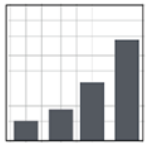
Usage

```
geom_smooth(
```

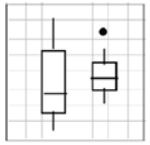
Part 1: There are many such geom layers

Different Geoms (Plot Type) in ggplot2

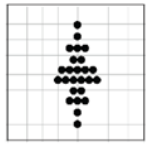
Two Variables (X,Y)
- Discrete X, continuous Y
- Visualise distribution of Y with respect to X



geom_col()
- heights of bars represent values



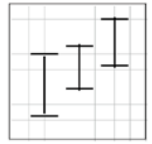
geom_boxplot()
- summarise distribution using median, hinges and whiskers



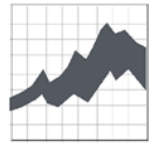
geom_jitter()
- adds jitter to prevent overplotting

geom_violin()
- mirrored density plot (smoothed distribution)

Visualising Errors and Uncertainties



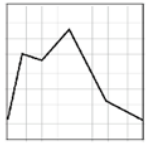
geom_errorbar()
- uncertainty in continuous Y against discrete X



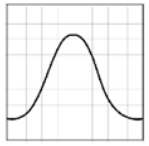
geom_ribbon()
- uncertainty in continuous Y against continuous X

One Variable (X)
- Continuous X
- Visualise distribution of X

geom_histogram()
- divide X into bins and count no. observation

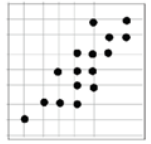


geom_freqpoly()
- display counts with lines
- able to overlay multiple distributions

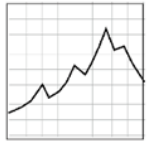


geom_density()
- smoothed version of the histogram

Two Variables (X,Y)
- Continuous X, continuous Y
- Visualise relationship between X and Y

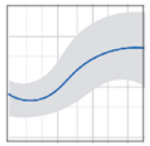


geom_point()
- scatterplot of X vs Y

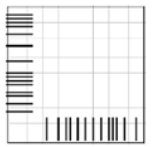


geom_line()
- connect data points, ordered by X
- alt: geom_path()

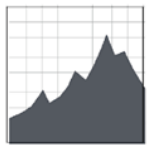
geom_text()
- labelling data points



geom_smooth()
- add smoothed curve
- helps to see trends



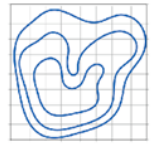
geom_rug()
- supplement 2D plot with 1D distribution along X and Y



geom_area()
- can be stacked to see cumulative contribution

Contour Plots
- Representing a third dimension using contours

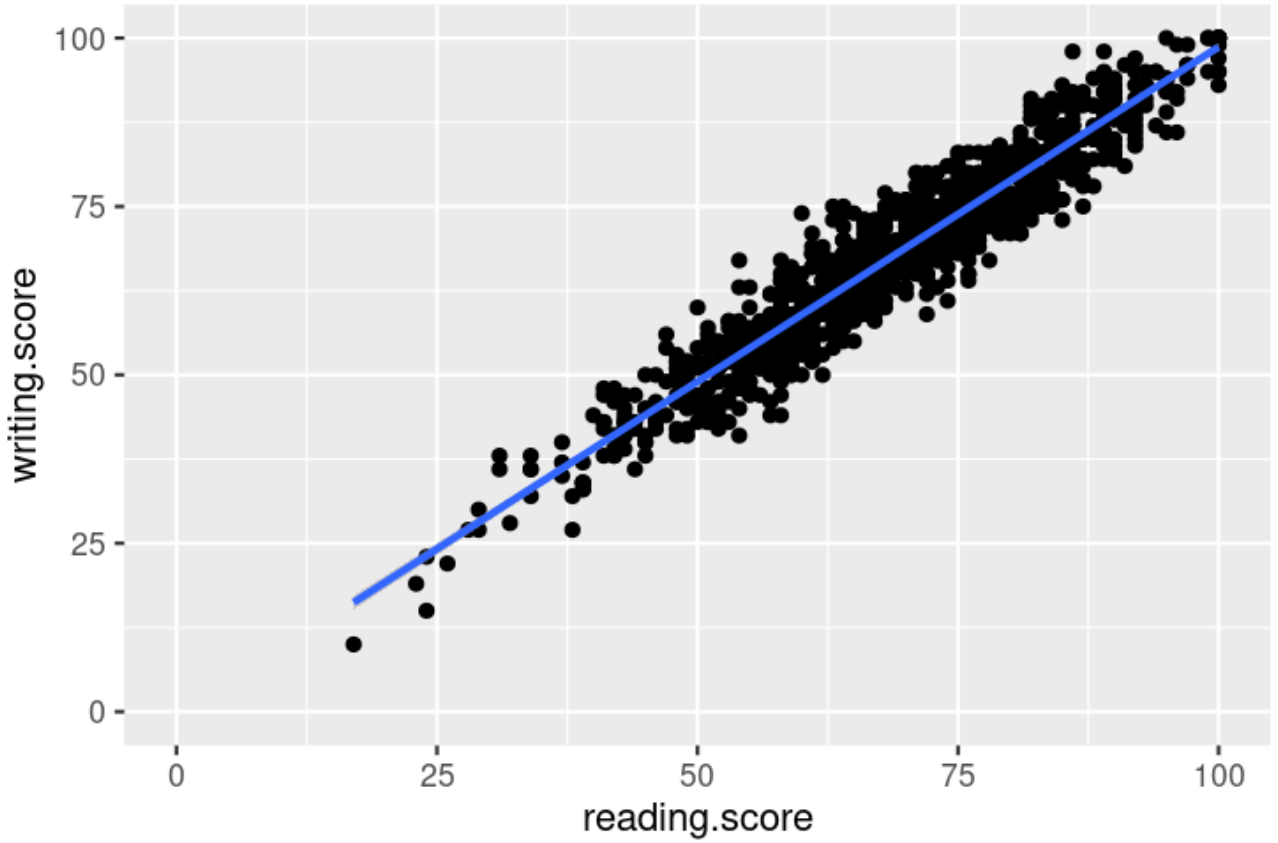
geom_density2d()
- contour represents 2D density of data points



geom_contour()
- contour represents z-axis value / height

Part 1: Adjusting the X and Y axis limits

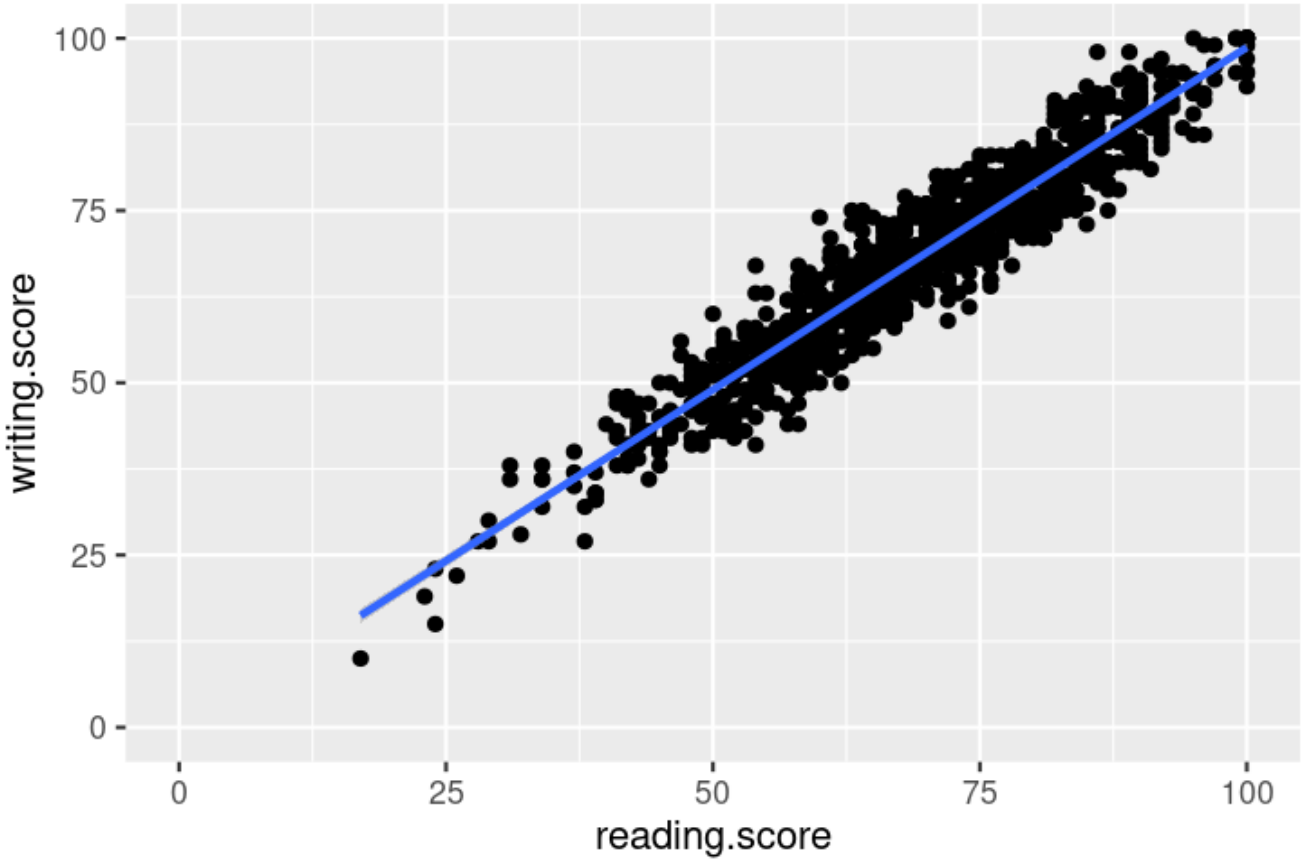
***Method 1**: Using xlim, ylim*




```
g <- ggplot(data, aes(x=reading.score,
y=writing.score)) + geom_point() +
  geom_smooth(method="lm")


# edit the value of axis
g + xlim(c(0, 100)) + ylim(c(0, 100))
```

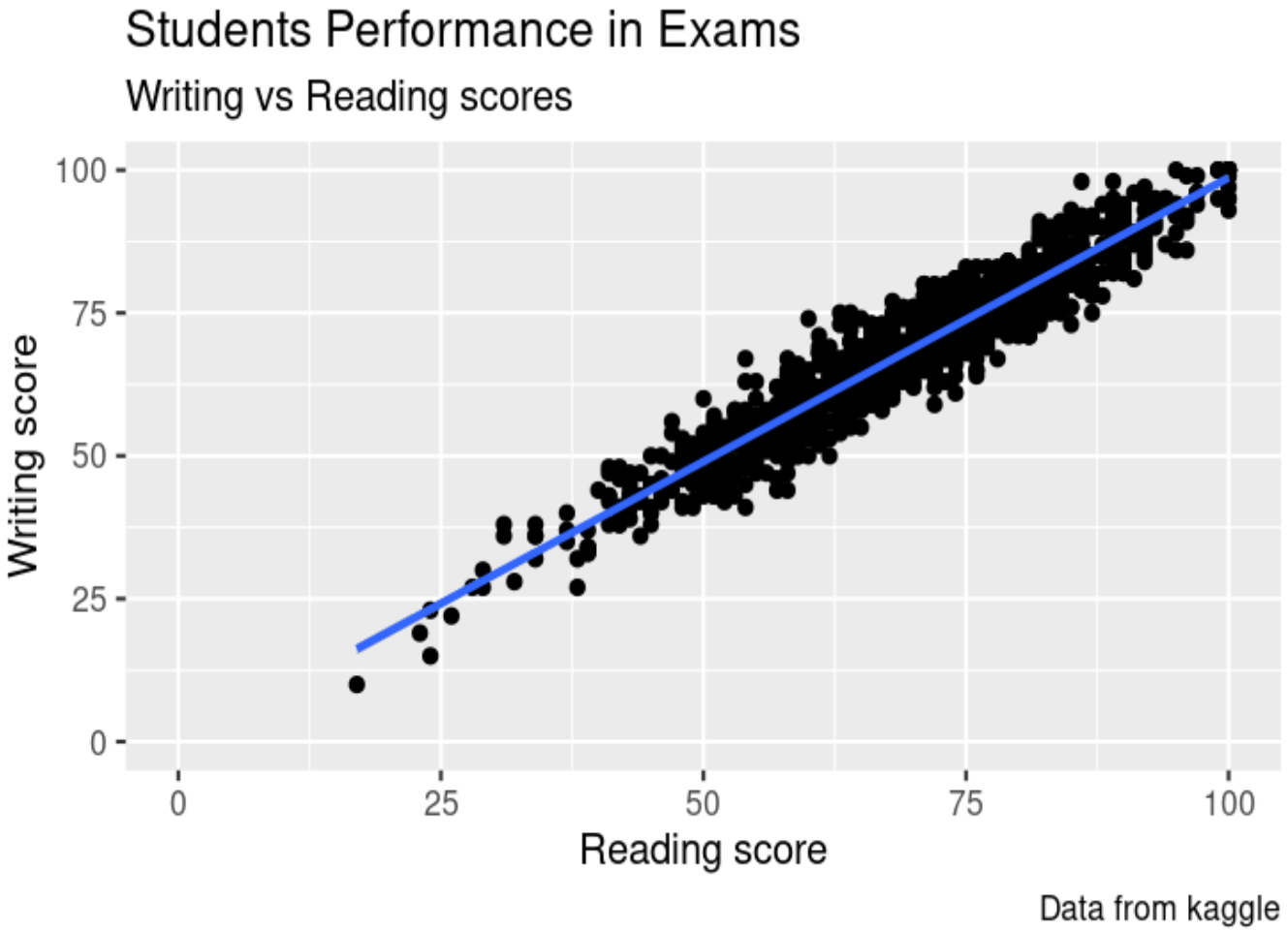
Part 1: Adjusting the X and Y axis limits

Method 2: *Zooming In*




```
g <- ggplot(data, aes(x=reading.score,
y=writing.score)) + geom_point() +
  geom_smooth(method="lm")


# zooms in
g1 <- g + coord_cartesian(xlim=c(0,100), ylim=c(0,
100))  plot(g1)
```

Part 1: How to Change the Title and Axis Labels



 # Method 1

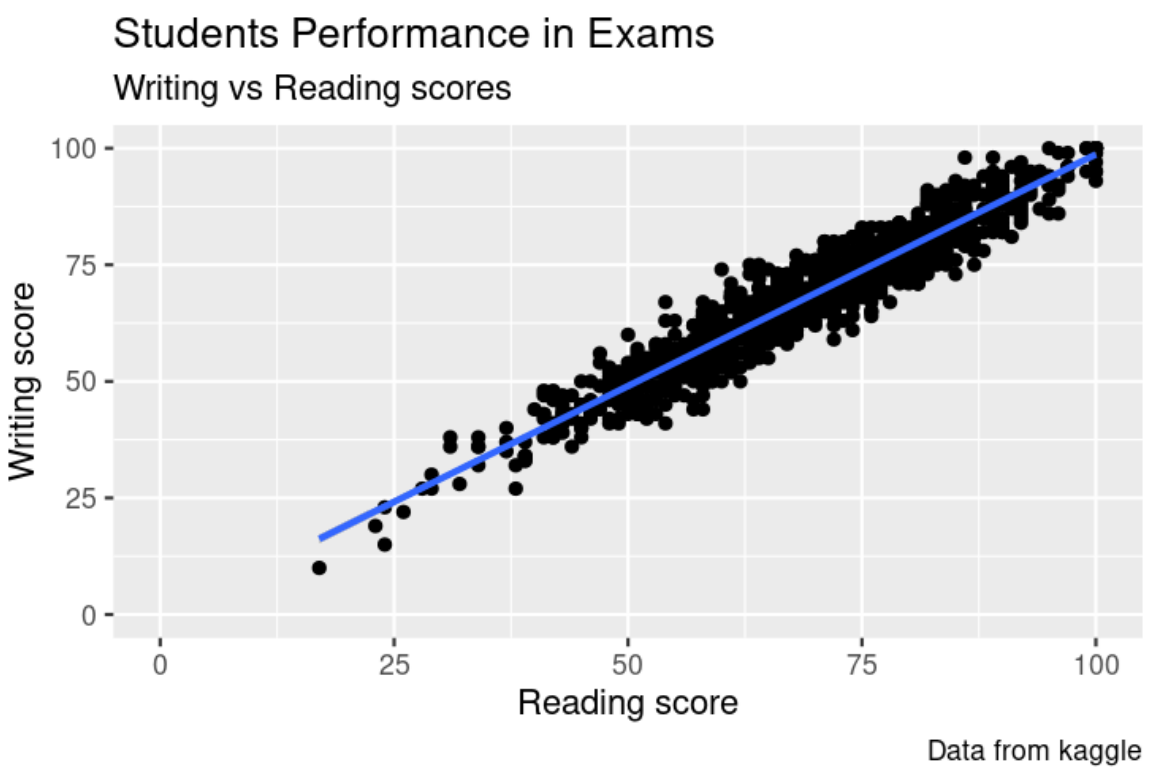
```
g1 + labs(title="Students Performance in Exams",
          subtitle="Writing vs Reading scores",
          y="Writing score",
          x="Reading score",
          caption="Data from kaggle")
```

 # Method 2

```
g1 + ggtitle("Students Performance in Exams",
subtitle="Writing vs Reading scores") +
            xlab("Writing score") + ylab("Reading
score")
```

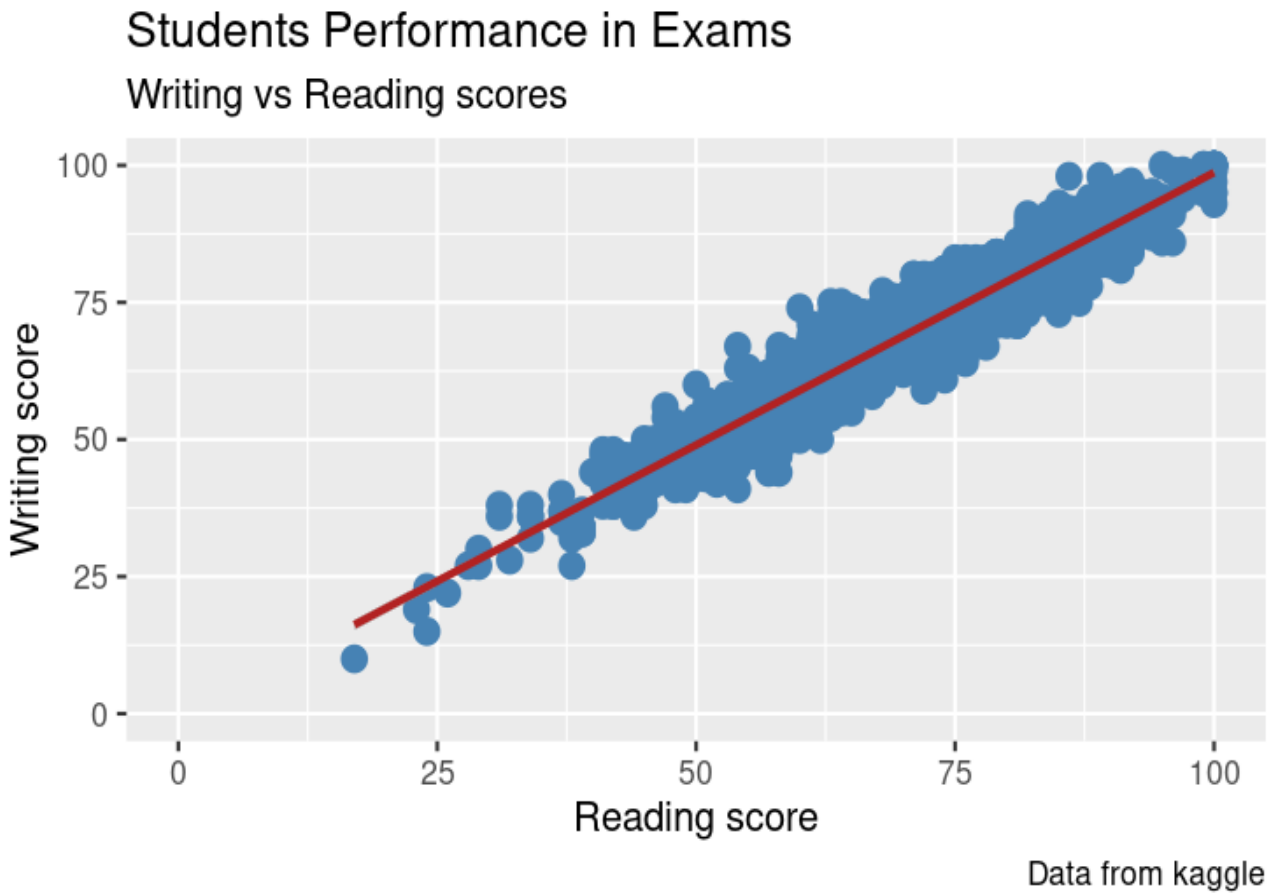
Part 1: So here is the full function call

```
library(ggplot2)
ggplot(data, aes(x=reading.score, y=writing.score)) + geom_point() +
  geom_smooth(method="lm") + xlim(c(0, 100)) + ylim(c(0, 100)))+
  labs(title="Students Performance in Exams",
       subtitle="Writing vs Reading scores",
       y="Writing score",
       x="Reading score",
       caption="Data from kaggle")
```



Part 1:How to Change the Color and Size of Points

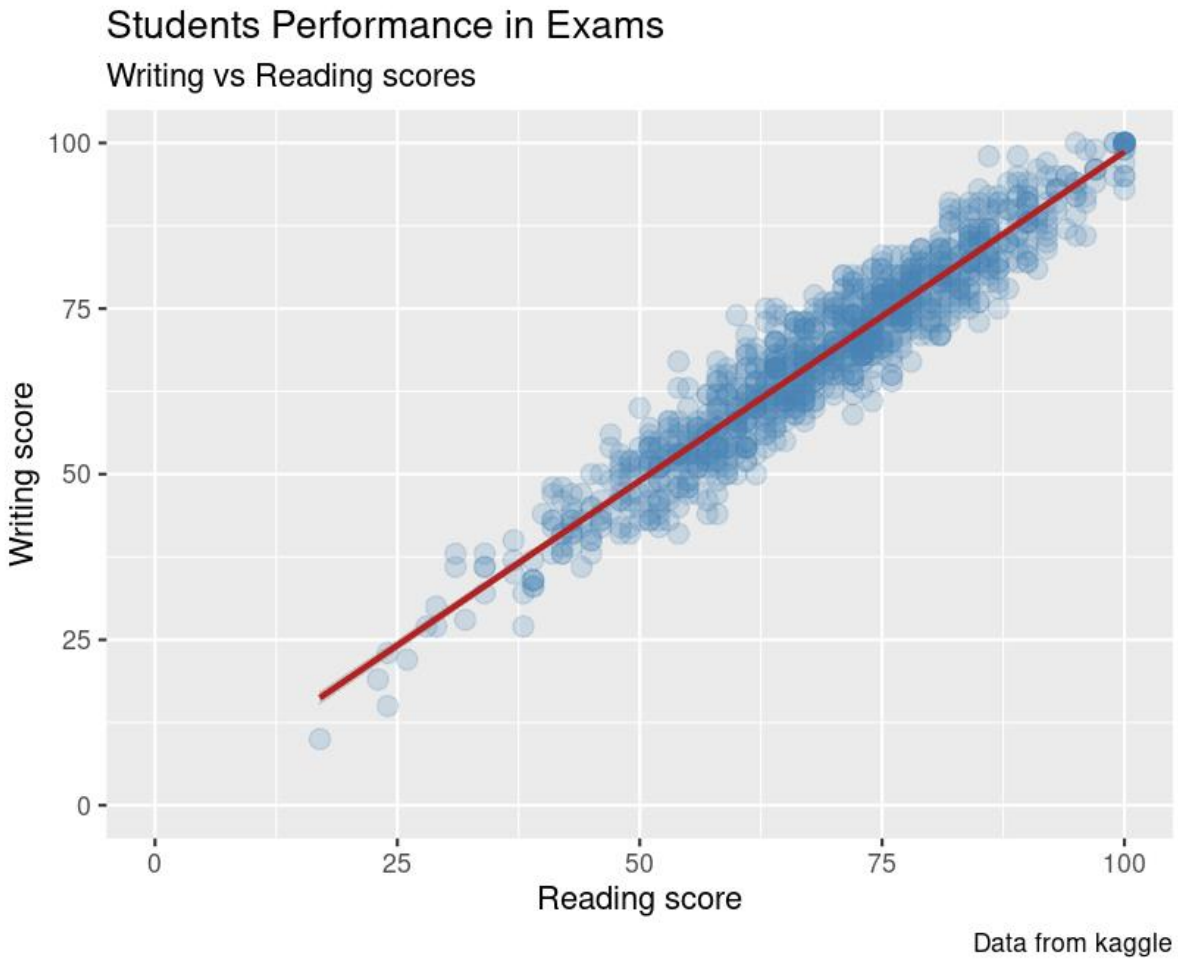
We can change the aesthetics of a geom layer by modifying the respective geoms. Let's change the color of the points and the line to a static value



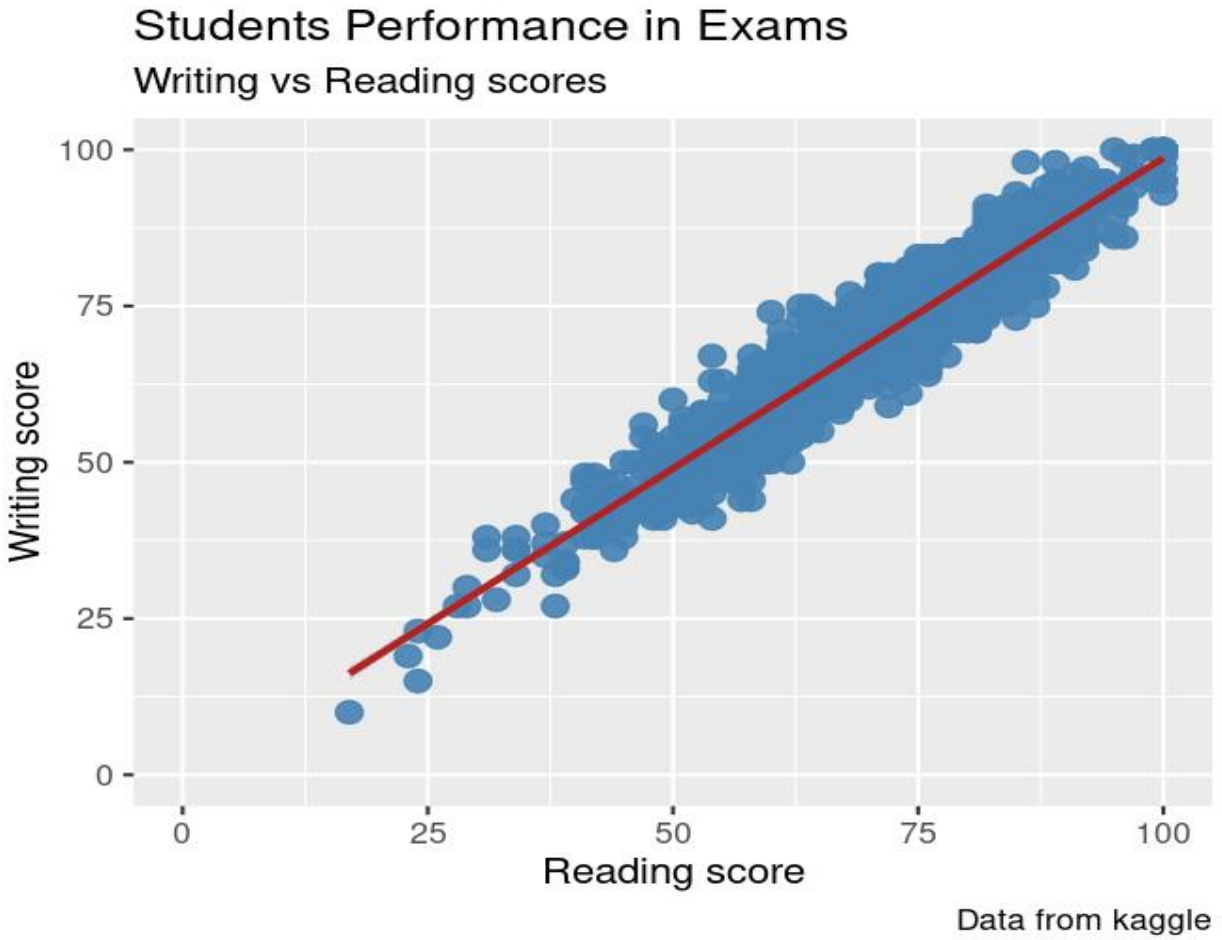

```
ggplot(data, aes(x=reading.score, y=writing.score)) +
  geom_point(col="steelblue", size=3) +
  geom_smooth(method="lm", col="firebrick") +
  xlim(c(0, 100)) + ylim(c(0, 100))+
  labs(title="Students Performance in Exams",
       subtitle="Writing vs Reading scores",
       y="Writing score",
       x="Reading score",
       caption="Data from kaggle")
```

Part 1:How to avoid overlap datapoint

geom_point(col="steelblue", size=3, **alpha = 0.2**)



geom_point(col="steelblue", size=3, **alpha = 0.9**)



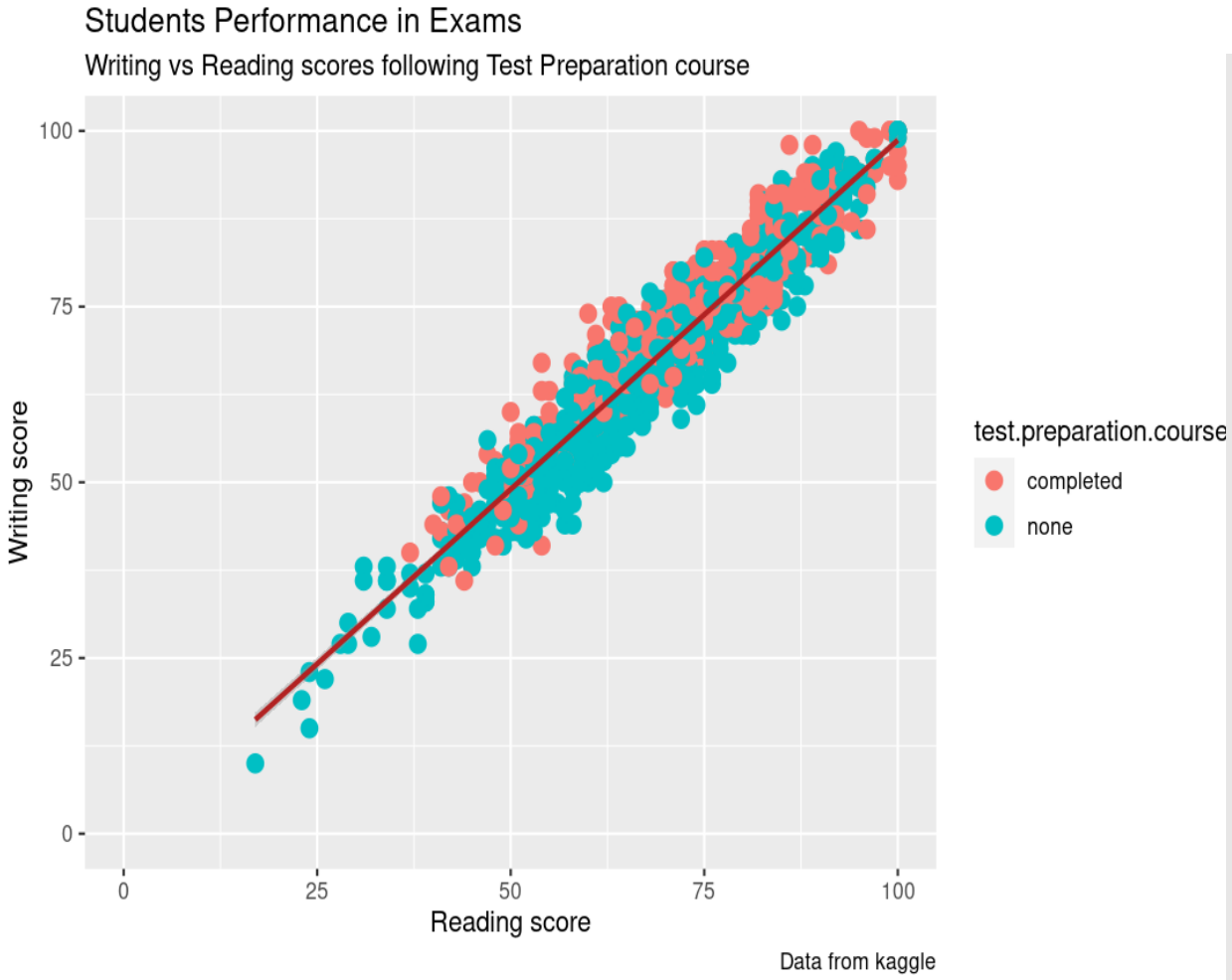
Part 1:How to Change the Color To Reflect Categories in Another Column?

Students Performance in Exams

```
> head(data)
  gender race.ethnicity parental.level.of.education       lunch test.preparation.course math.score
1 female        group B           bachelor's degree    standard                  none         72
2 female        group C                some college    standard             completed         69
3 female        group B             master's degree    standard                  none         90
4   male        group A          associate's degree free/reduced                  none         47
5   male        group C                some college    standard                  none         76
6 female        group B          associate's degree    standard                  none         71
  reading.score writing.score
1            72            74
2            90            88
3            95            93
4            57            44
5            78            75
6            83            78
```

Part 1:How to Change the Color To Reflect Categories in Another Column?

Suppose if we want the color to change based on another column in the source dataset (midwest), it must be specified inside the aes() function.

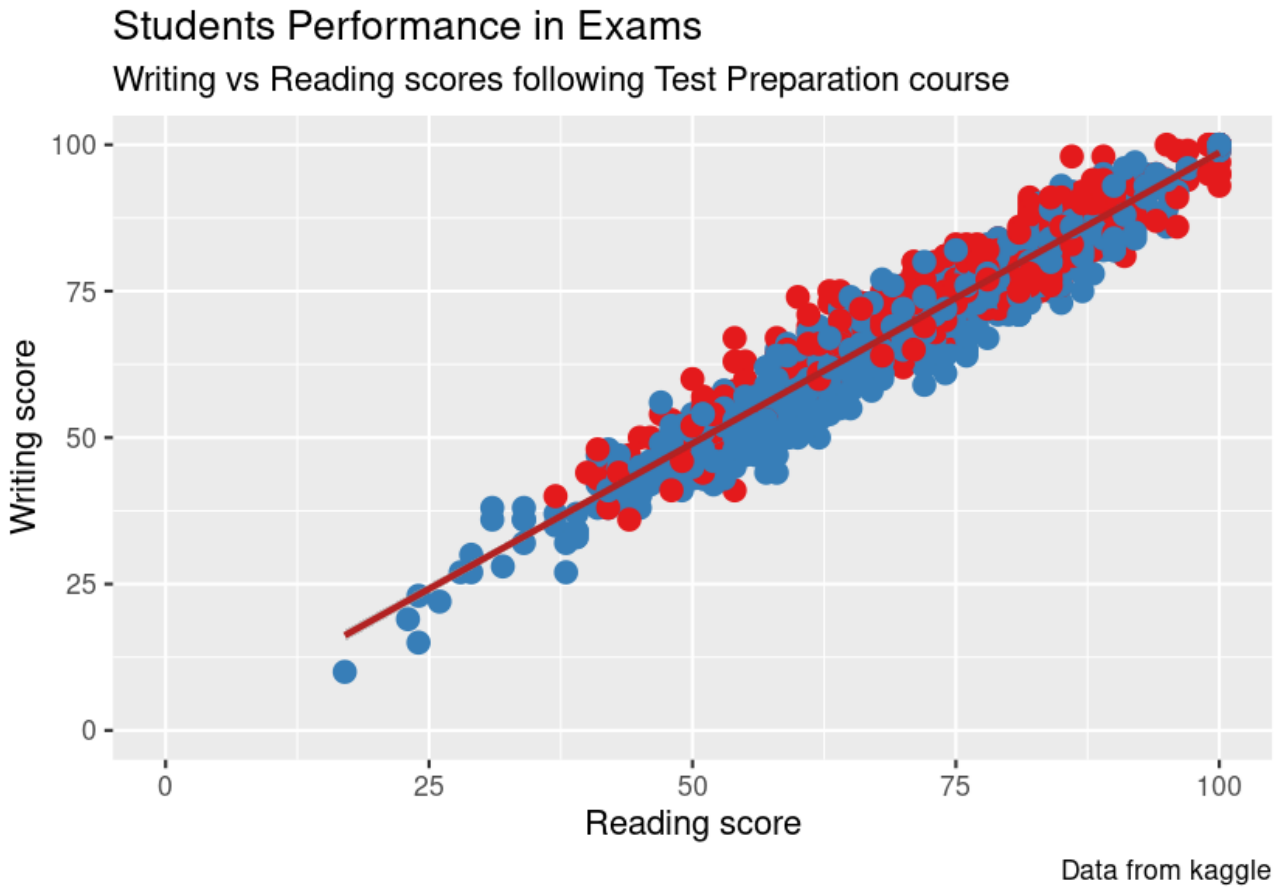



```
ggplot(data, aes(x=reading.score, y=writing.score)) +
# Set color to vary based on state categories
  geom_point(aes(col=test.preparation.course), size=3) +
  geom_smooth(method="lm", col="firebrick") + xlim(c(0,
100)) + ylim(c(0, 100))+
  labs(title="Students Performance in Exams",
       subtitle="Writing vs Reading scores following Test
Preparation",
       y="Writing score",
       x="Reading score",
       caption="Data from kaggle")
```

Part 1:How to Change the Color To Reflect Categories in Another Column?

```
# remove legend
gg + theme(legend.position="None")
```


```
# change color palette
gg + scale_colour_brewer(palette = "Set1")
```



Part 1: Color in RColorBrewer package

```
library(RColorBrewer)
head(brewer.pal.info, 10)
```

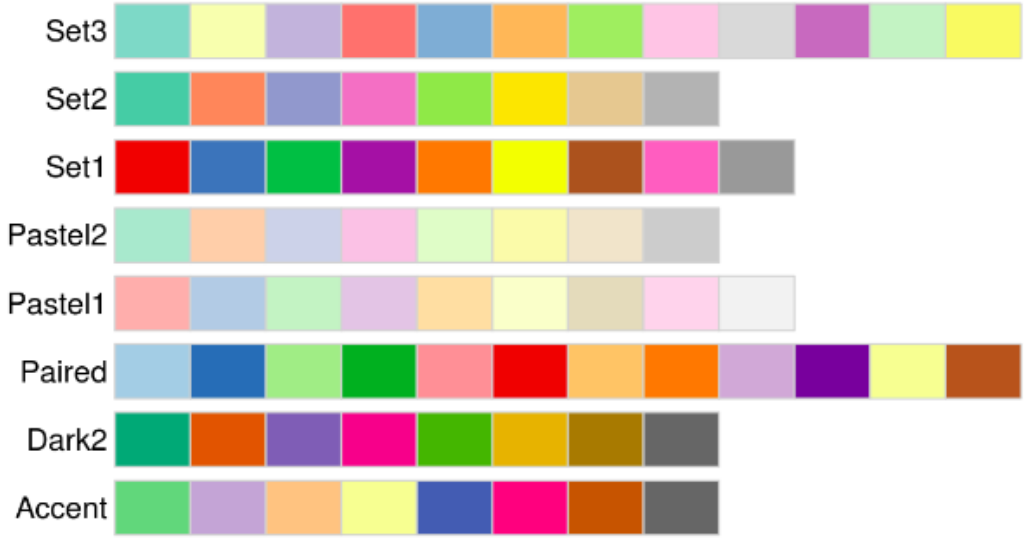
```
#>          maxcolors category colorblind
#> BrBG            11      div       TRUE
#> PiYG            11      div       TRUE
#> PRGn            11      div       TRUE
#> PuOr            11      div       TRUE
#> RdBu            11      div       TRUE
#> RdGy            11      div      FALSE
#> RdYlBu          11      div       TRUE
#> RdYlGn          11      div      FALSE
#> Spectral        11      div      FALSE
#> Accent           8     qual      FALSE
```

There are 3 types of palettes:

Sequential: ordered data

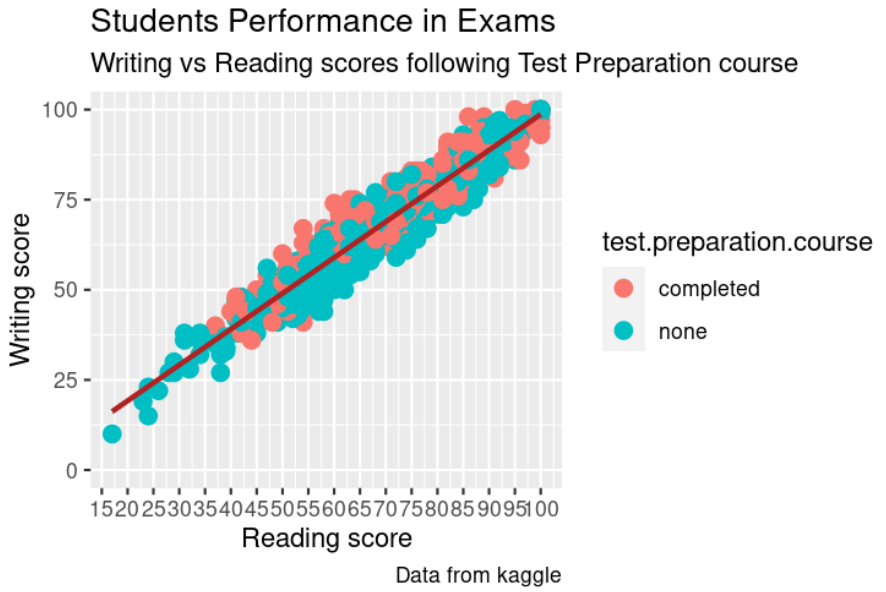
Diverging: equal emphasis on mid-range critical values and extremes at both ends of the data range.

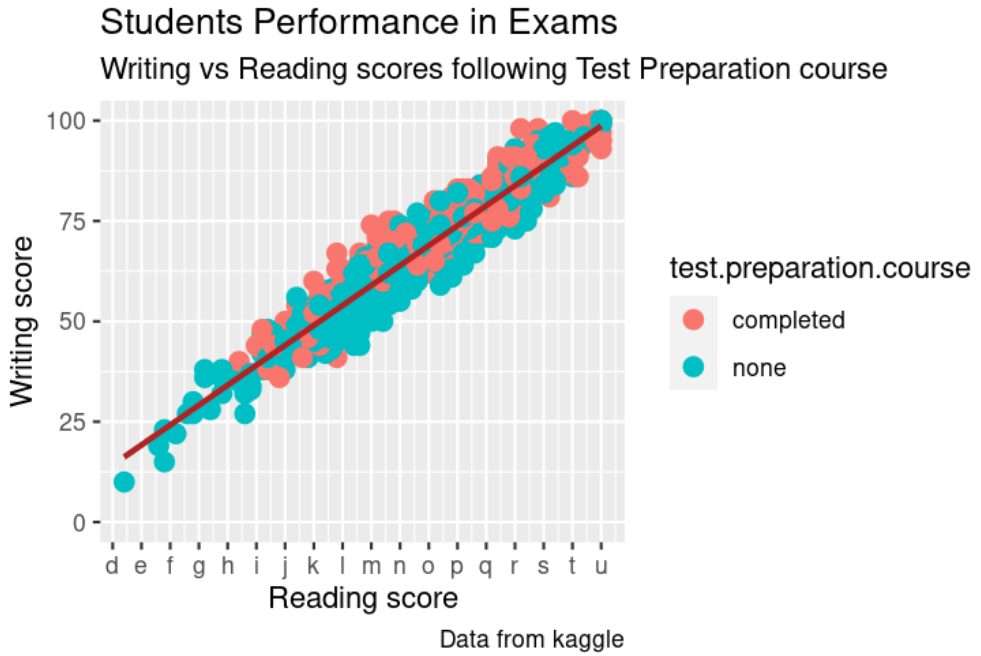
Qualitative: primary visual differences between classes.



Part 1: How to Change the X Axis Texts and Ticks Location

```
gg <- ggplot(data, aes(x=reading.score, y=writing.score)) +
  geom_point(aes(col=test.preparation.course), size=3) +
  geom_smooth(method="lm", col="firebrick") + xlim(c(0, 100)) +
ylim(c(0, 100))+
  labs(title="Students Performance in Exams",
       subtitle="Writing vs Reading scores following Test Preparation
course",
       y="Writing score",
       x="Reading score",
       caption="Data from kaggle")
# Change breaks
gg + scale_x_continuous(breaks=seq(0, 100, 5))
# Change breaks + label
gg + scale_x_continuous(breaks=seq(0, 100, 5), labels = letters[1:21])
```

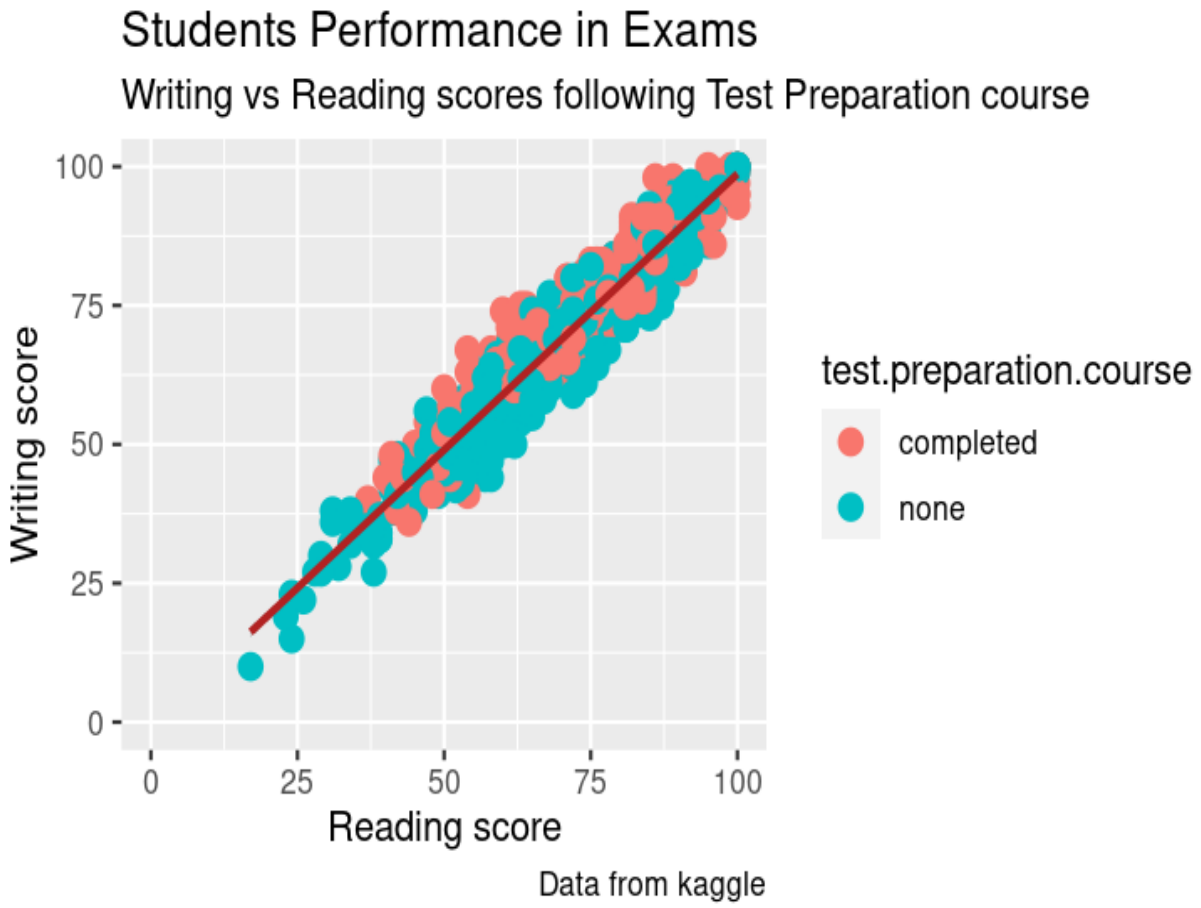


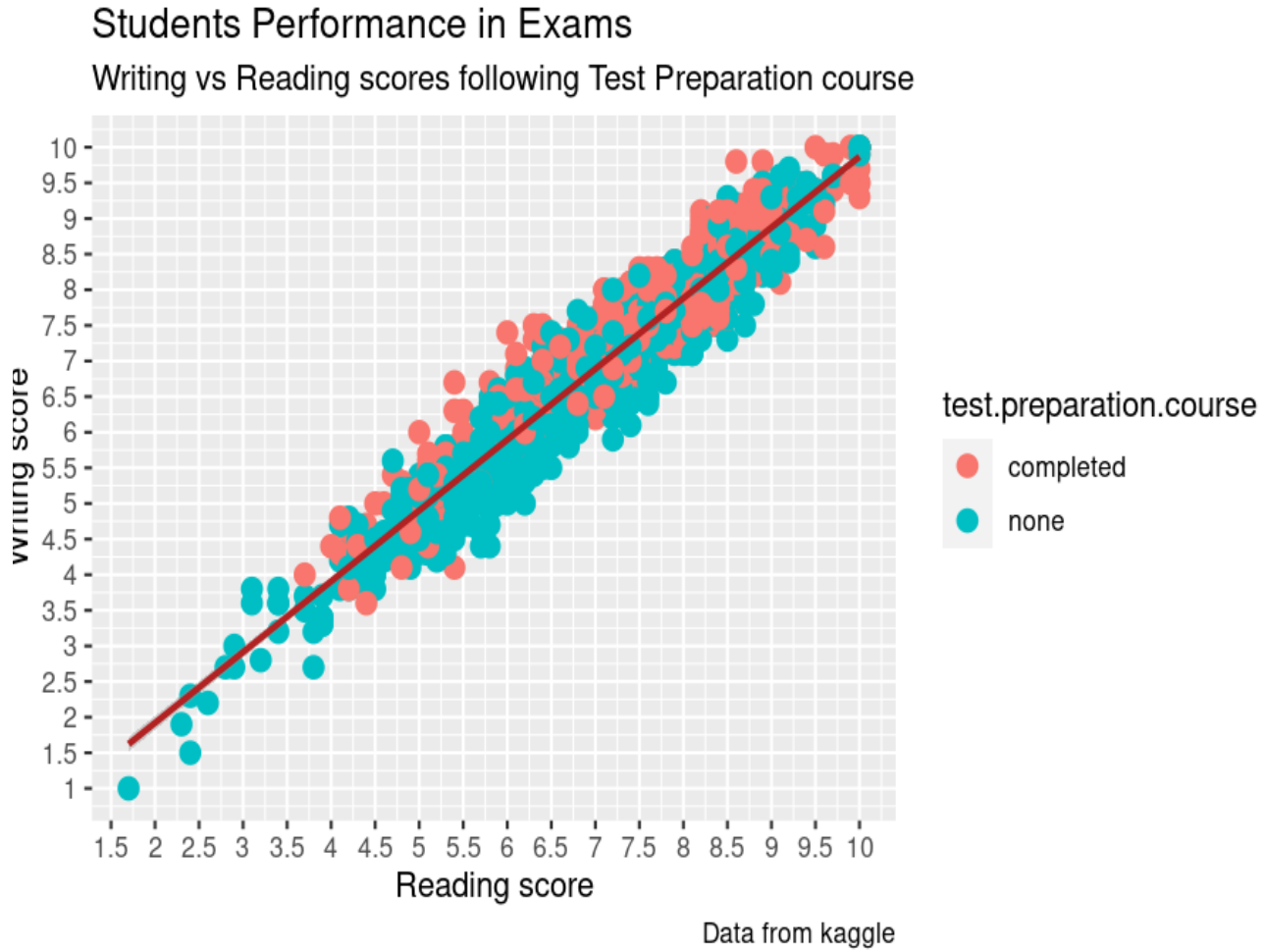


Part 1: How to Write Customized Texts for Axis Labels, by Formatting the Original Values?

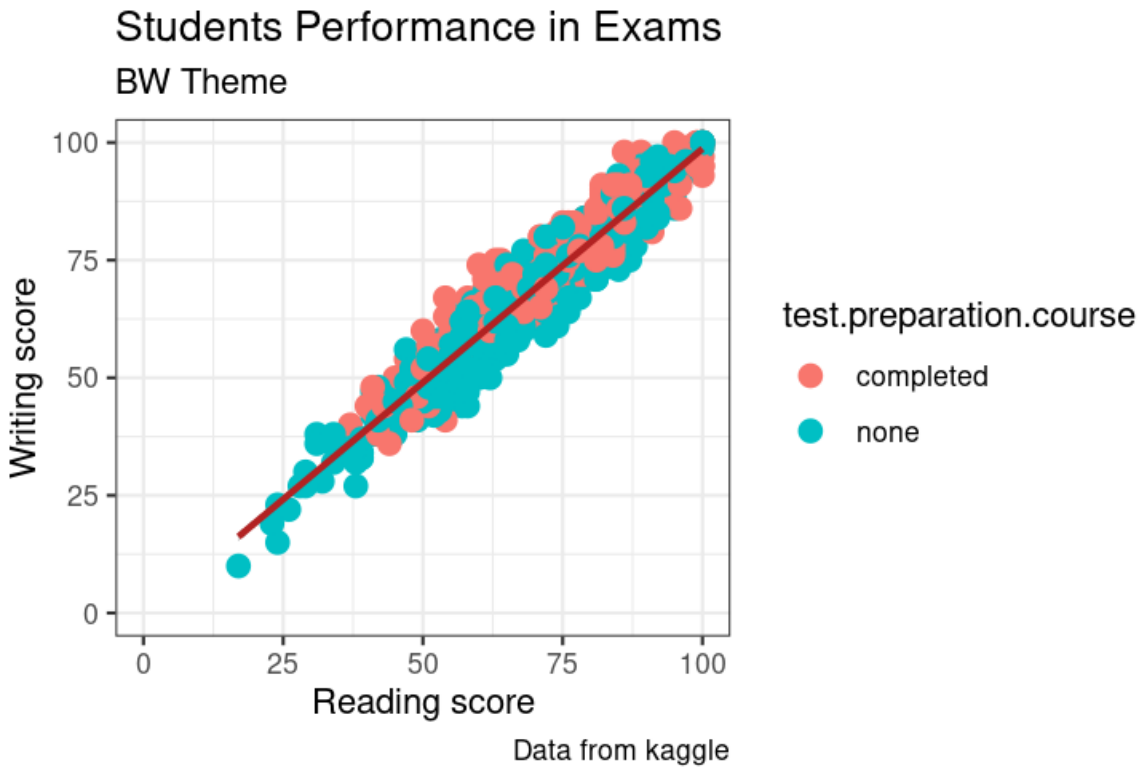
 # Change Axis Texts

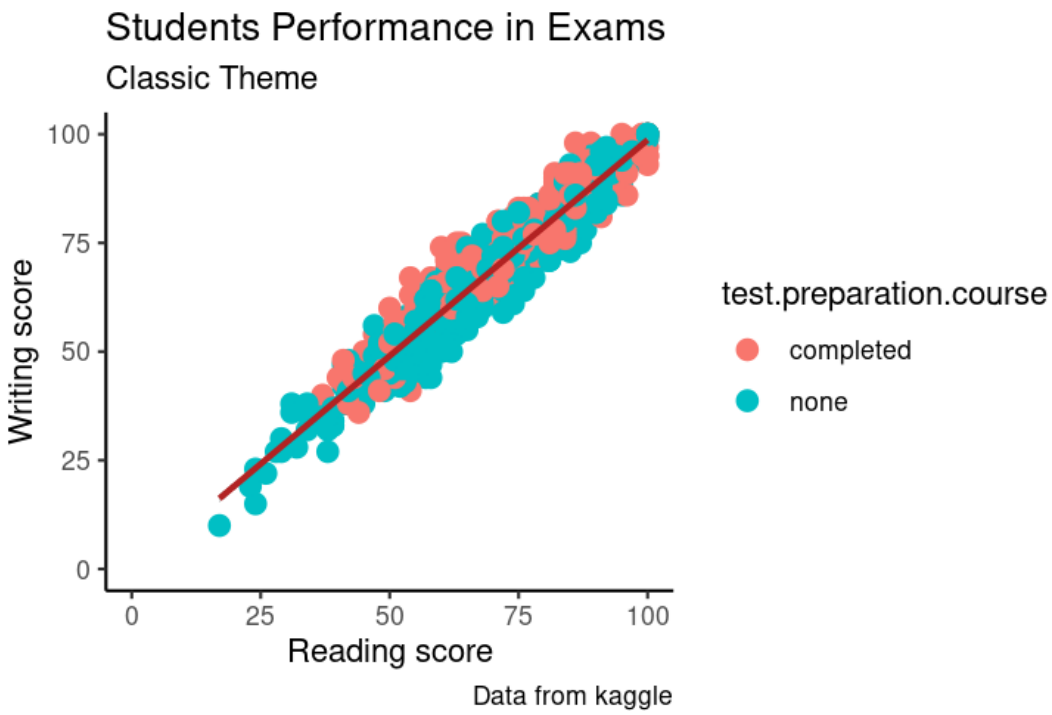
```
gg + scale_x_continuous(breaks=seq(0, 100, 5), labels = function(x){paste0(x/10)}) +
  scale_y_continuous(breaks=seq(0, 100, 5), labels = function(x){paste0(x/10)})
```





Part 1: How to Customize the Entire Theme in One Shot using Pre-Built Themes?





 # method 1: Using theme_set()

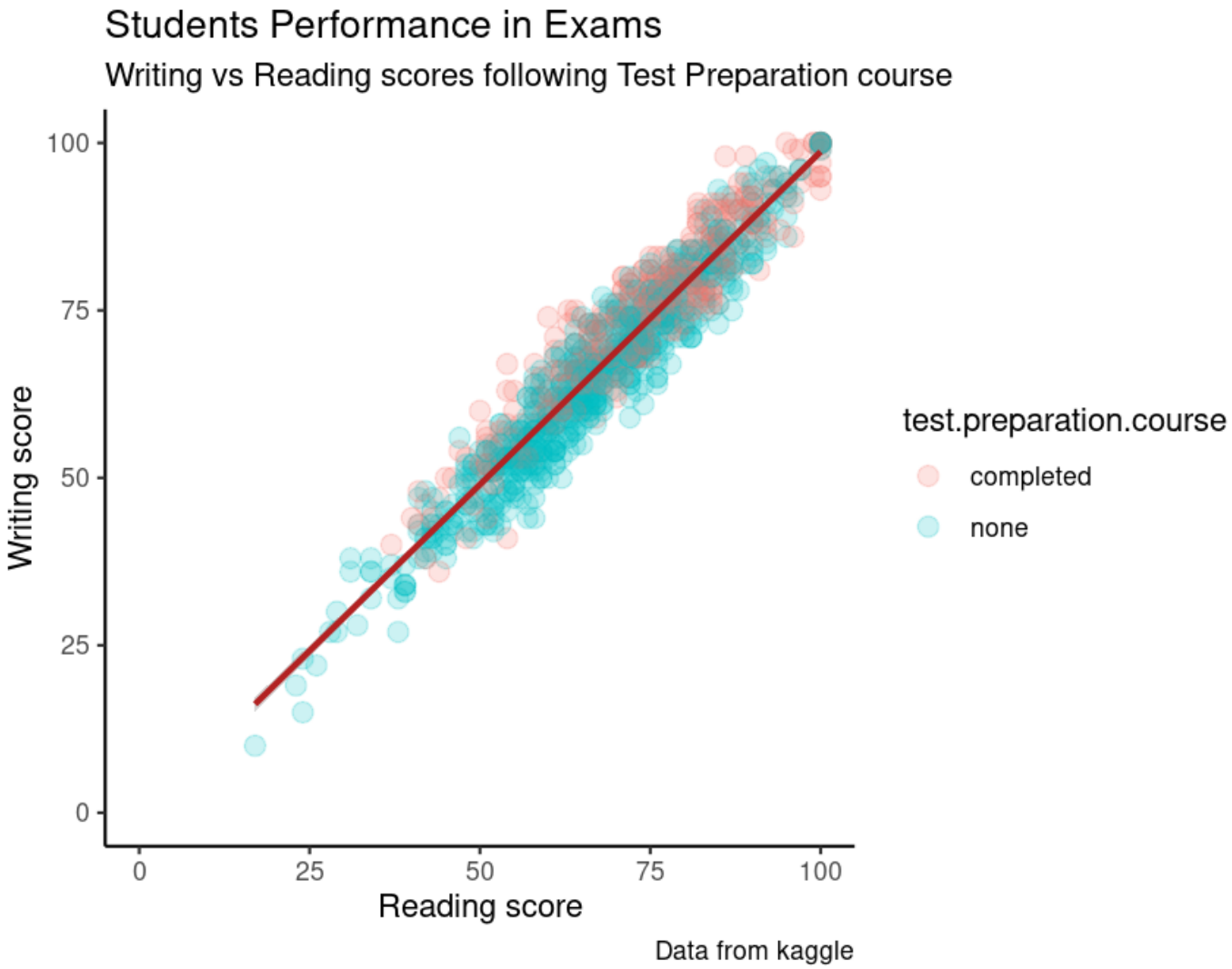
```
gg + theme_set(theme_classic())
```

 # method 2: Adding theme Layer itself.

```
gg + theme_bw() + labs(subtitle="BW Theme")
gg + theme_classic() + labs(subtitle="Classic Theme")
```

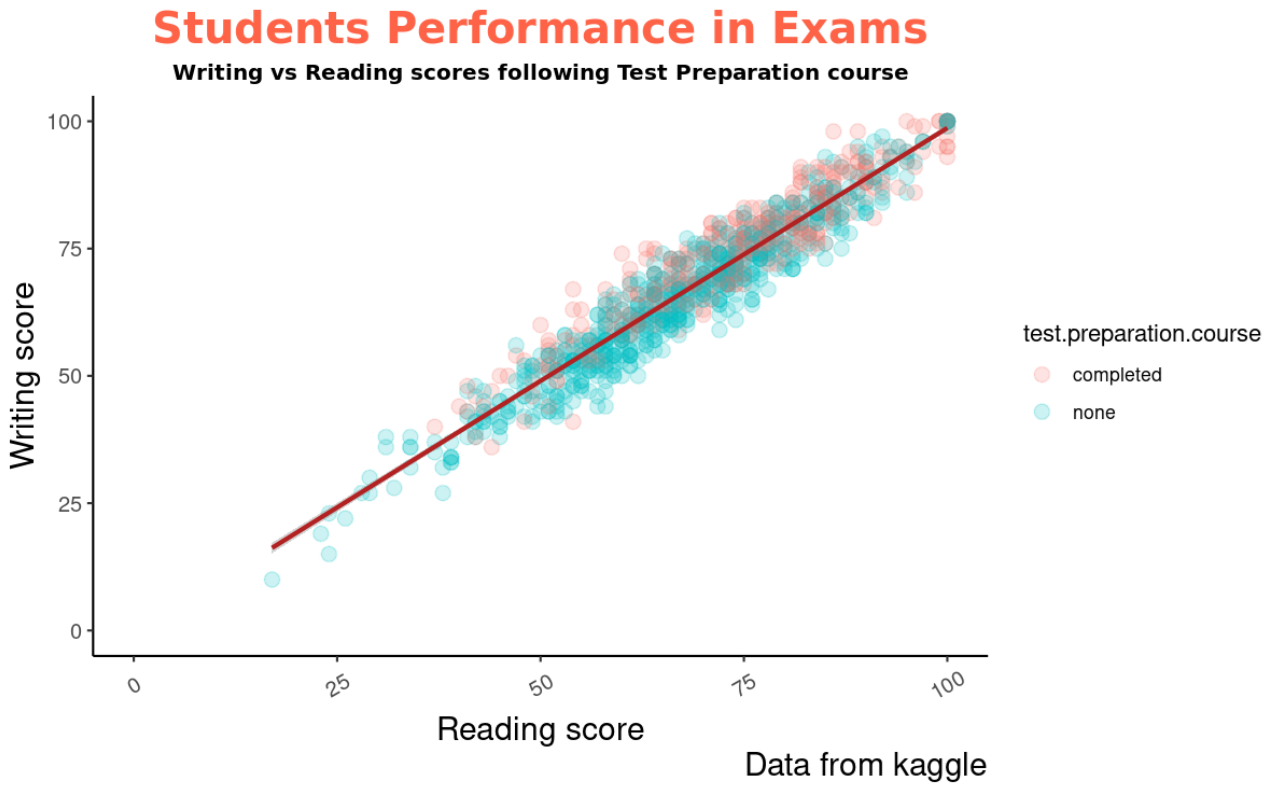
Part 2: How To Customize ggplot2

1. Adding Plot and Axis Titles
2. Modifying Legend
3. Adding Text, Label and Annotation
4. Flipping and Reversing X and Y Axis
5. Faceting: Draw multiple plots within one figure
6. Modifying Plot Background, Major and Minor Axis



2.1 Adding Plot and Axis Titles

```
gg +
# title
theme(plot.title=element_text(size=20,
                              face="bold",
                              family="American Typewriter",
                              color="tomato",
                              hjust=0.5,
                              lineheight=1.2),
# subtitle
 plot.subtitle=element_text(size=15,
                              family="American Typewriter",
                              face="bold",
                              hjust=0.5),
 # caption
 plot.caption=element_text(size=15),
        # X axis title
        axis.title.x=element_text(vjust=0, size=15),
        # Y axis title
        axis.title.y=element_text(size=15),
        # X axis text
        axis.text.x=element_text(size=10,  angle = 30, vjust=.5),
        # Y axis text
        axis.text.y=element_text(size=10))
```



2.2 Modifying Legend

Method 1: Using labs()

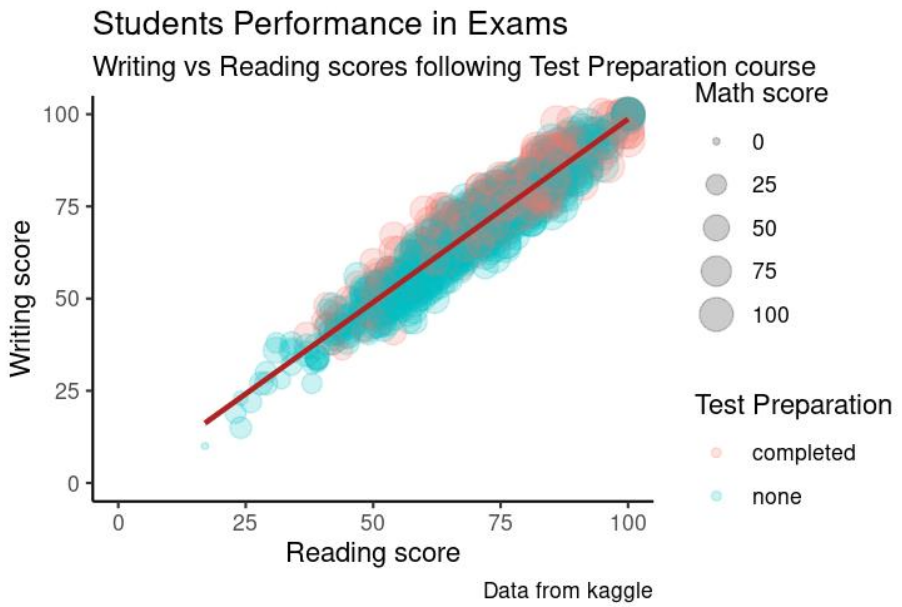
```
gg + labs(color="Test Preparation", size="Math score")
```

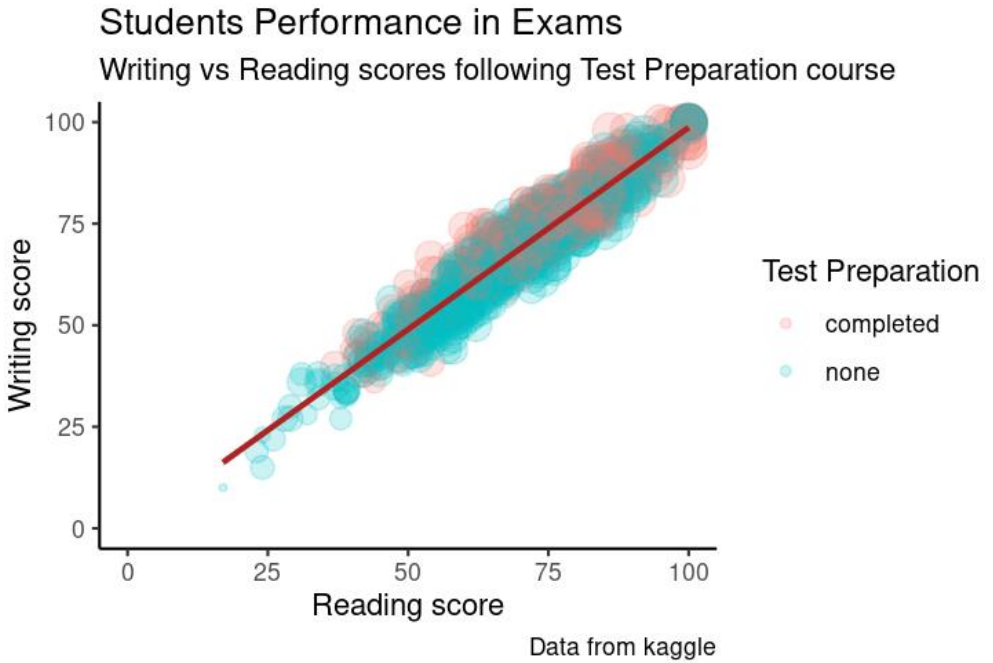
Method 2: Using guides()

```
gg + guides(color=guide_legend("Test Preparation"), size=guide_legend("Math score"))
```

Method 3: Using scale_aesthetic_vartype() format

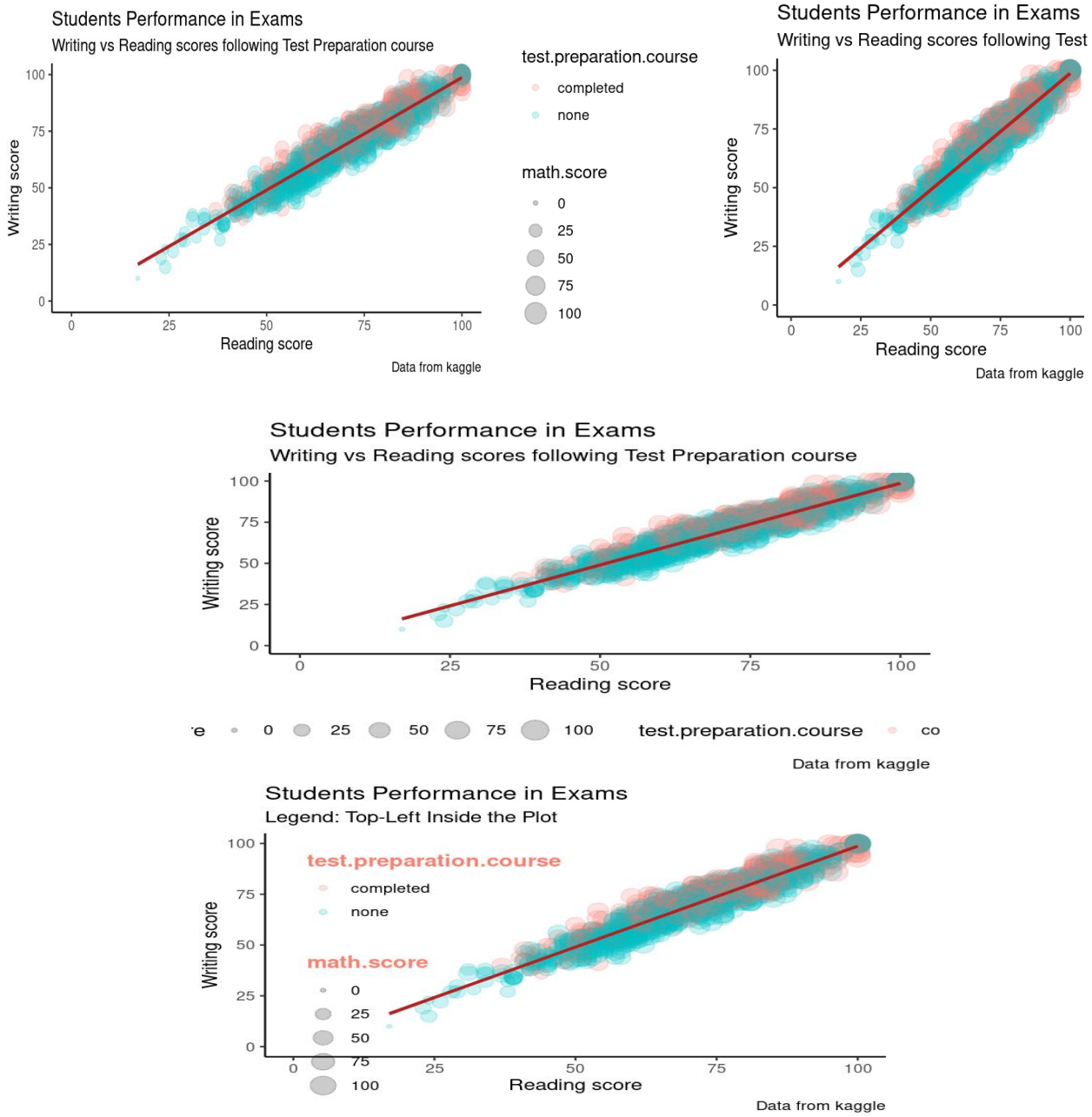
```
gg + scale_color_discrete(name="Test Preparation") + scale_size_continuous(name = "Math score", guide = FALSE)
```





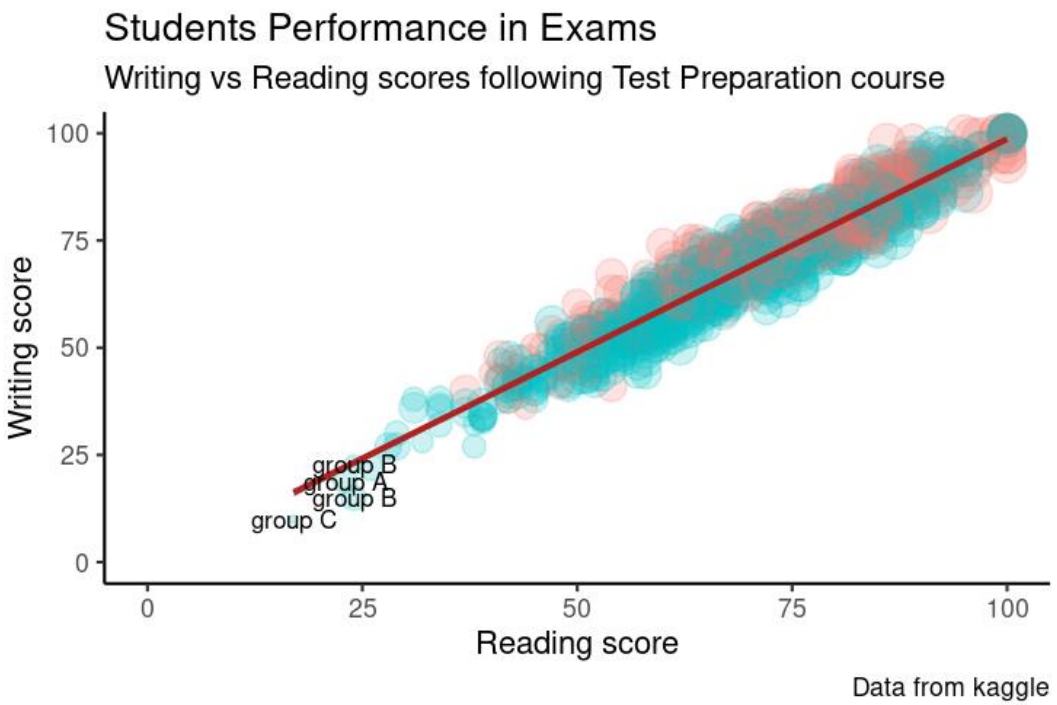
2.2.1 How to Remove the Legend and Change Legend Positions

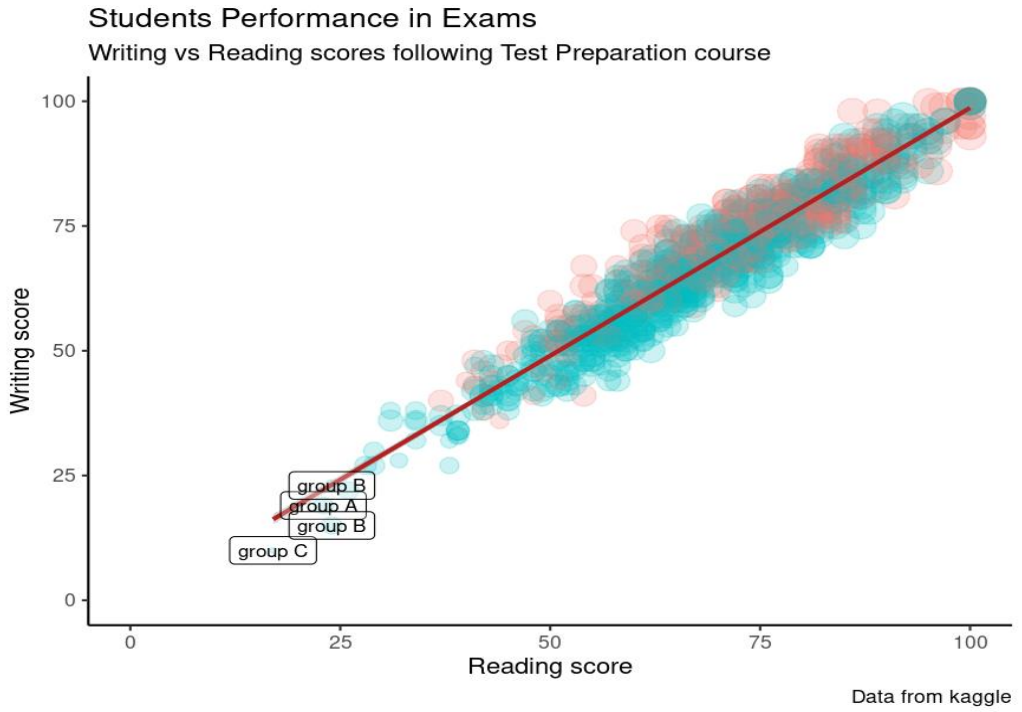
```
# No legend ----------------------------------------------
gg + theme(legend.position="None")

# Legend to the left -------------------------------------
gg + theme(legend.position="left")

# legend at the bottom and horizontal ---------------------
gg + theme(legend.position="bottom", legend.box = "horizontal")

# legend at bottom-right, inside the plot ----------------
gg + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
         legend.justification=c(1,0),
         legend.position=c(0.95, 0.05),
         legend.background = element_blank(),
         legend.key = element_blank()) +

# legend at top-left, inside the plot ----------------------
gg + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
         legend.justification=c(0,1),
         legend.position=c(0.05, 0.95),
         legend.background = element_blank(),
         legend.key = element_blank()) +
  labs(subtitle="Legend: Top-Left Inside the Plot")
```



2.3. Adding Text, Label and Annotation

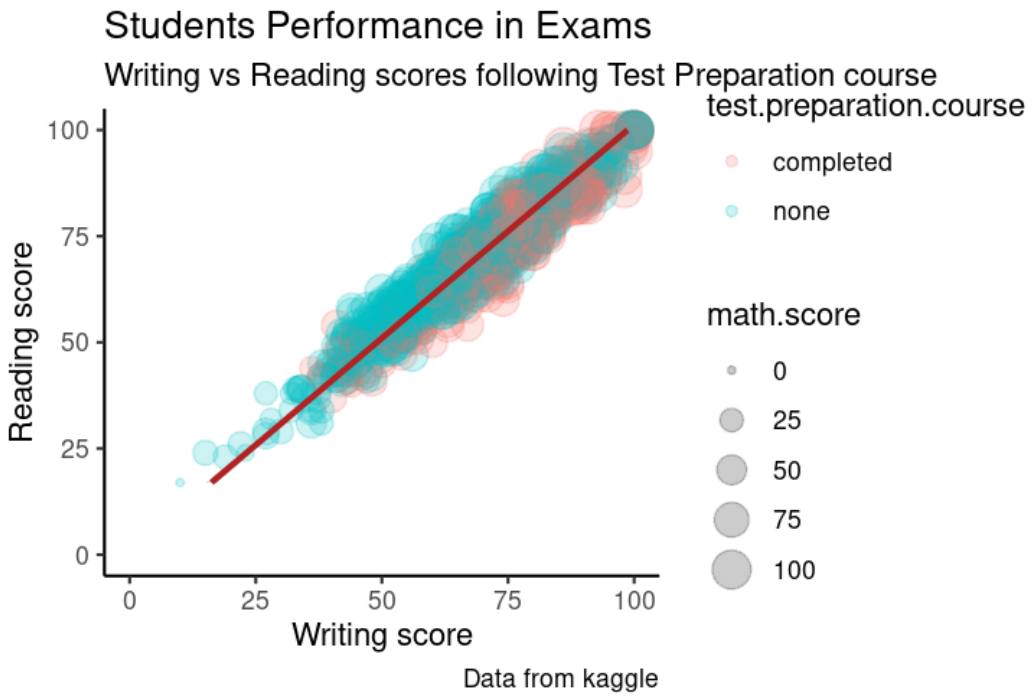
```
# fill data with score
subdata <- subset(data, reading.score<25 & writing.score<25)


# add label
gg + geom_text(aes(label=race.ethnicity), size=3, data=subdata) + theme(legend.position = "None")
gg + geom_label(aes(label=race.ethnicity), size=3, data=subdata, alpha=0.25) + theme(legend.position = "None")  # label
```



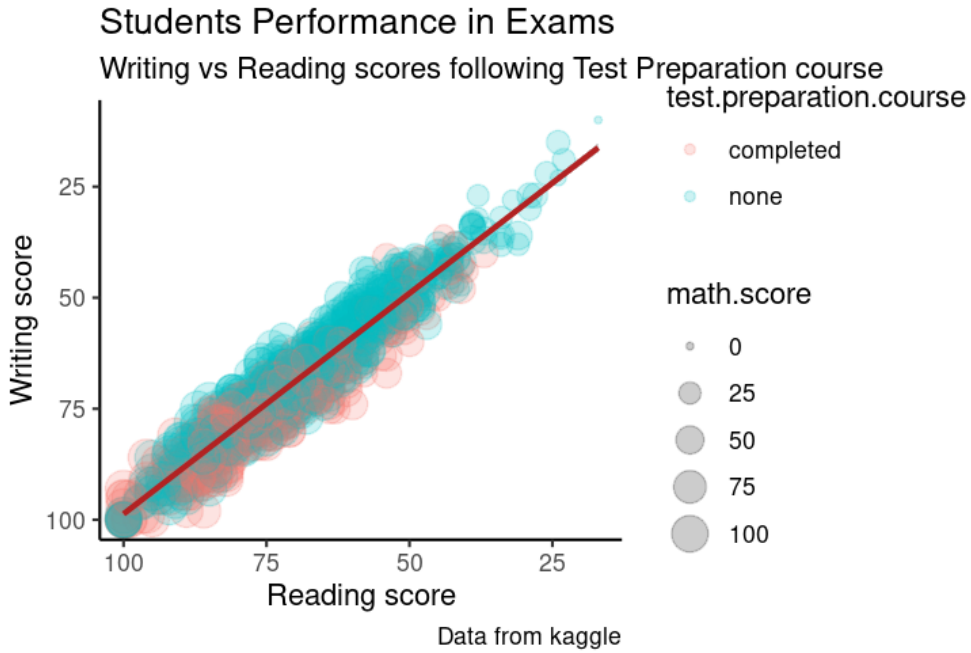


2.4. Flipping and Reversing X and Y Axis

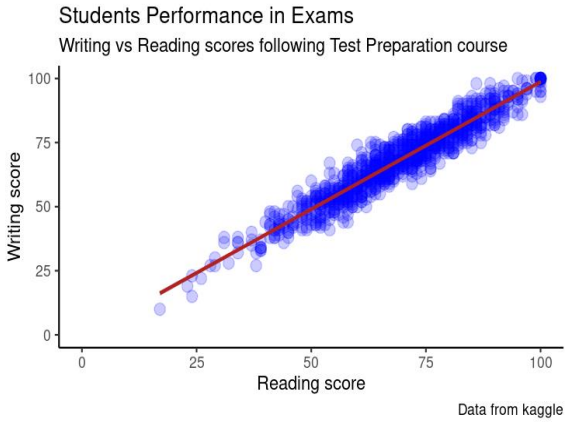
```
# Flip the X and Y axis
gg + coord_flip()
```



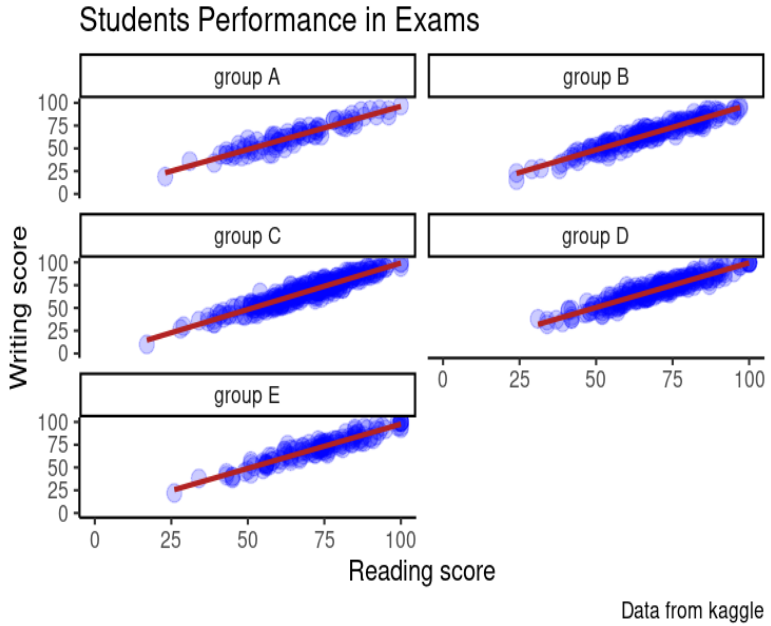

```
# Reverse the X and Y Axis
gg + scale_x_reverse() + scale_y_reverse()
```



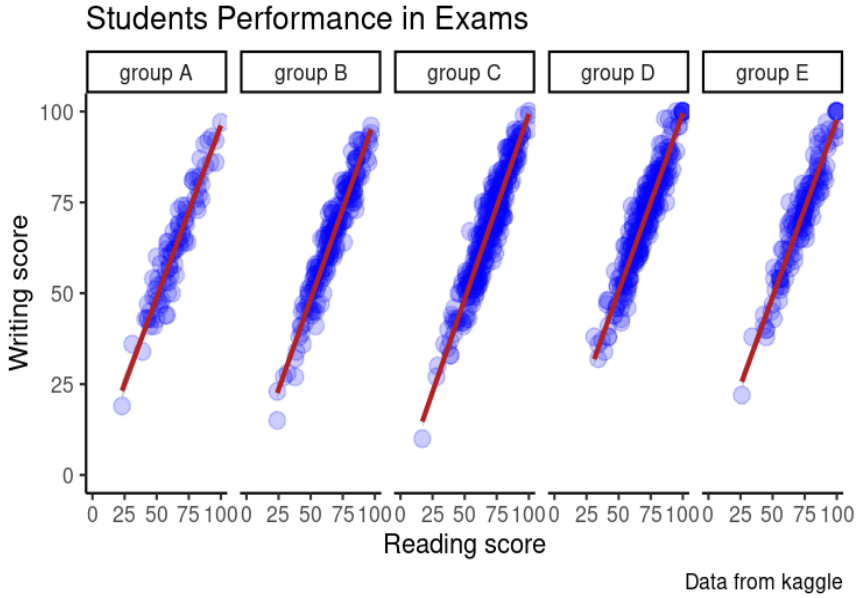
2.5. Faceting: Draw multiple plots within one figure



Facet Wrap



Facet Grid

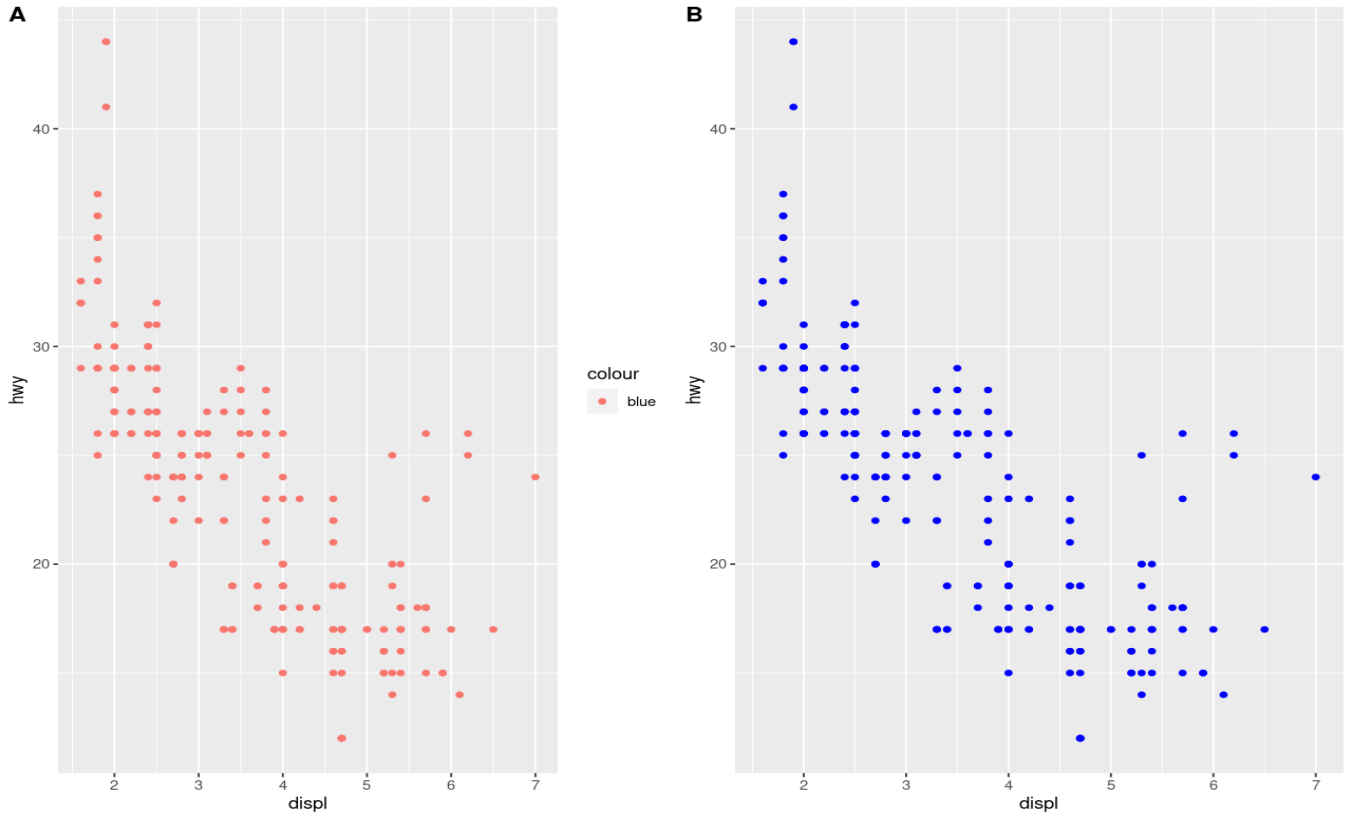



```
# Facet wrap with common scales

gg + facet_wrap( ~ race.ethnicity, nrow=3)
```

```
# Add Facet Grid

gg + facet_grid( ~ race.ethnicity)
```

2.5. Faceting: Draw multiple plots within one figure

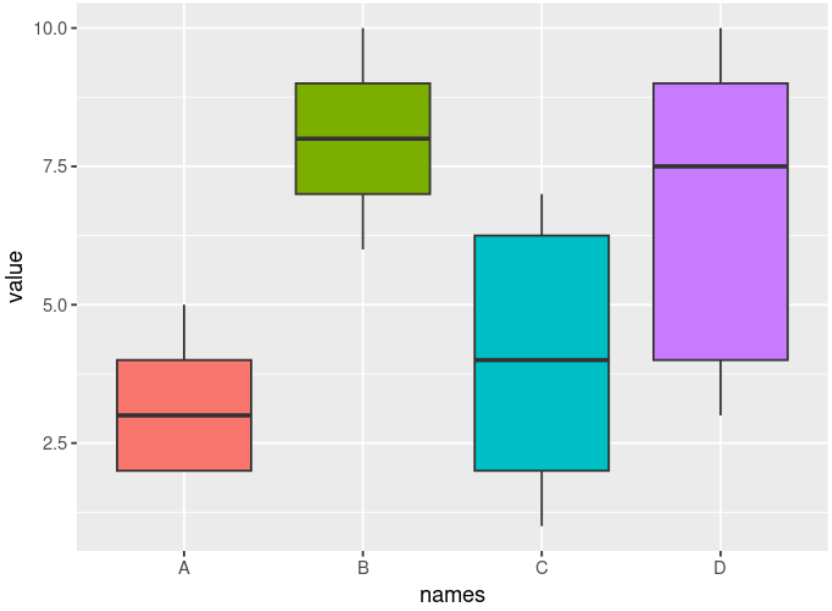
```
library("ggpubr")
a <- ggplot(mpg, aes(displ, hwy)) + geom_point(aes(colour = "blue"))
b <- ggplot(mpg, aes(displ, hwy)) + geom_point(colour = "blue")
ggarrange(a, b, labels = c("A", "B"), ncol = 2, nrow = 1)
```



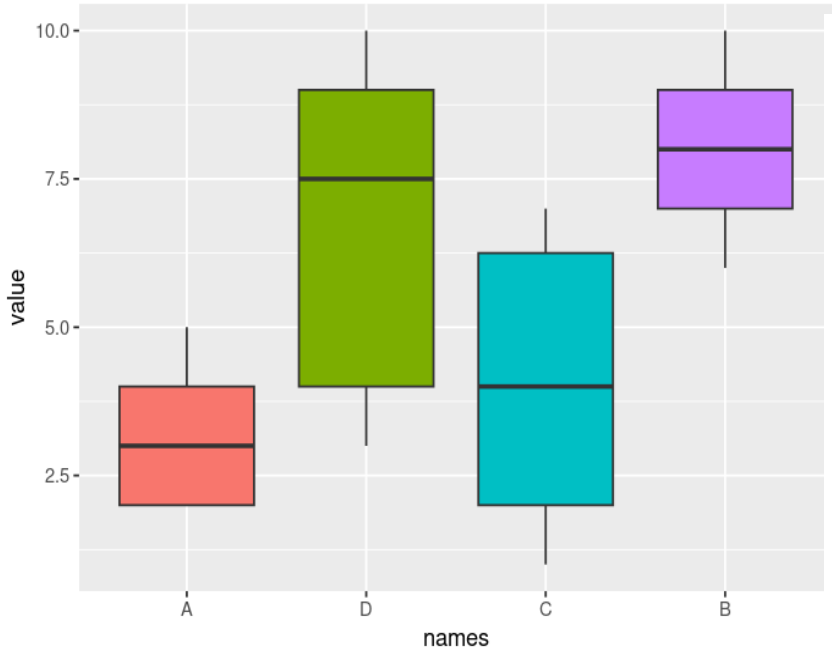
http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/

2.6 Change the Order

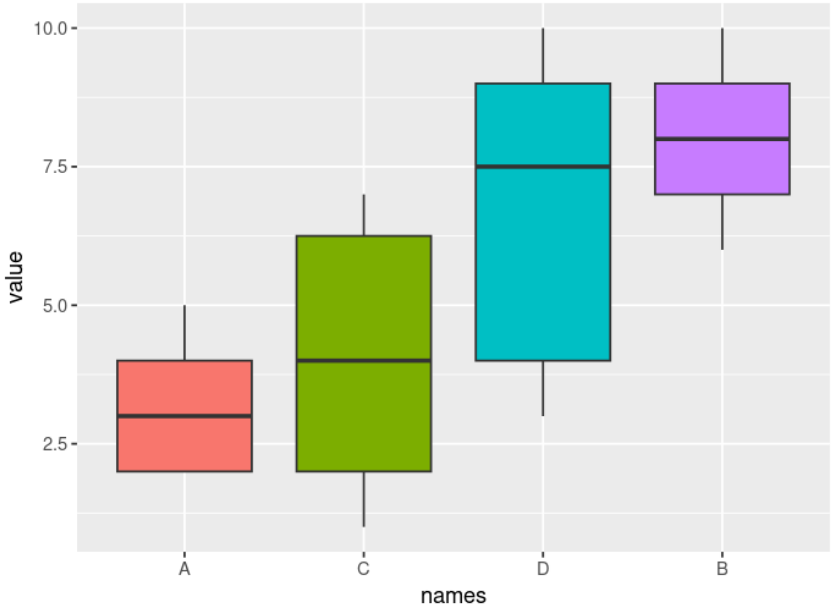
```
#Creating data
names <- c(rep("A", 20) , rep("B", 20) , rep("C", 20), rep("D", 20))
value <- c( sample(2:5, 20 , replace=T) , sample(6:10, 20 , replace=T),
            sample(1:7, 20 , replace=T), sample(3:10, 20 , replace=T) )
data <- data.frame(names,value)
ggplot(data, aes(x=names, y=value, fill=names)) + geom_boxplot() + theme(legend.position="NONE")

# I reorder the groups order : I change the order of the factor data$names
data$names <- factor(data$names , levels=c("A", "D", "C", "B"))
ggplot(data, aes(x=names, y=value, fill=names)) + geom_boxplot() + theme(legend.position="NONE")

# reoder by mean
data$names <- with(data, reorder(names , value, median , na.rm=T))
ggplot(data, aes(x=names, y=value, fill=names)) + geom_boxplot() + theme(legend.position="NONE")
```



original



order by desire



order by median

2.7 Save the plot

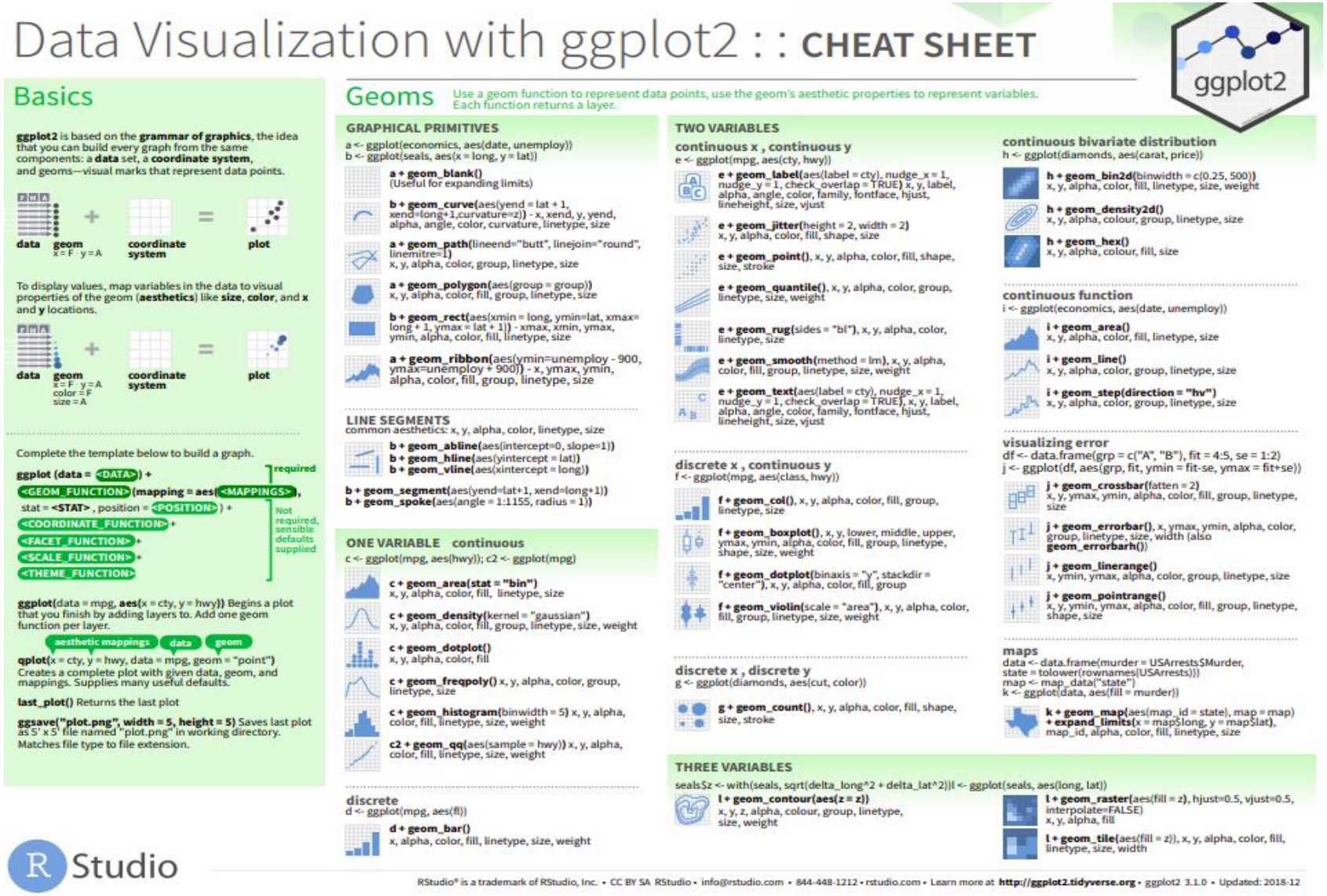
Method 1

```
## 1. Open a pdf file, can add the path
pdf("medthod1.pdf")
## 2. Create a plot
ggplot(data, aes(x=names, y=value, fill=names)) + geom_boxplot() + theme(legend.position="NONE")
## 3. Close the pdf file
dev.off()
```

Method 2

```
#  ggsave
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
ggsave("method2.pdf", width = 6,   height = 4)
```

Cheat Sheet

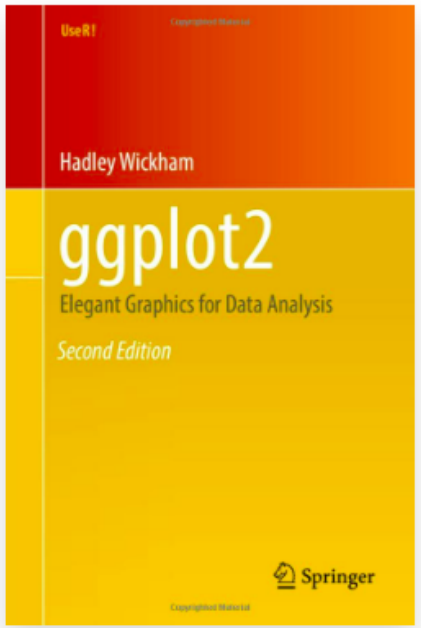


https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf

Book

Welcome

This is the on-line version of work-in-progress **3rd edition** of "ggplot2: elegant graphics for data analysis" published by Springer. You can learn what's changed from the 2nd edition in the Preface.



While this book gives some details on the basics of ggplot2, its primary focus is explaining the Grammar of Graphics that ggplot2 uses, and describing the full details. It is not a cookbook, and won't necessarily help you create any specific graphic that you need. But it will help you understand the details of the underlying theory, giving you the power to tailor any plot specifically to your needs.

The book is written by Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen.

Preface to the third edition »

Link: https://ggplot2-book.org/index.html

Part 3: Fread function in data.table packages

Similar to read.table but **faster and more convenient**. All controls such as **sep, colClasses and nrows are automatically detected.**

fread is for regular delimited files; i.e., where every row has the same number of columns.

```
#install.packages(data.table)
library(data.table)
traindata1 <- fread('https://raw.githubusercontent.com/EFPTTT-THYROID/Shiniapp/main/traindata_model.tsv')
traindata2 <- read.table('https://raw.githubusercontent.com/EFPTTT-THYROID/Shiniapp/main/traindata_model.tsv',
header=T)
```

https://www.rdocumentation.org/packages/data.table/versions/1.14.8/topics/fread

Part 3: Reshape2

```
#install.packaeges(reshpae2)
library(reshape2)
w_data <- data.frame(team=c('A', 'B', 'C', 'D'),
                     points=c(88, 91, 99, 94),
                     assists=c(12, 17, 24, 28),
                     rebounds=c(22, 28, 30, 31))
long_df <- melt(w_data, id='team')
View(long_df)
```

Wide Format

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Long Format

Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

https://www.statology.org/melt-in-r/

Part 3: do.call and Reduce

‘**Reduce’** uses a binary function to successively combine the elements of a given vector and a possibly given initial value.
‘**do.call’** constructs and executes a function call from a name or a function and a list of arguments to be passed to it.

```
[[1]]
  cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1       0.35        0.5        0.5       0.39       0.15       0.47       0.33       0.08       0.44
  cg15639951 cg02192520 cg08675585 cg07447773 subtype
1       0.25       0.67       0.53       0.27   NIFTP

[[2]]
  cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1       0.82       0.83       0.83       0.72       0.16       0.28       0.19       0.63       0.79
  cg15639951 cg02192520 cg08675585 cg07447773 subtype
1       0.66       0.85       0.87       0.76      FA

[[3]]
  cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1       0.62       0.81       0.82       0.72       0.14       0.18       0.21       0.17       0.47
  cg15639951 cg02192520 cg08675585 cg07447773 subtype
1       0.34       0.32       0.86       0.49      FA
```

```
# do.call
data_1 <- do.call("rbind", data_f )

# reduce
data_2<- Reduce(function(x, y) rbind(x,y), data_f)
```

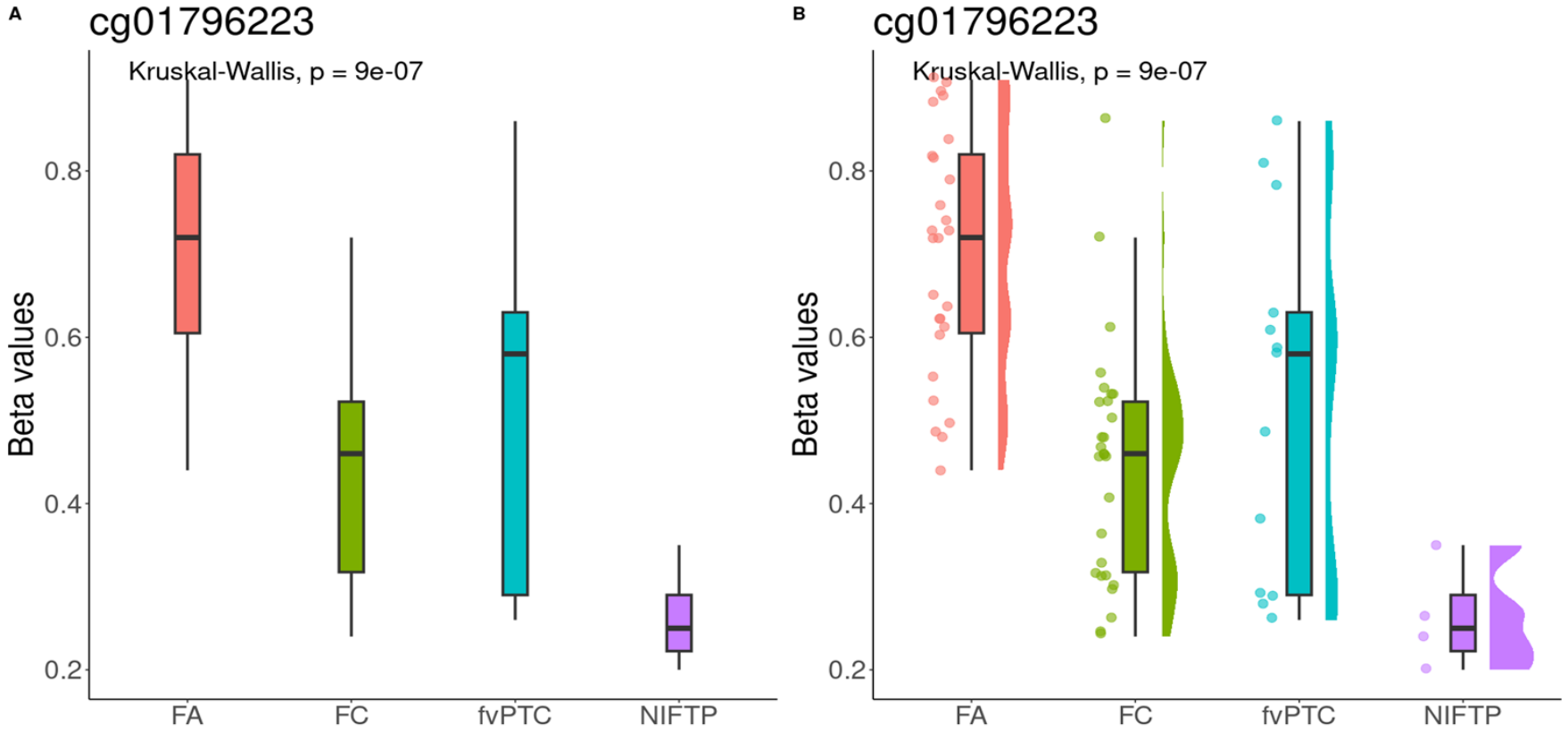
Part 3: Result of data

beta-value of 13 CpGs of 4 subtypes

cg01796223	cg01091565	cg04705866	cg11484872	cg26607785	cg03409548	cg06454226	cg03001305	cg19282250	cg15639951	cg02192520	cg08675585	cg07447773	subtype
0.35	0.5	0.5	0.39	0.15	0.47	0.33	0.08	0.44	0.25	0.67	0.53	0.27	NIFTP
0.23	0.44	0.6	0.44	0.1	0.46	0.16	0.13	0.26	0.29	0.2	0.48	0.4	NIFTP
0.27	0.59	0.55	0.36	0.1	0.4	0.19	0.15	0.71	0.35	0.51	0.61	0.48	NIFTP
0.2	0.58	0.71	0.49	0.15	0.35	0.33	0.1	0.63	0.22	0.79	0.66	0.55	NIFTP
0.73	0.82	0.83	0.75	0.2	0.17	0.25	0.21	0.51	0.42	0.31	0.84	0.5	FA
0.61	0.87	0.86	0.72	0.16	0.17	0.19	0.64	0.31	0.67	0.82	0.85	0.7	FA
0.64	0.85	0.84	0.75	0.14	0.18	0.46	0.25	0.41	0.38	0.71	0.83	0.34	FA
0.62	0.83	0.78	0.73	0.16	0.18	0.39	0.58	0.65	0.62	0.86	0.88	0.77	FA
0.82	0.87	0.87	0.72	0.19	0.19	0.32	0.65	0.52	0.62	0.73	0.89	0.53	FA
0.82	0.83	0.83	0.72	0.16	0.28	0.19	0.63	0.79	0.66	0.85	0.87	0.76	FA
0.62	0.81	0.82	0.72	0.14	0.18	0.21	0.17	0.47	0.34	0.32	0.86	0.49	FA
0.53	0.79	0.85	0.66	0.44	0.16	0.33	0.13	0.74	0.41	0.6	0.85	0.51	FC
0.86	0.87	0.83	0.69	0.38	0.25	0.17	0.63	0.54	0.64	0.82	0.8	0.74	FC
0.32	0.82	0.83	0.65	0.37	0.18	0.51	0.44	0.73	0.62	0.75	0.79	0.67	FC
0.3	0.83	0.86	0.74	0.47	0.21	0.25	0.52	0.76	0.44	0.77	0.83	0.32	FC
0.52	0.85	0.86	0.75	0.41	0.15	0.44	0.21	0.69	0.38	0.79	0.85	0.55	FC
0.3	0.83	0.83	0.68	0.17	0.13	0.73	0.15	0.76	0.49	0.7	0.82	0.79	FC

How to visualize the beta-value of each CpGs following the subtypes?

The results

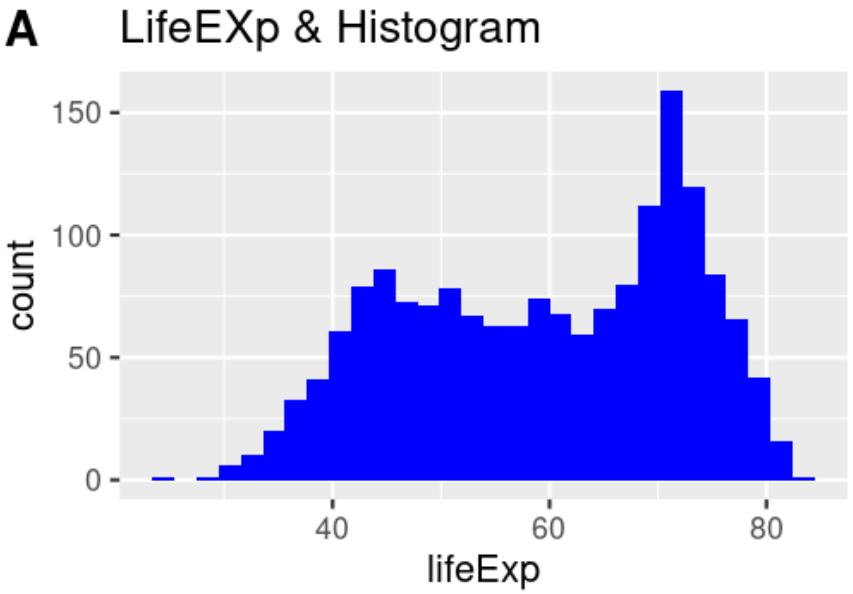


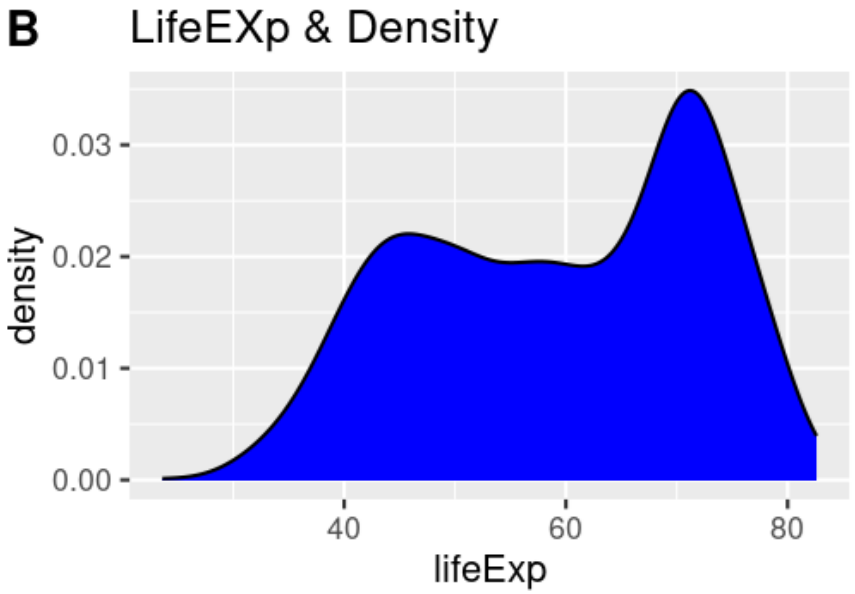
http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

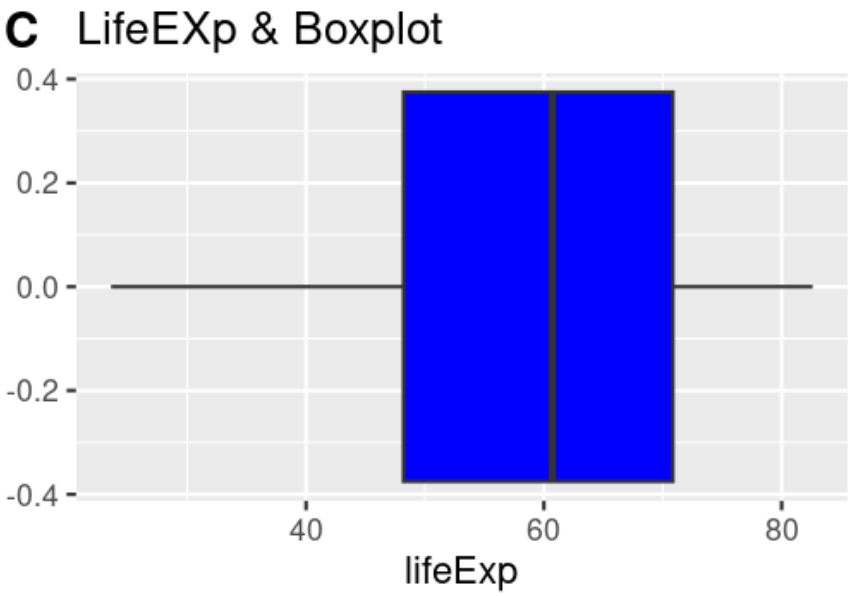
Some examples

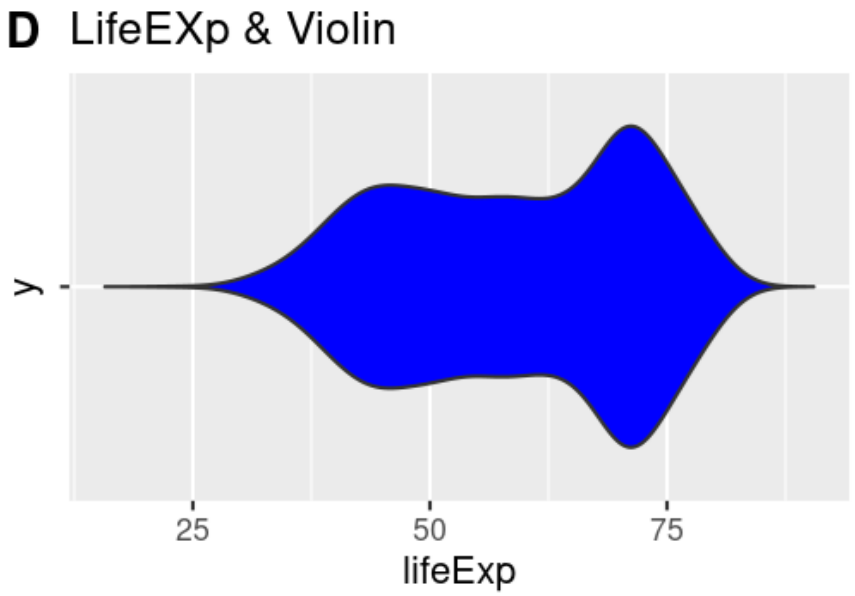
ONE VARIABLE - continuous

```
> head(data[,c(1,4)])
      country lifeExp
1 Afghanistan  28.801
2 Afghanistan  30.332
3 Afghanistan  31.997
4 Afghanistan  34.020
5 Afghanistan  36.088
6 Afghanistan  38.438
```





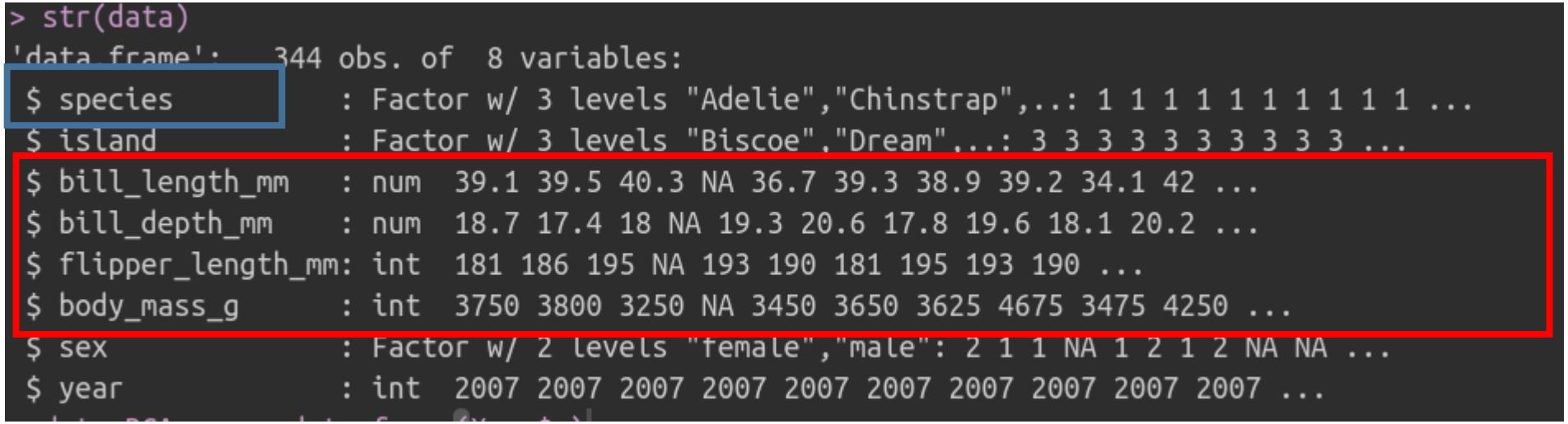




Part 4: Principal component analysis (PCA)

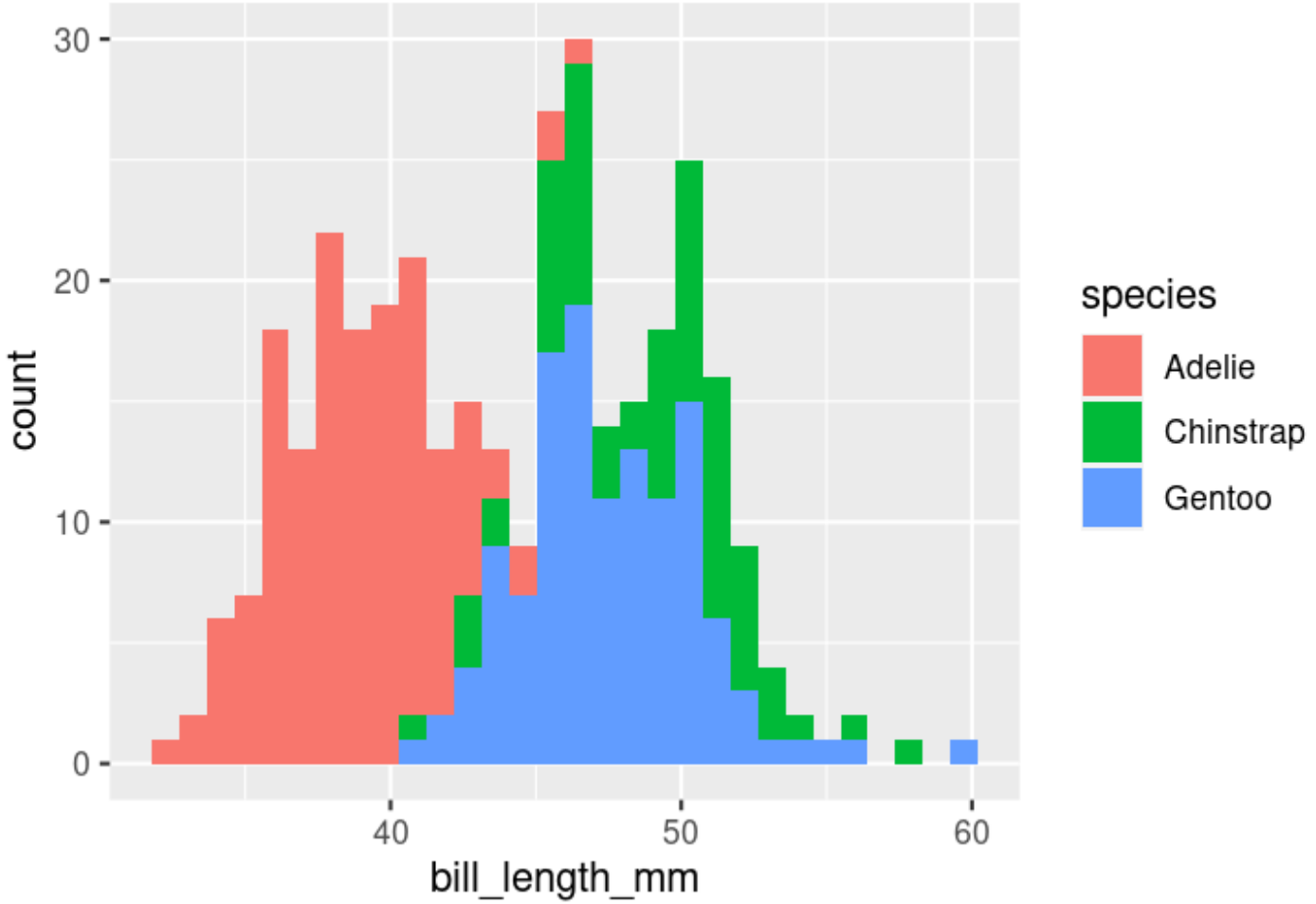
What is PCA ?

PCA is **a statistical approach** that can be used to **analyze high-dimensional data** and **capture the most important information from it**. This is done by **transforming the original data into a lower-dimensional space** while collating highly correlated variables together.



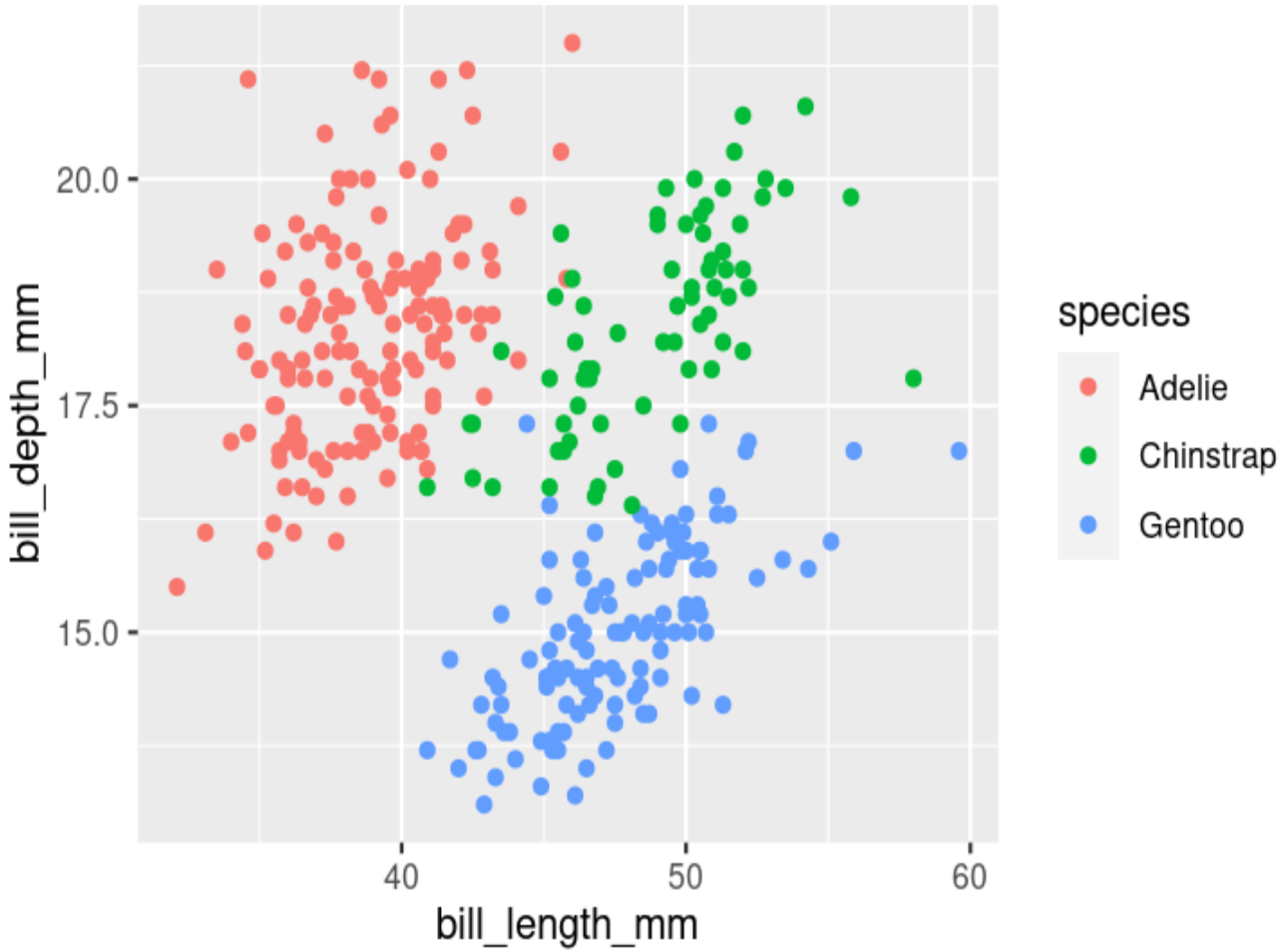
Part 4: Data with 1 variable

```
> head(data_try)
  species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1  Adelie           39.1          18.7               181        3750
2  Adelie           39.5          17.4               186        3800
3  Adelie           40.3          18.0               195        3250
5  Adelie           36.7          19.3               193        3450
6  Adelie           39.3          20.6               190        3650
7  Adelie           38.9          17.8               181        3625
```



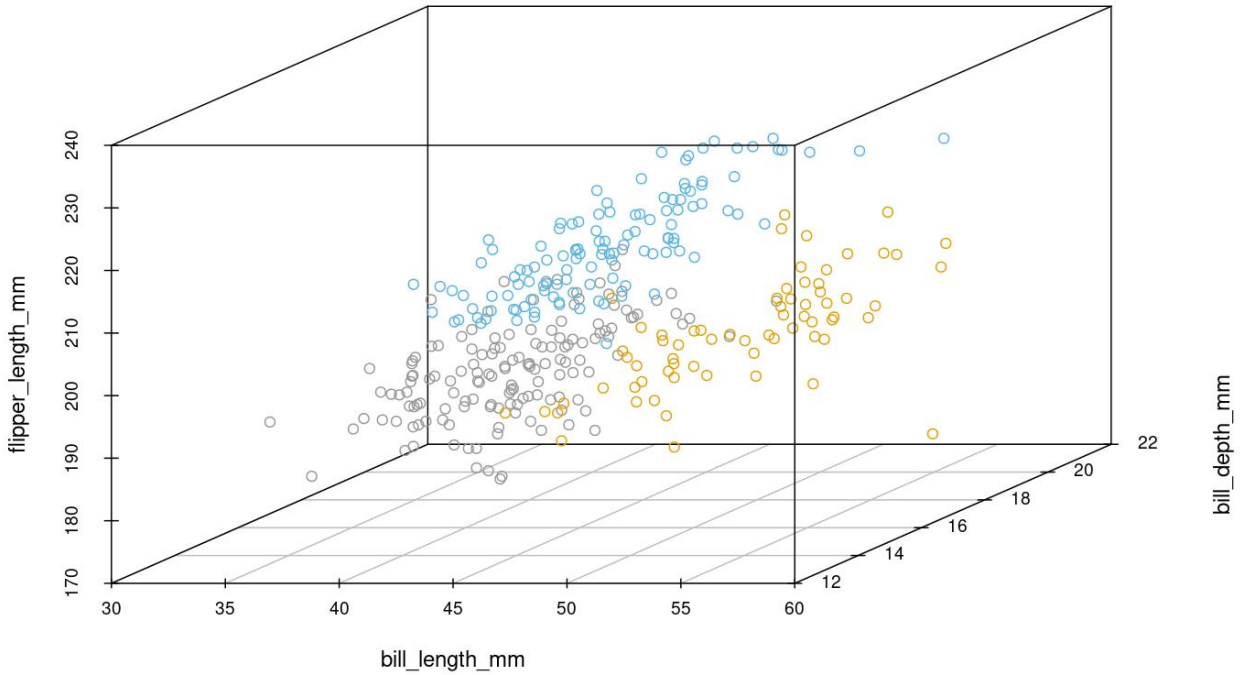
Part 4: Data with 2 variables

```
> head(data_try)
  species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1  Adelie           39.1          18.7               181        3750
2  Adelie           39.5          17.4               186        3800
3  Adelie           40.3          18.0               195        3250
5  Adelie           36.7          19.3               193        3450
6  Adelie           39.3          20.6               190        3650
7  Adelie           38.9          17.8               181        3625
```



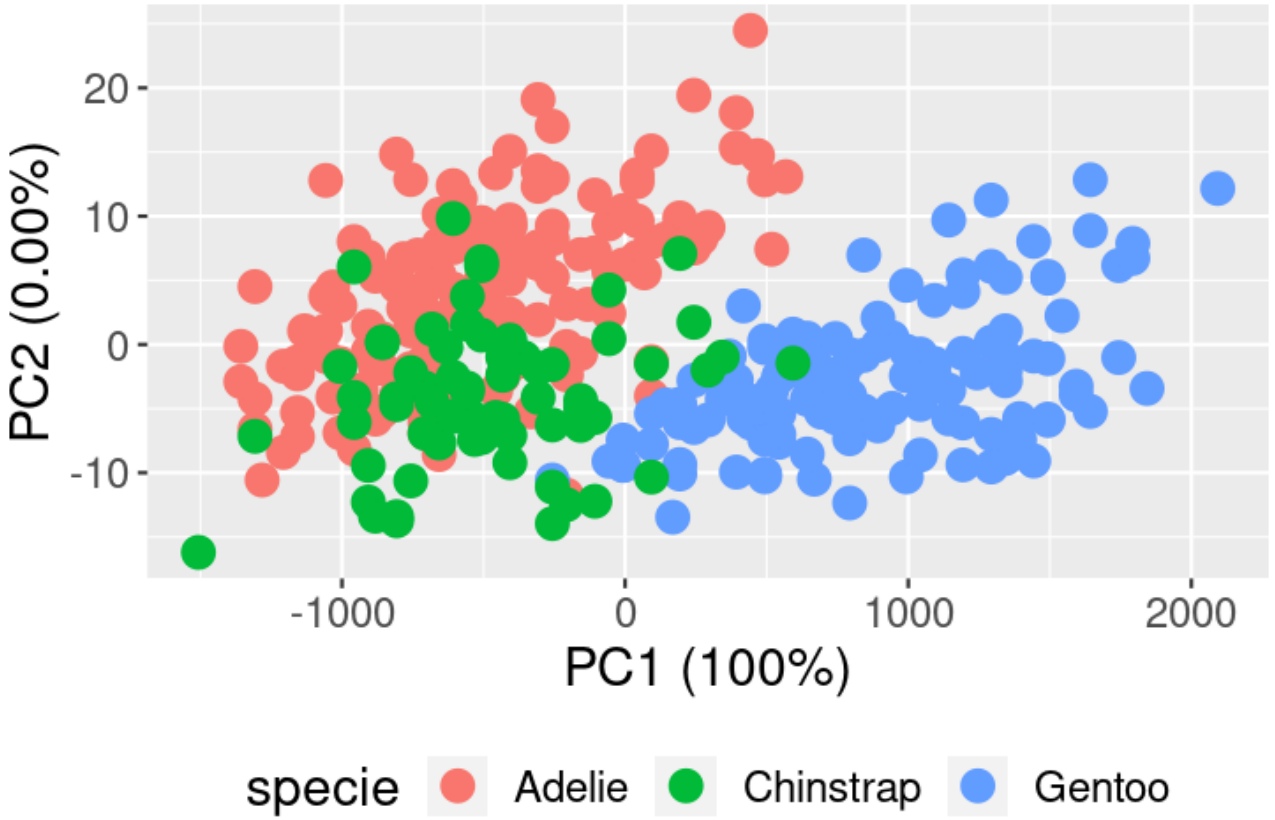
Part 4: Data with 3 variables

```
> head(data_try)
  species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1  Adelie           39.1          18.7               181        3750
2  Adelie           39.5          17.4               186        3800
3  Adelie           40.3          18.0               195        3250
5  Adelie           36.7          19.3               193        3450
6  Adelie           39.3          20.6               190        3650
7  Adelie           38.9          17.8               181        3625
```



Part 4: Data with 4 variables

```
> head(data_try)
  species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1  Adelie           39.1          18.7               181        3750
2  Adelie           39.5          17.4               186        3800
3  Adelie           40.3          18.0               195        3250
5  Adelie           36.7          19.3               193        3450
6  Adelie           39.3          20.6               190        3650
7  Adelie           38.9          17.8               181        3625
```



Part 4: Draw PCA in R

Loading data

```
data <- as.data.frame(penguins)
head(data)
dim(data)
str(data)
```

check and remove NA

```
colSums(is.na(data))
data_rm <- na.omit(data)
head(data_rm)
```

numeric data

```
number_data <- data_rm[,3:6]
```

Original data

```
> head(data)
  species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g    sex year
1  Adelie Torgersen           39.1          18.7                181        3750   male 2007
2  Adelie Torgersen           39.5          17.4                186        3800 female 2007
3  Adelie Torgersen           40.3          18.0                195        3250 female 2007
4  Adelie Torgersen             NA            NA                 NA          NA   <NA> 2007
5  Adelie Torgersen           36.7          19.3                193        3450 female 2007
6  Adelie Torgersen           39.3          20.6                190        3650   male 2007
```

Data input

```
> head(number_data)
  bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1           39.1          18.7                181        3750
2           39.5          17.4                186        3800
3           40.3          18.0                195        3250
5           36.7          19.3                193        3450
6           39.3          20.6                190        3650
7           38.9          17.8                181        3625
```

```
'data.frame':   333 obs. of  4 variables:
 $ bill_length_mm   : num  39.1 39.5 40.3 36.7 39.3 38.9 39.2 41.1 38.6 34.6 ...
 $ bill_depth_mm    : num  18.7 17.4 18 19.3 20.6 17.8 19.6 17.6 21.2 21.1 ...
 $ flipper_length_mm: int  181 186 195 193 190 181 195 182 191 198 ...
 $ body_mass_g      : int  3750 3800 3250 3450 3650 3625 4675 3200 3800 4400 ...
```

Part 4: Calculation PCA

```
### calculation PCA

library(factoextra)

Xpca <- prcomp(number_data)

str(Xpca)
```

Eigenvectors
Xpca$rotation

Eigenvalue
Xpca$sdev

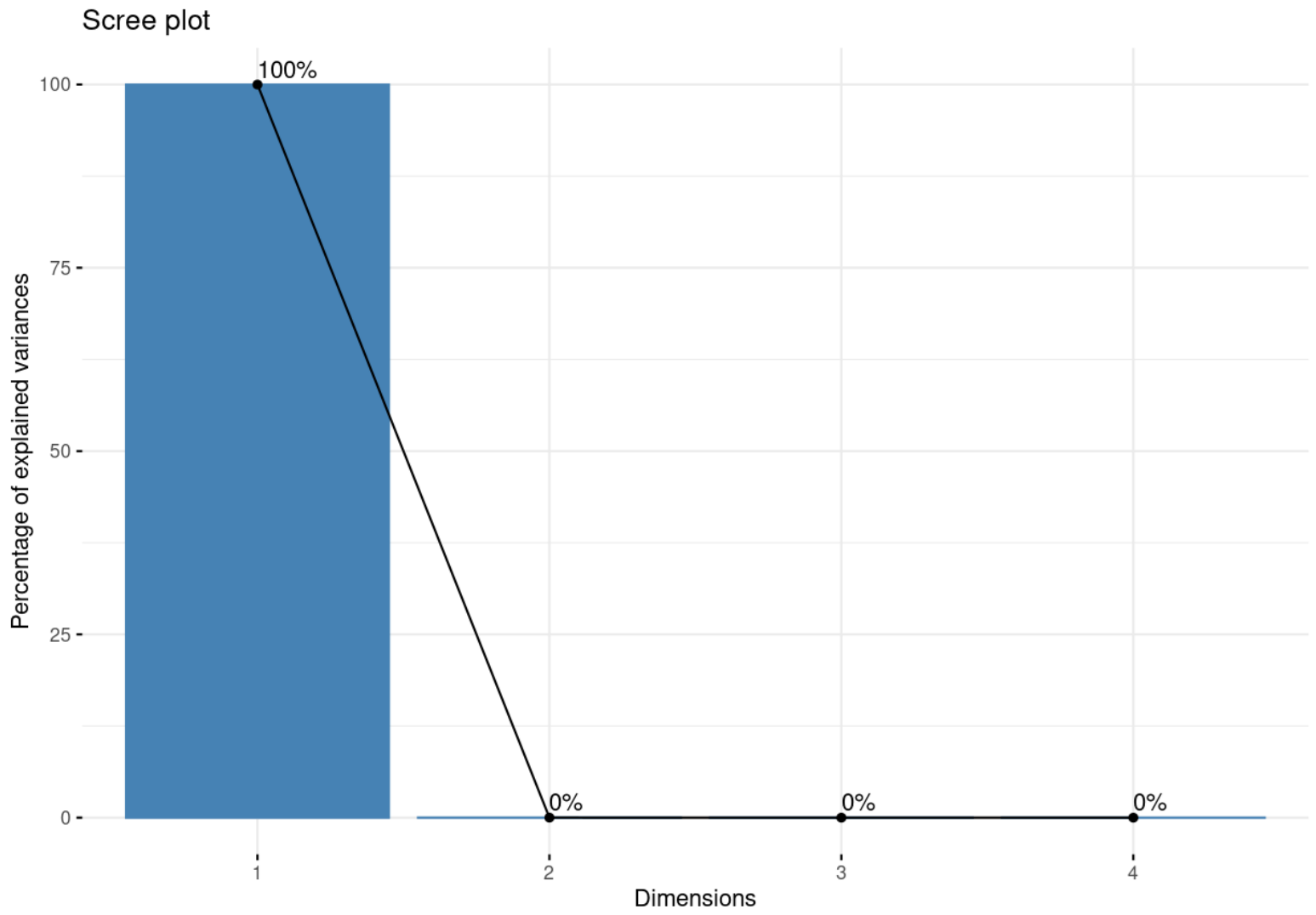
New coordinates
Xpca$x

```
> str(Xpca)
List of 5
 $ sdev    : num [1:4] 805.32 7.12 4.02 1.54
 $ rotation: num [1:4, 1:4] 0.004 -0.00115 0.01519 0.99988 -0.31928 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "bill_length_mm" "bill_depth_mm" "flipper_length_mm" "body_mass_g"
  .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
 $ center  : Named num [1:4] 44 17.2 201 4207.1
  ..- attr(*, "names")= chr [1:4] "bill_length_mm" "bill_depth_mm" "flipper_length_mm" "body_mass_g"
 $ scale   : logi FALSE
 $ x       : num [1:333, 1:4] -457 -407 -957 -757 -557 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:333] "1" "2" "3" "5" ...
  .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
 - attr(*, "class")= chr "prcomp"
```

https://www.youtube.com/watch?v=FgakZw6K1QQ&t=619s

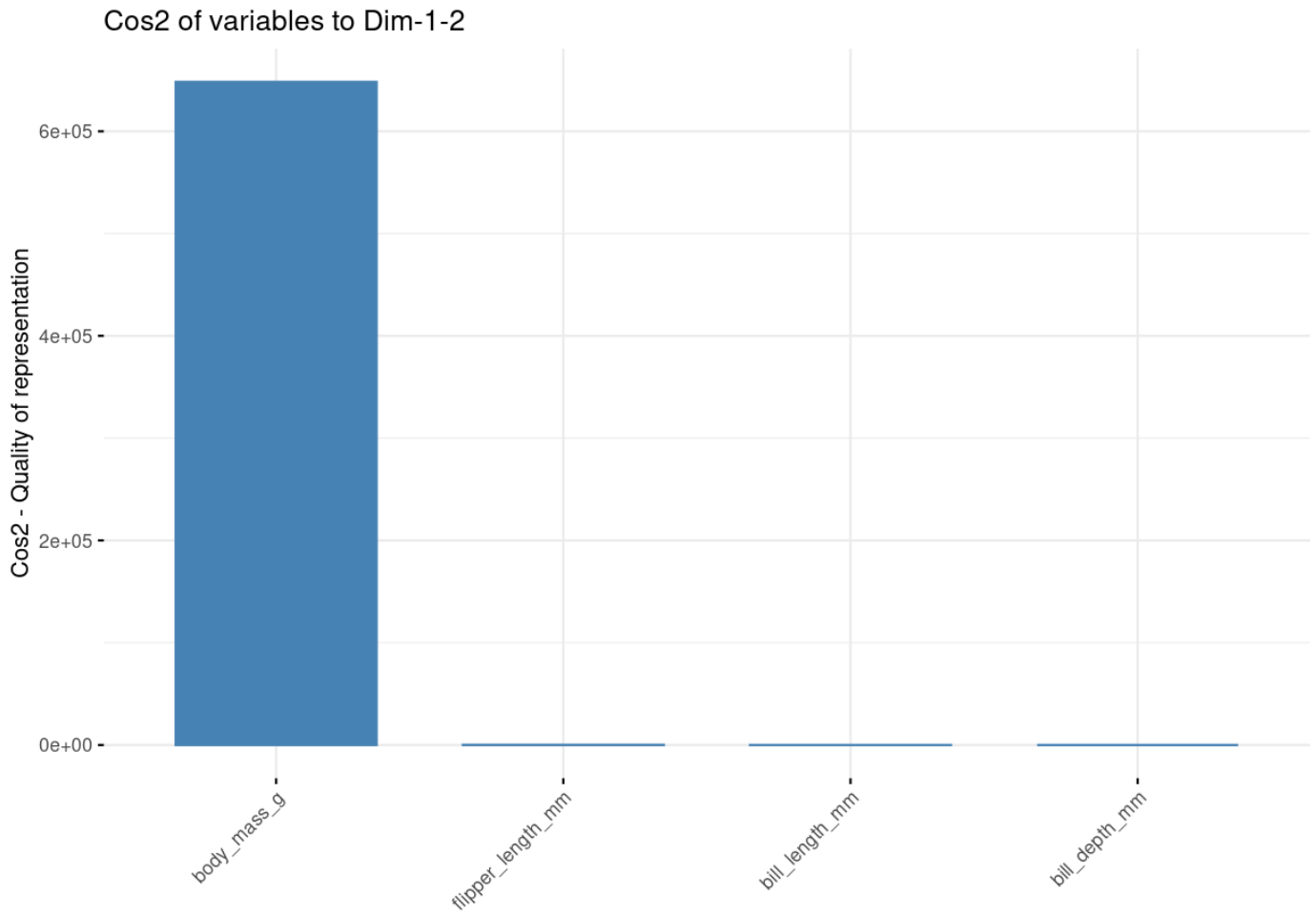
Part 4: Scree Plot

```
# Scree Plot

fviz_eig(Xpca, addlabels=TRUE, ncp=20)
```



Part 4: Contribution of each variable

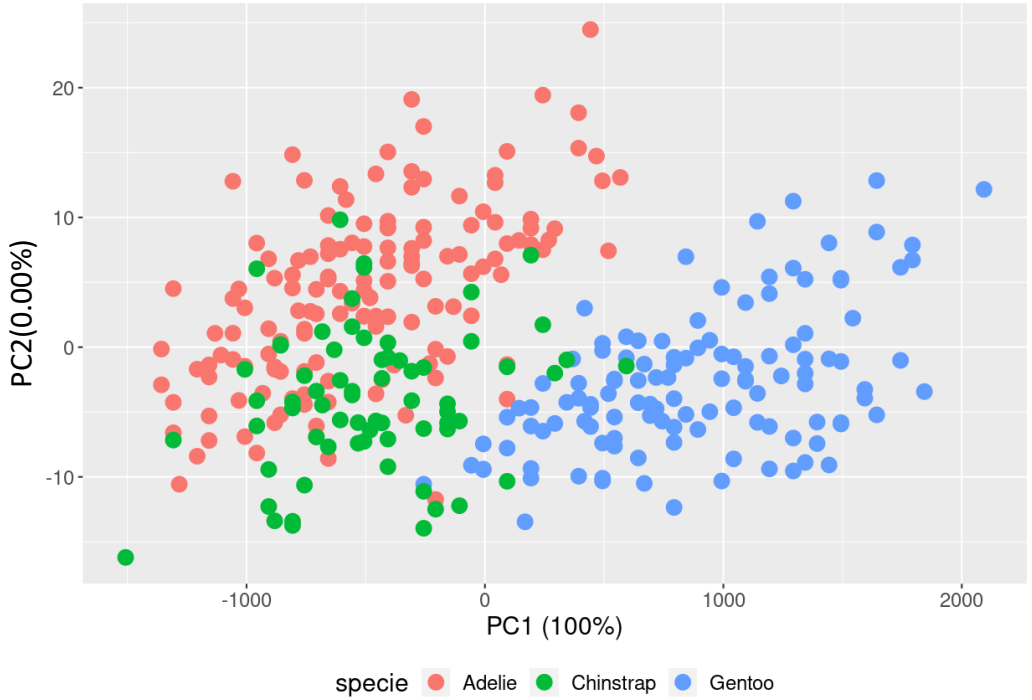
```
# Contribution of each variable

fviz_cos2(Xpca, choice = "var", axes = 1:2)
```



Part 4: Plot PCA

```
### draw PCA
data_PCA <- as.data.frame(Xpca$x)
data_PCA$specie <- data_rm$specie
ggplot(data_PCA, aes(x = PC1, y = PC2, col=specie)) + geom_point(size=4, alpha=2) +
 theme(legend.position="bottom") + theme(text = element_text(size = 15))+
 labs(x="PC1 (100%)", y="PC2 (0.00%)")
```

```
> head(data_PCA)
        PC1       PC2         PC3        PC4 specie
1 -457.3251 13.351587 -1.23656032 -0.3358031 Adelie
2 -407.2522  9.179113  0.04892085 -1.0399675 Adelie
3 -957.0447 -8.160444  2.52578133  0.8157454 Adelie
5 -757.1158 -1.867653  4.90888951  2.1636557 Adelie
6 -557.1773  3.389158  1.15096638  2.7026235 Adelie
7 -582.3093 11.372652 -0.78874833 -1.1552328 Adelie
```

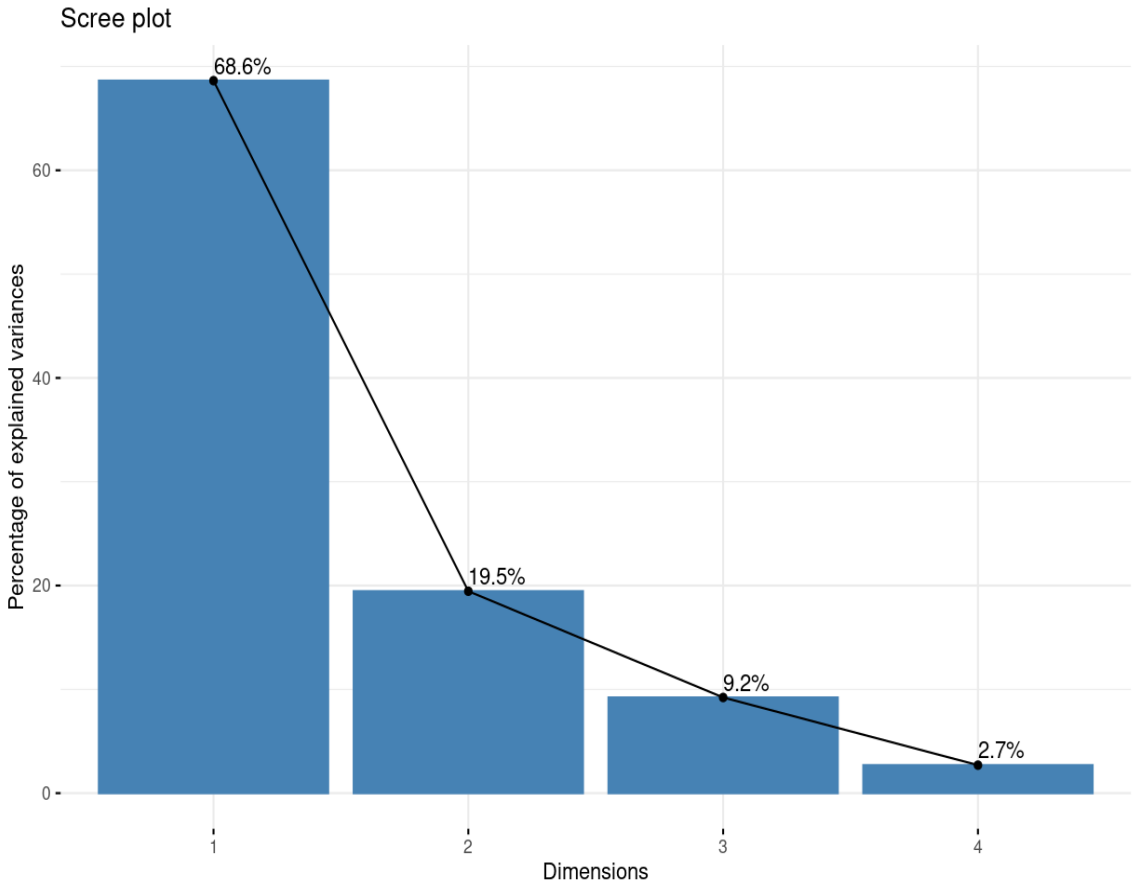


Part 4: Normalise data

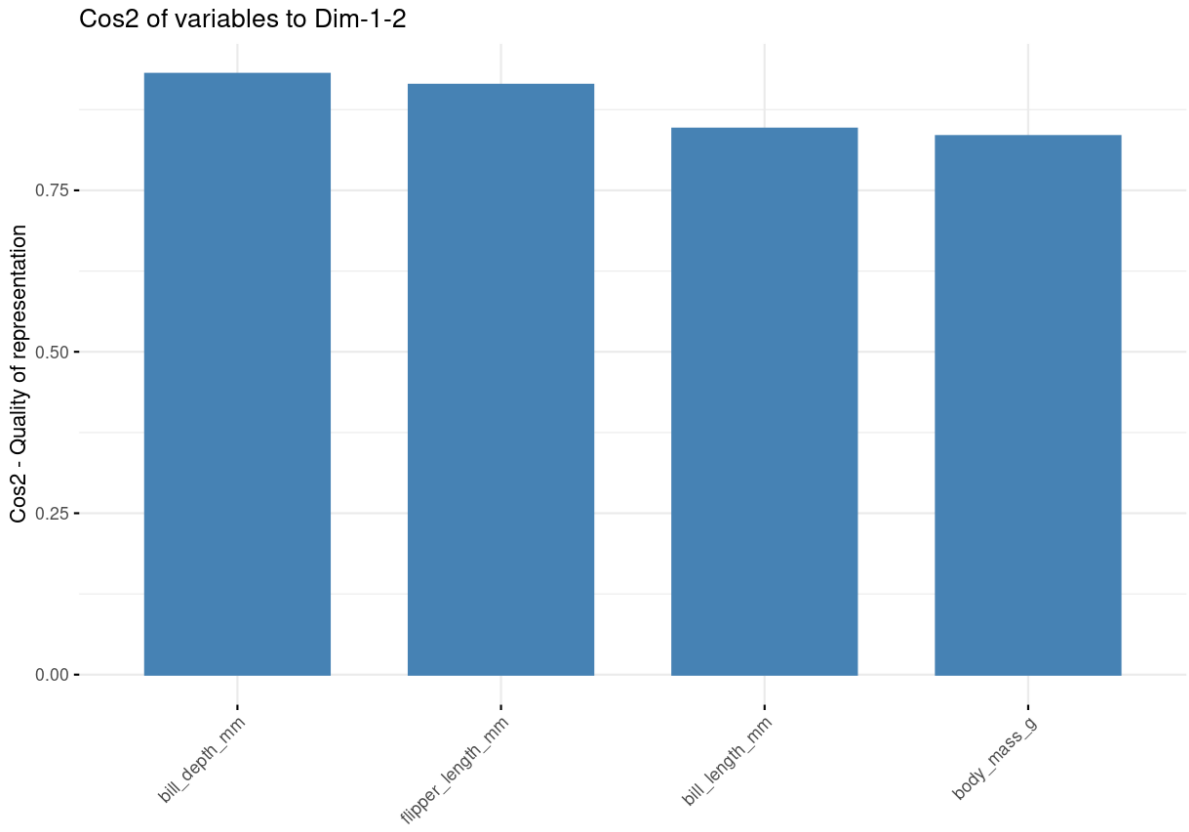
```
 > colMeans(number_data)
   bill_length_mm     bill_depth_mm  flipper_length_mm       body_mass_g
        43.99279          17.16486          200.96697              4207.05706
### normalise
> data_normalized <- scale(number_data)
> colMeans(data_normalized)
bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g
    3.552797e-16      5.572578e-16      1.834379e-16     -9.274780e-17
```

Part 4: Normalise data

fviz_eig(Xpca1, addlabels=TRUE, ncp=20)

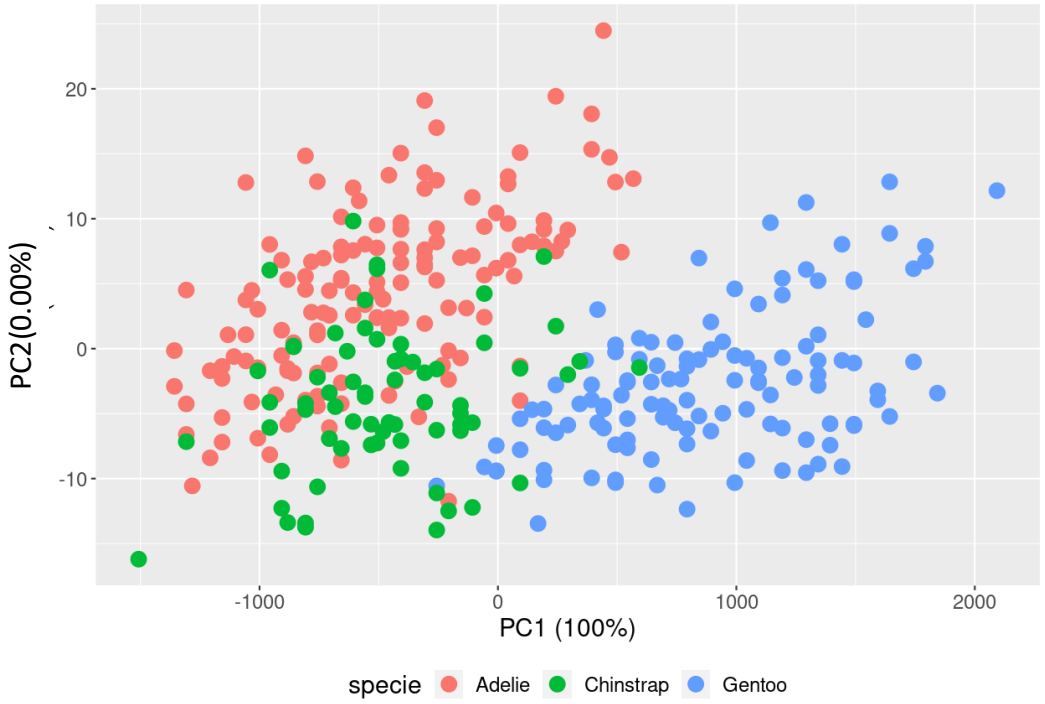


fviz_cos2(Xpca1, choice = "var", axes = 1:2)

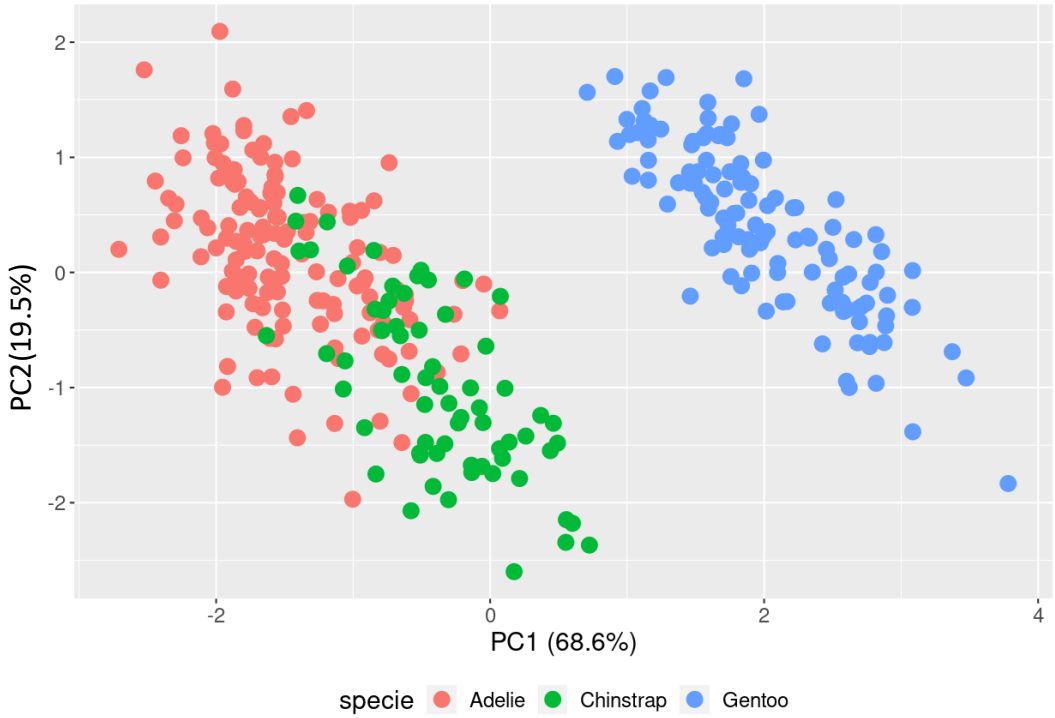


Part 4: Normalise data

Before



After



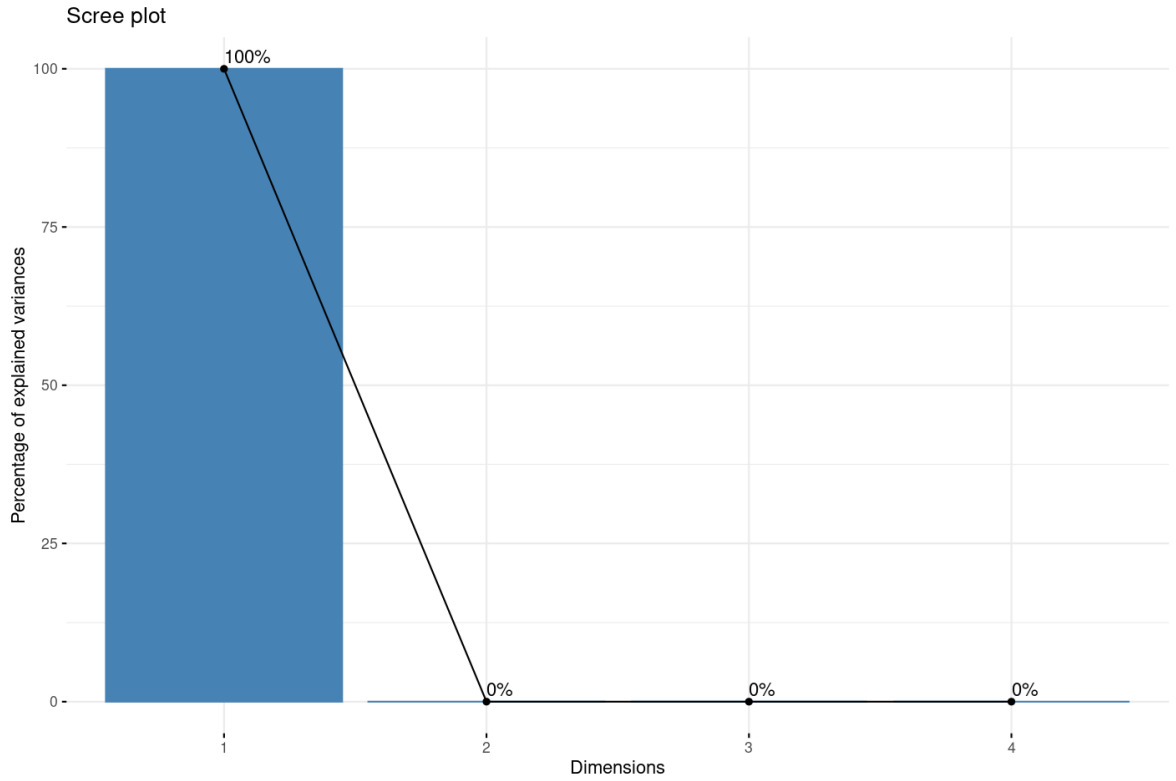
Part 4: Structure of data

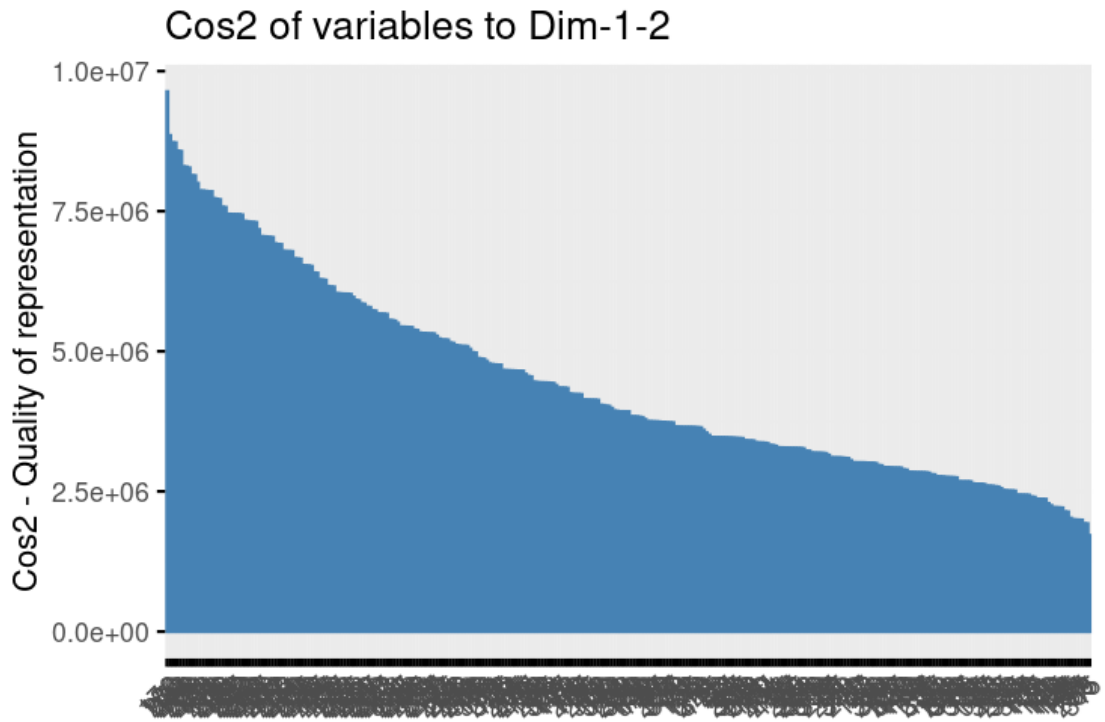
```
> head(new_data)
                    1     2     3     5     6     7     8    13    14    15    16    17    18    19
bill_length_mm     39.1  39.5  40.3  36.7  39.3  38.9  39.2  41.1  38.6  34.6  36.6  38.7  42.5  34.4
bill_depth_mm      18.7  17.4  18.0  19.3  20.6  17.8  19.6  17.6  21.2  21.1  17.8  19.0  20.7  18.4
flipper_length_mm 181.0 186.0 195.0 193.0 190.0 181.0 195.0 182.0 191.0 198.0 185.0 195.0 197.0 184.0
```

```
dim(number_data)
[1] 333   4
```

→

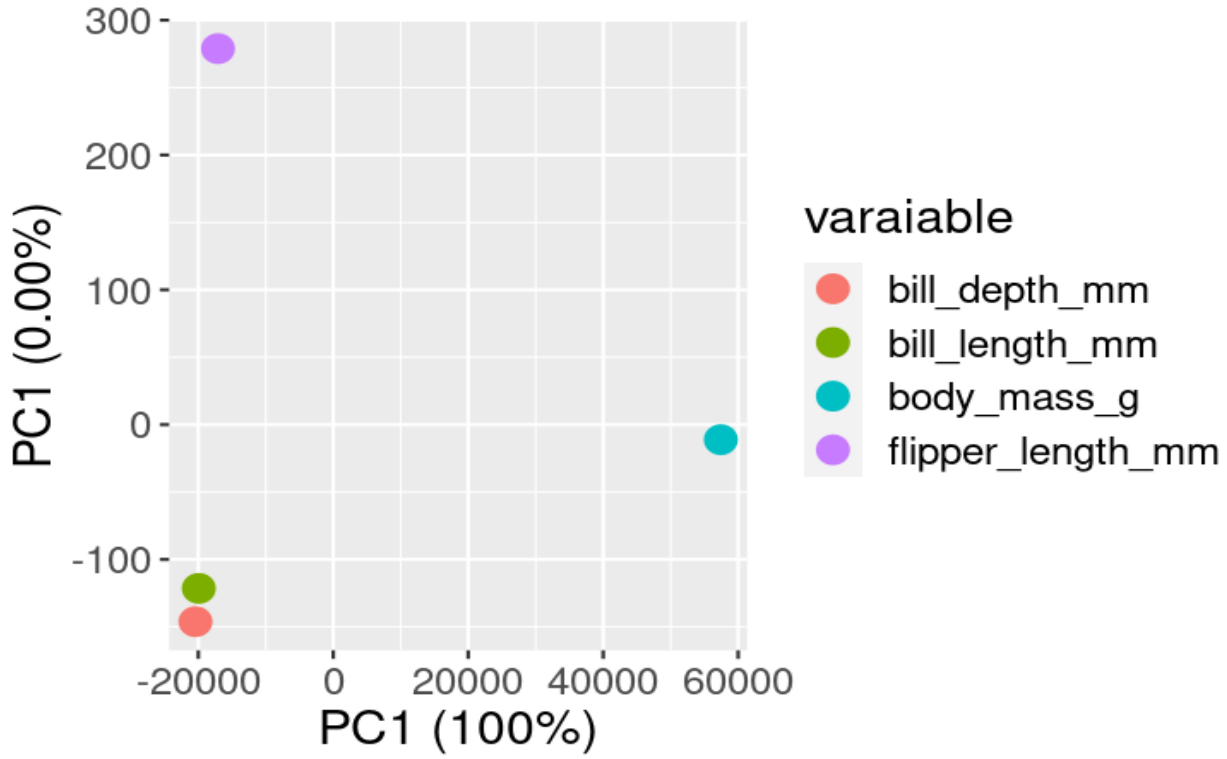
```
> dim(new_data)
[1]   4 333
```





Part 4: Structure of data

```
### draw PCA
data_PCA <- as.data.frame(Xpca2$x)
data_PCA$varaiable  <- rownames(new_data)
head(data_PCA)
ggplot(data_PCA, aes(x = PC1, y = PC2, col=varaiable)) + geom_point(size=4, alpha=2) +
  theme(legend.position="right") + theme(text = element_text(size = 15))+
  labs(x="PC1 (100%)", y="PC1 (0.00%)")
```

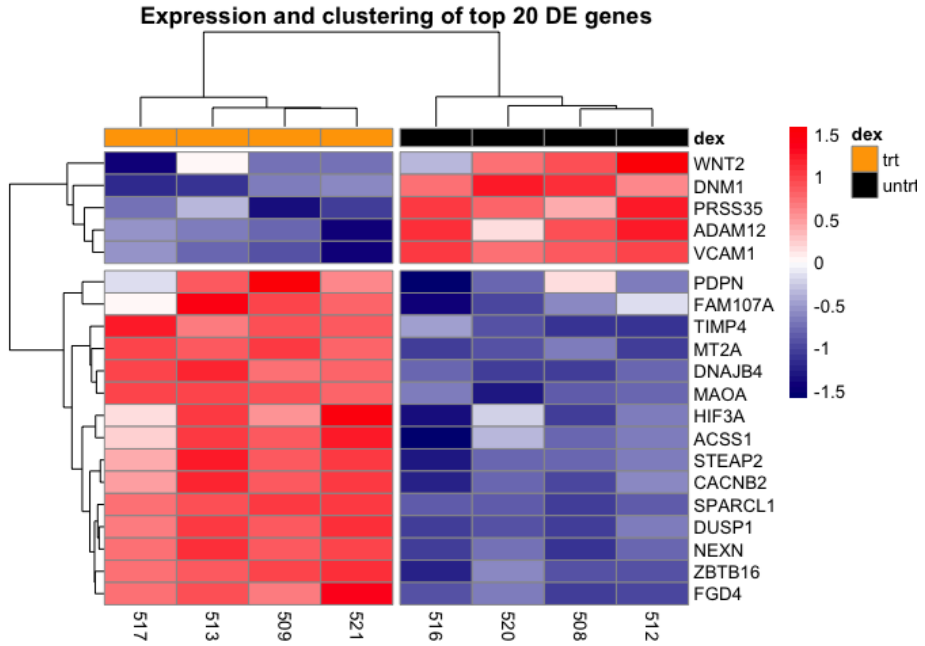


Part 5: Heatmap

A heatmap (or heat map) is another way to **visualize hierarchical clustering**. It's also called a false colored image, where data values are transformed to color scale.

Heat maps allow us to simultaneously **visualize clusters of samples and features.**

- First **hierarchical clustering is done** of both the **rows and the columns** of the data matrix. The columns/rows of the data **matrix are re-ordered according to the hierarchical clustering result**, putting similar observations close to each other.
- The **blocks of 'high' and 'low' values are adjacent in the data matrix**.
- Finally, a c**olor scheme is applied** for the visualization and the data matrix is displayed. Visualizing the data matrix in this way can help to find the variables that appear to be characteristic for each sample cluster.



https://btep.ccr.cancer.gov/docs/data-visualization-with-r/Lesson5_intro_to_ggplot/

Part 5: Type of Heatmap

Clustered Heatmap

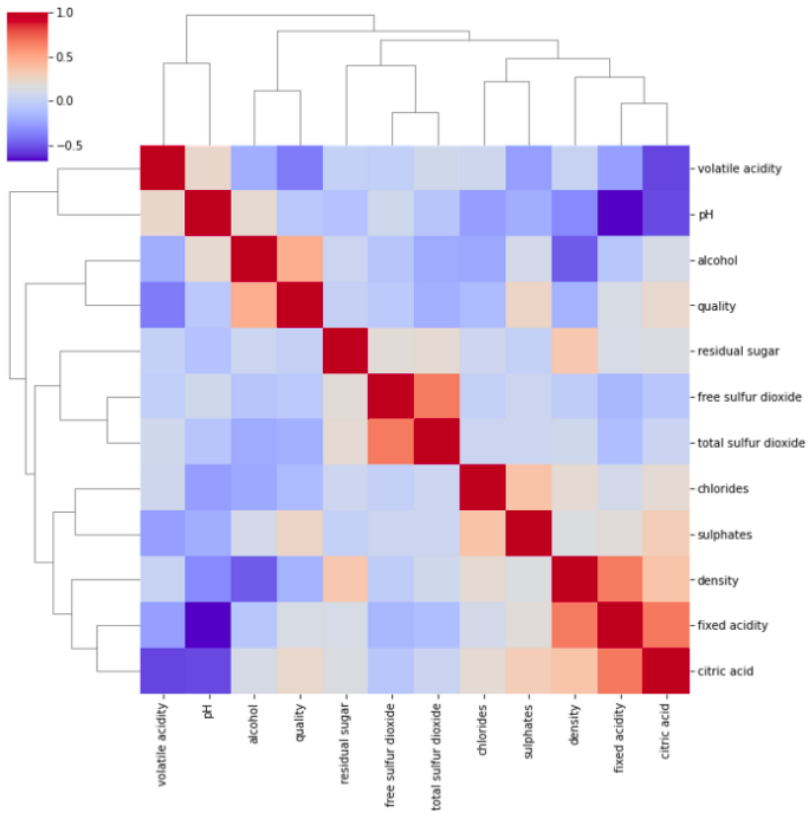


Image by the Author

Correlogram

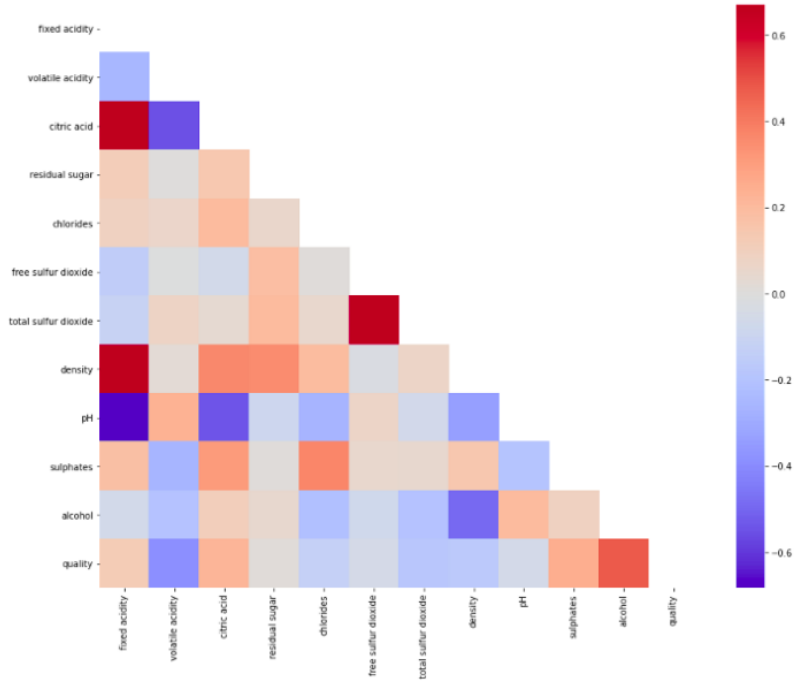
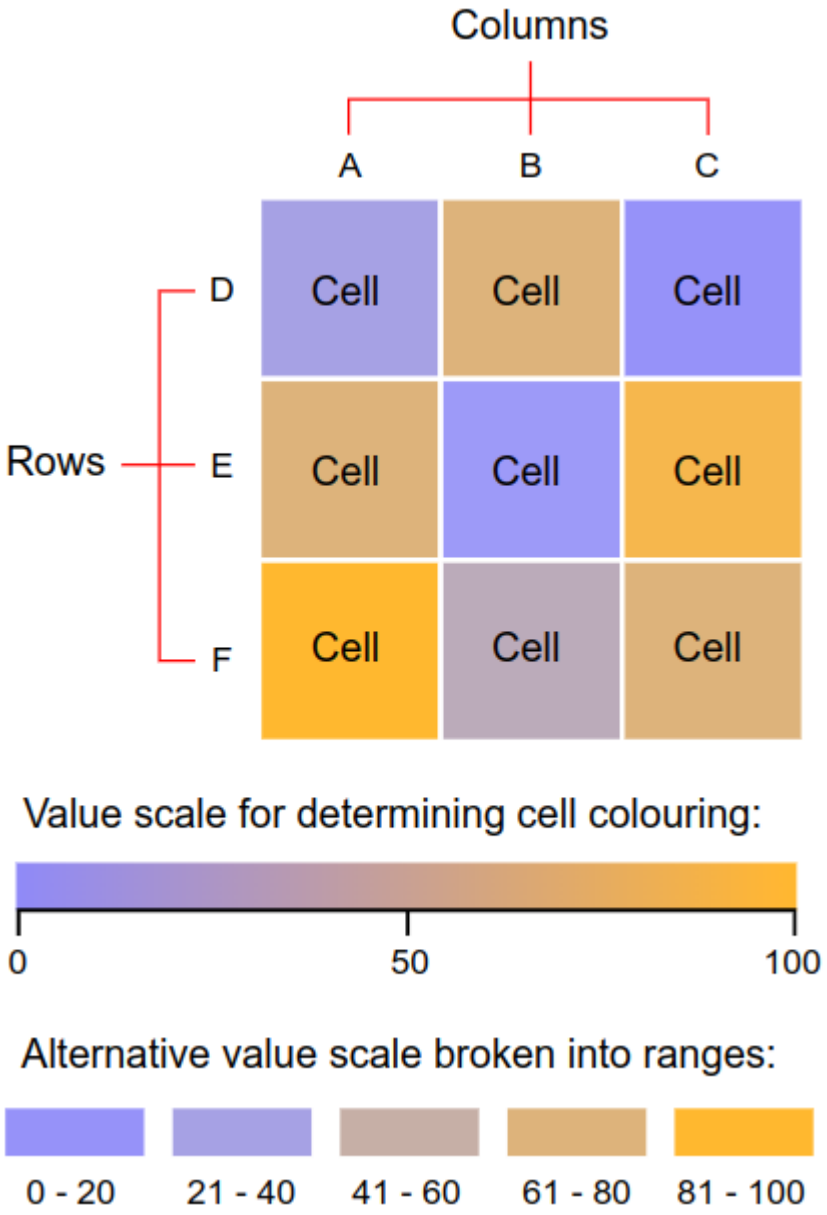


Image by the Author

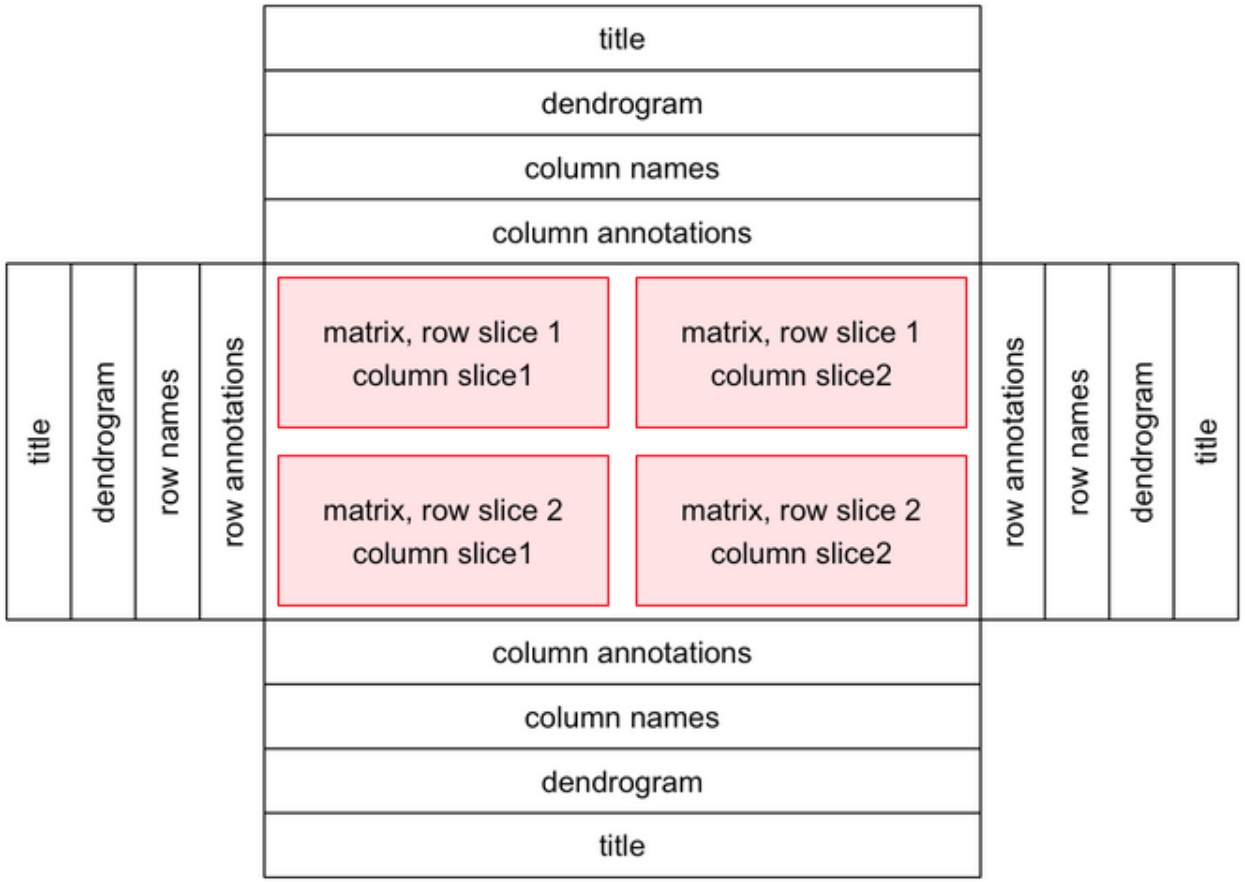
Clustered Heatmap: The goal of Clustered Heatmap is to build associations between both the data points and their features. This type of heatmap implements clustering as part of the process of grouping similar features. Clustered Heatmaps are widely used in biological sciences for studying gene similarities across individuals.

Correlogram: A correlogram replaces each of the variables on the two axes with numeric variables in the dataset. Each square depicts the relationship between the two intersecting variables, which helps to build descriptive or predictive statistical models.

Part 5: Structure of Heatmap



packages ComplexHeatmap



Part 5: Introduction data

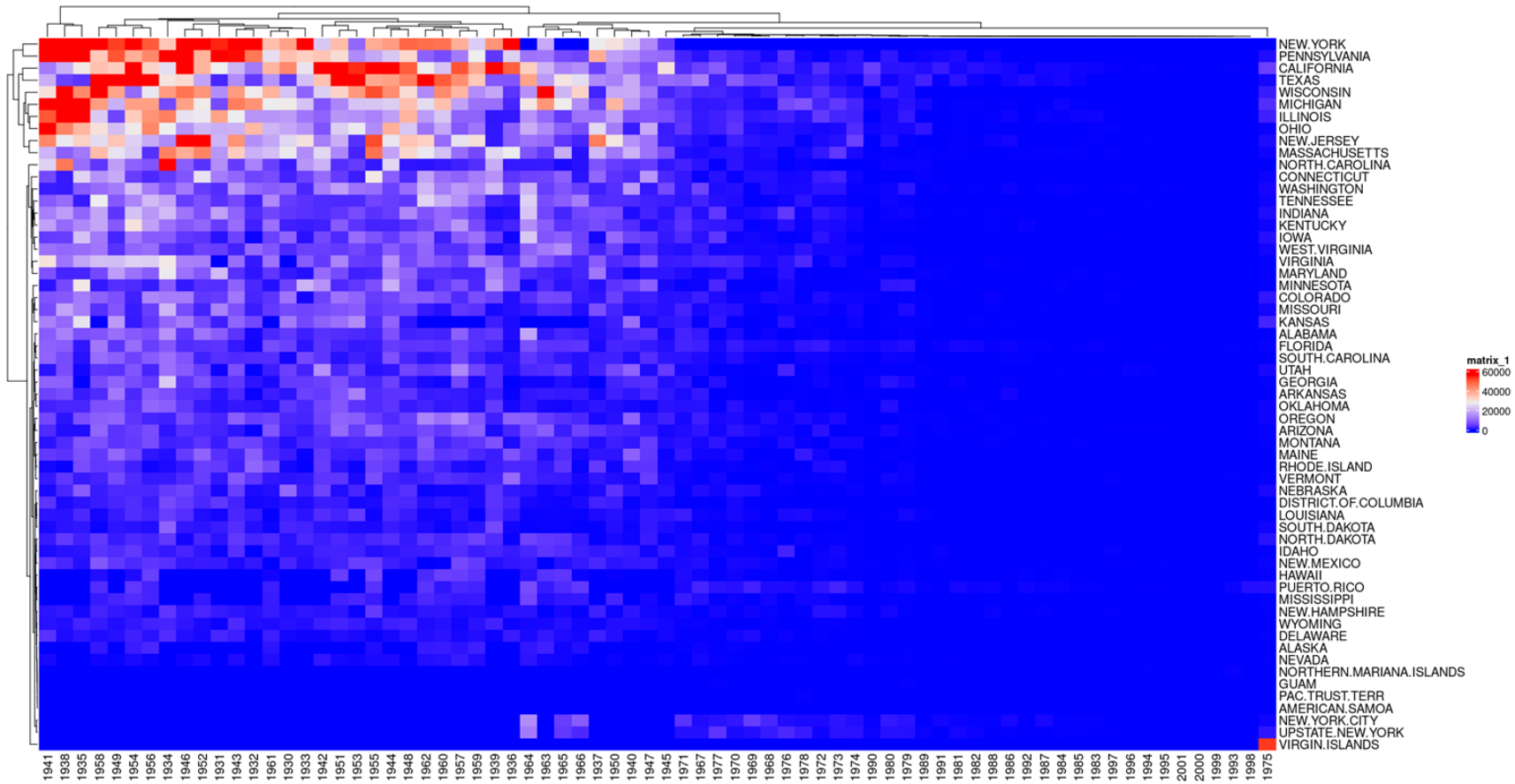
Measles cases in US states 1930-2001

```
> head(data)
               1930  1931  1932  1933  1934  1935  1936 1937  1938  1939  1940  1941  1942  1943  1944
ALABAMA        4389  8934   270  1735 15849  7214   572  620 13511  4381  3052  8696  3564  3865  7199
ALASKA            0     0     0     0     0     0     0    0     0     0     0     0     0     0     0
AMERICAN.SAMOA    0     0     0     0     0     0     0    0     0     0     0     0     0     0     0
ARIZONA        2107  2135    86  1261  1022   586  2378 3793   604   479  2078  2929  3813  1002  4641
ARKANSAS        996   849    99  5438  7222  1518   107  322  6690  1724  1096  5474  4542  2867  3780
CALIFORNIA    43585 27807 12618 26551 25650 28799 49092 5107 19452 67180 10391 12876 97526 20093 58518
```

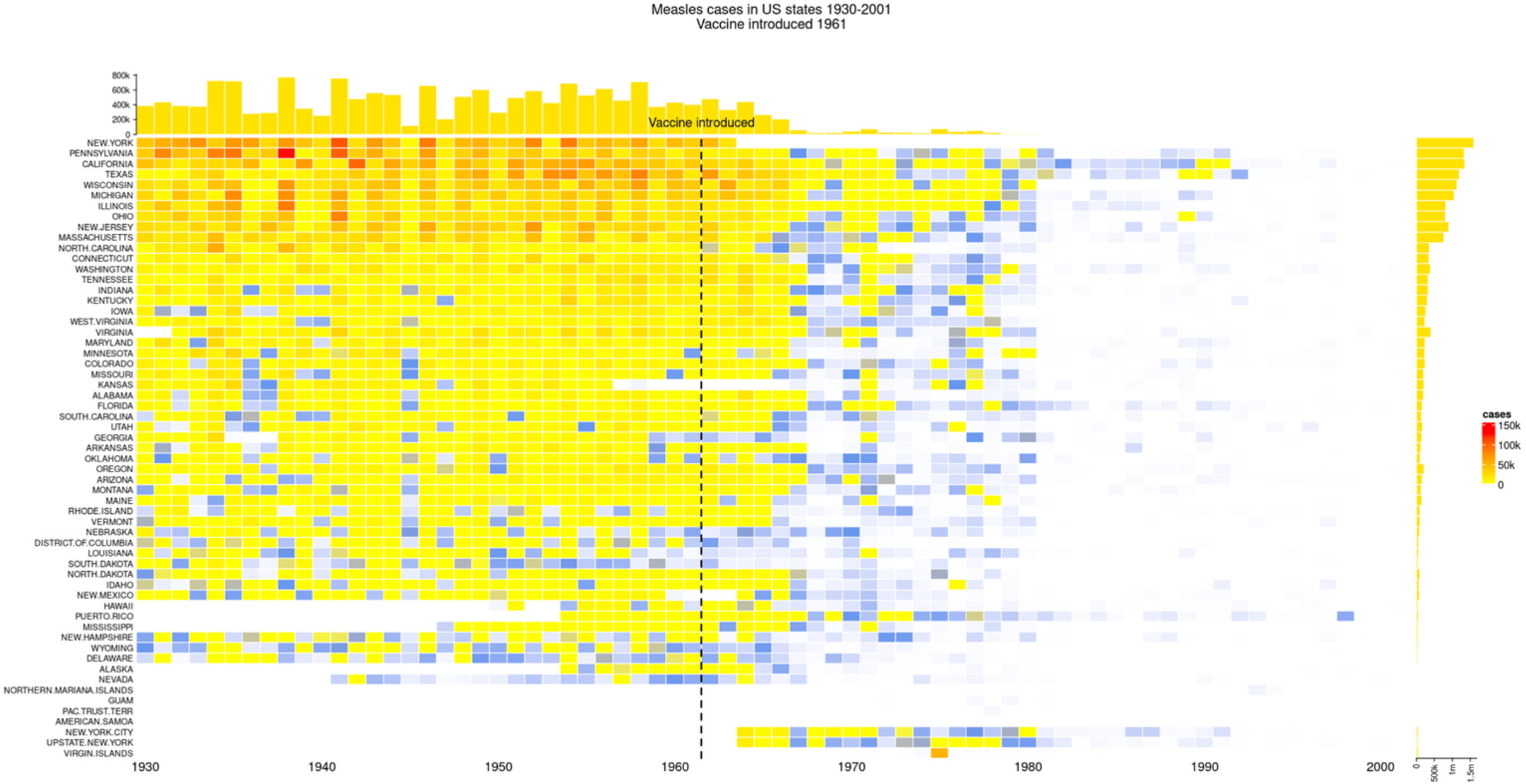
Vaccine introduced 1961

Part 5: Draw heatmap

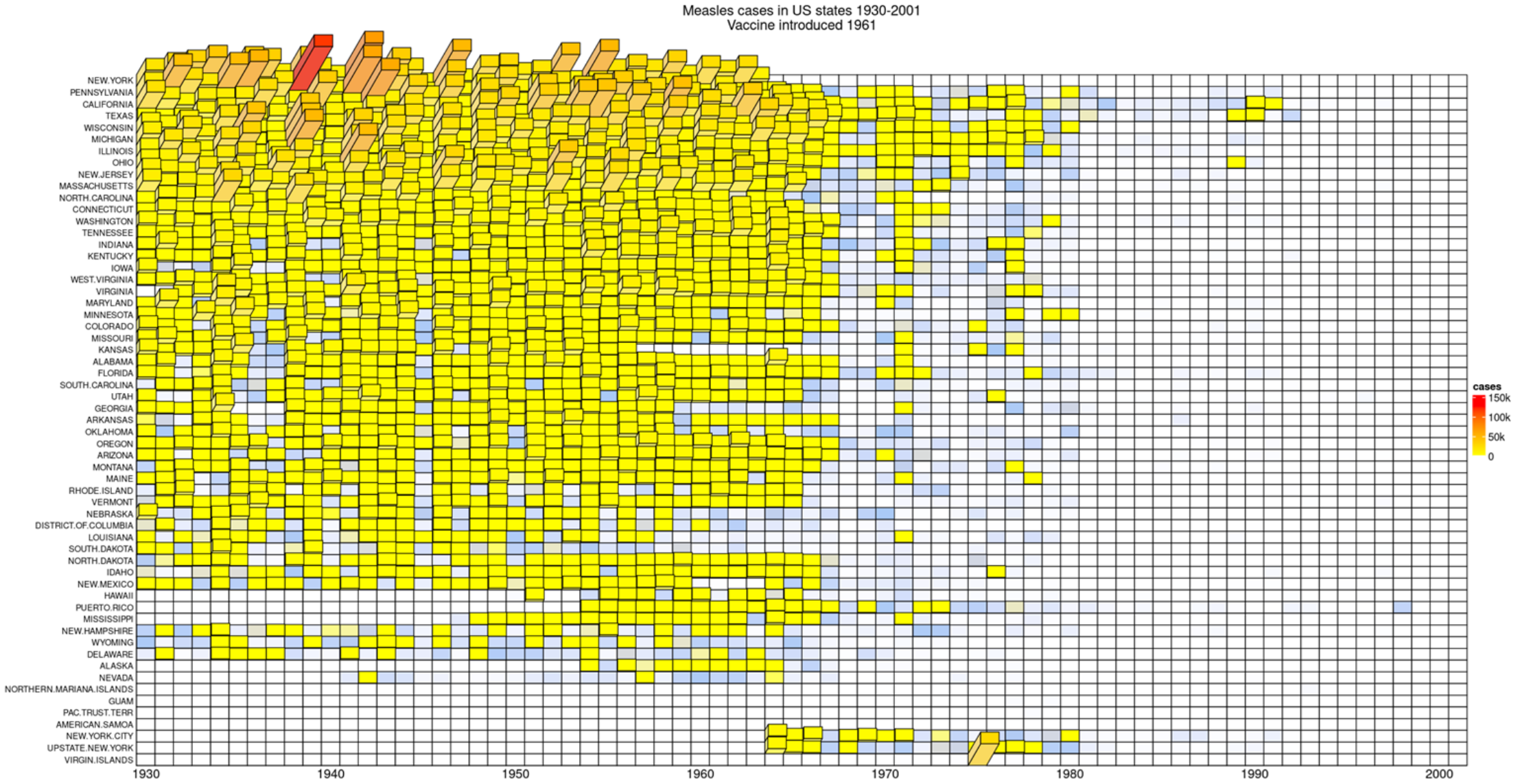
```
install.packages("ComplexHeatmap")

library(ComplexHeatmap)

## loading data

data  = readRDS(system.file("extdata", "measles.rds", package = "ComplexHeatmap"))

## check dimension

dim(data)

#[1] 59 72

head(data)

## check type of data

class(data)

#[1] "matrix" "array"

## draw simple heatmap

Heatmap(data)
```



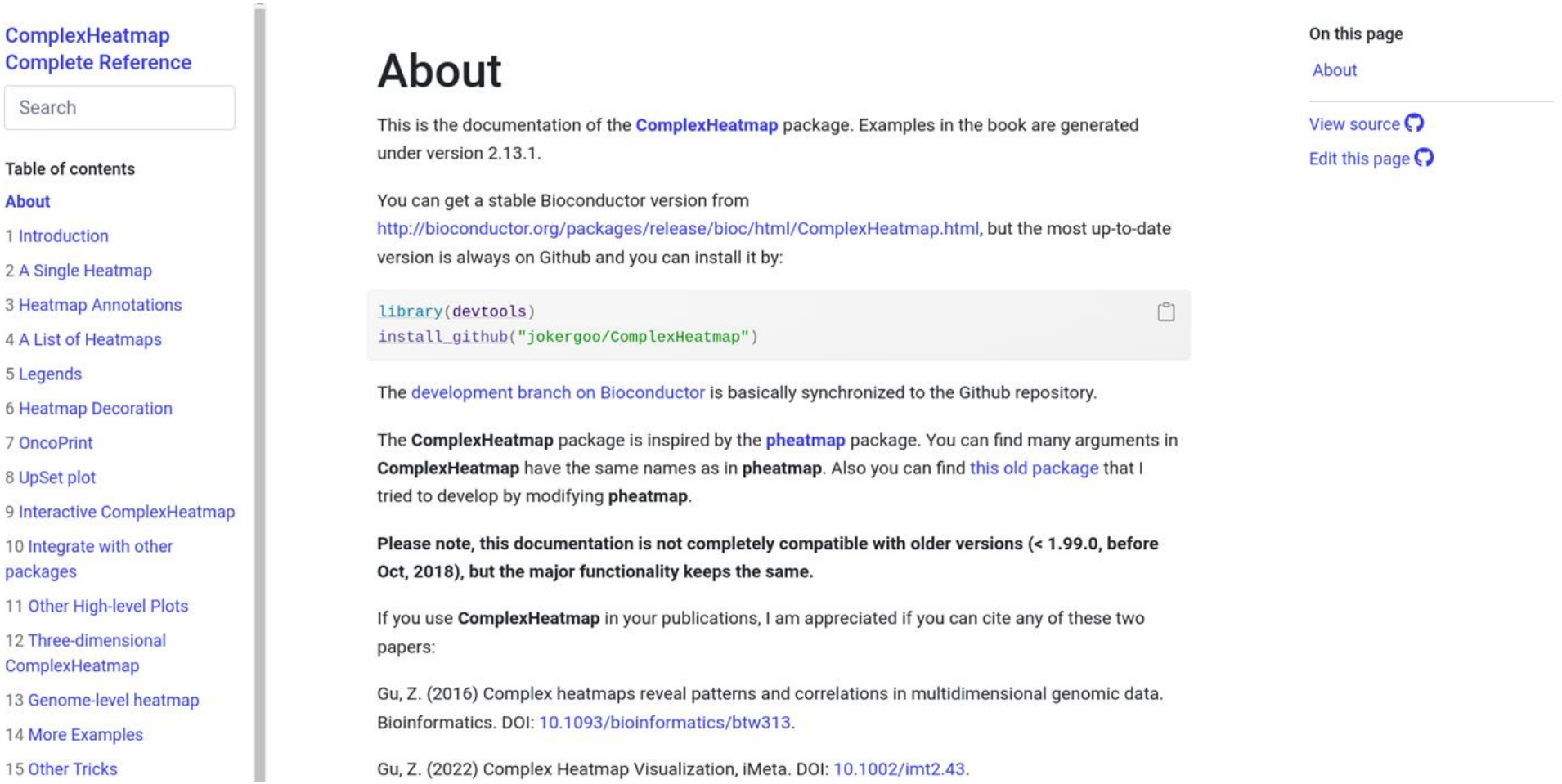
Part 5: Edit heatmap



Part 5: Make a 3D plot



Part 5: How to learn more?



ComplexHeatmap Complete Reference

Search

Table of contents

About
1 Introduction
2 A Single Heatmap
3 Heatmap Annotations
4 A List of Heatmaps
5 Legends
6 Heatmap Decoration
7 OncoPrint
8 UpSet plot
9 Interactive ComplexHeatmap
10 Integrate with other packages
11 Other High-level Plots
12 Three-dimensional ComplexHeatmap
13 Genome-level heatmap
14 More Examples
15 Other Tricks

About

This is the documentation of the **ComplexHeatmap** package. Examples in the book are generated under version 2.13.1.

You can get a stable Bioconductor version from http://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html, but the most up-to-date version is always on Github and you can install it by:

```
library(devtools)
install_github("jokergoo/ComplexHeatmap")
```

The development branch on Bioconductor is basically synchronized to the Github repository.

The **ComplexHeatmap** package is inspired by the **pheatmap** package. You can find many arguments in **ComplexHeatmap** have the same names as in **pheatmap**. Also you can find this old package that I tried to develop by modifying **pheatmap**.

Please note, this documentation is not completely compatible with older versions (< 1.99.0, before Oct, 2018), but the major functionality keeps the same.

If you use **ComplexHeatmap** in your publications, I am appreciated if you can cite any of these two papers:

Gu, Z. (2016) Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics. DOI: 10.1093/bioinformatics/btw313.

Gu, Z. (2022) Complex Heatmap Visualization, iMeta. DOI: 10.1002/imt2.43.

On this page

About

View source

Edit this page

Link: https://jokergoo.github.io/ComplexHeatmap-reference/book/

Reference

- http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R Code.html#1.%20Understanding%20the%20general%20ggplot%20format
- https://www.datacamp.com/tutorial/pca-analysis-r
- https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html
- https://statisticsglobe.com/heatmap-in-r
- http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/118-principal-component-analysis-in-r-prcomp-vs-princomp/
- data:
- https://www.eea.europa.eu/data-and-maps/indicators/13.2-development-in-consumption-of-2/assessment-1
- https://allisonhorst.github.io/palmerpenguins/

THANKS FOR YOUR LISTENING AND WATCHING