Intro to R: Part 2 - Data Visualization with ggplot2

2024-09-06

Data Visualization in R
BCBB Summer R Seminar
Mina Peyton

Shift + Command + Return == run current chunk

ggplot2 in tidyverse (https://ggplot2.tidyverse.org/)

ggplot2 package info (https://cran.r-project.org/web/packages/ggplot2/index.html)
"ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics
You provide the data, tell 'ggplot2' how to map variables to aesthetics,
what graphical primitives to use, and it takes care of the details."

Posit cheatsheets (https://posit.co/resources/cheatsheets/)

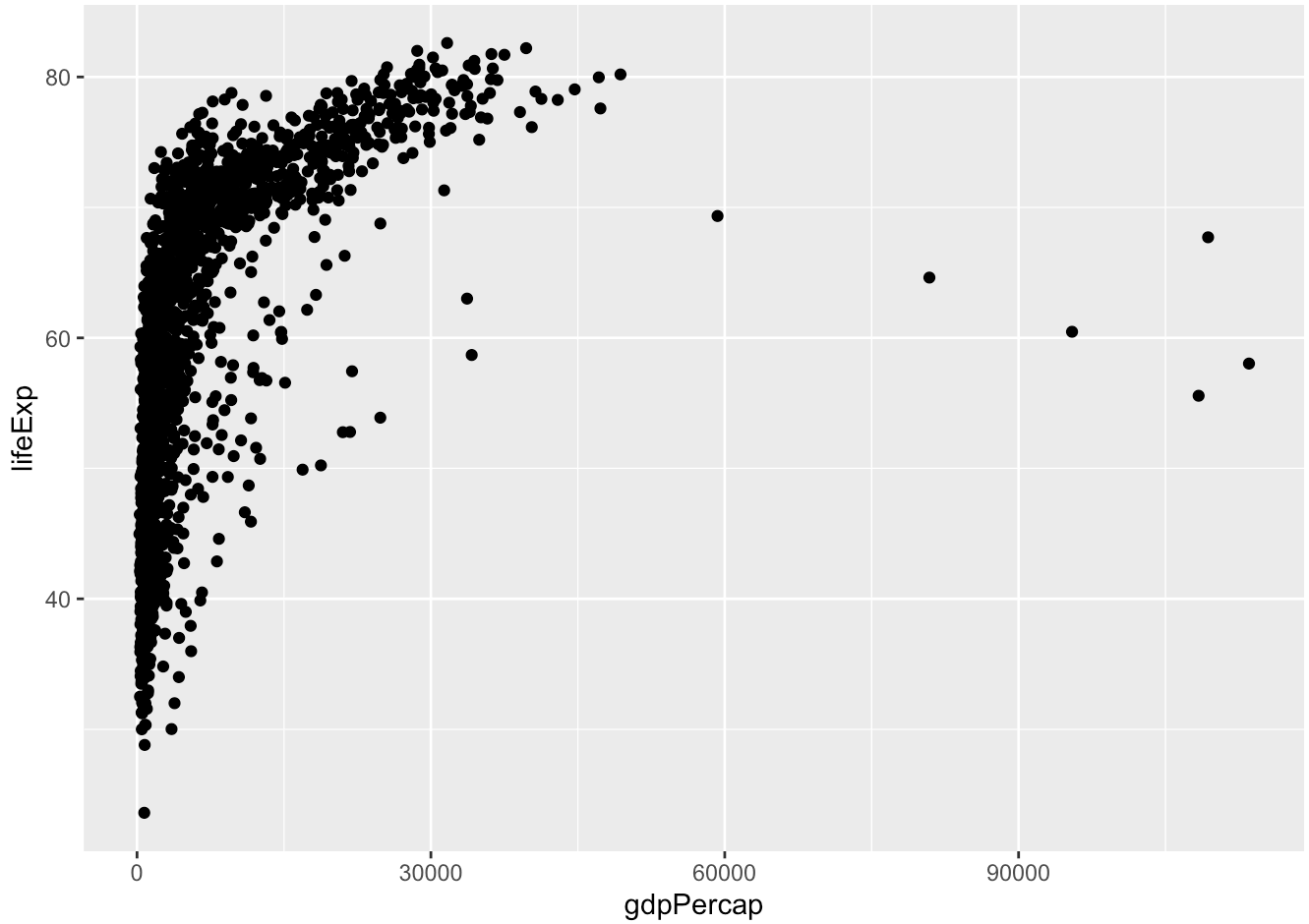
```
?ggplot()
```

Adapted from Software Carpentry (https://software-carpentry.org/lessons/)

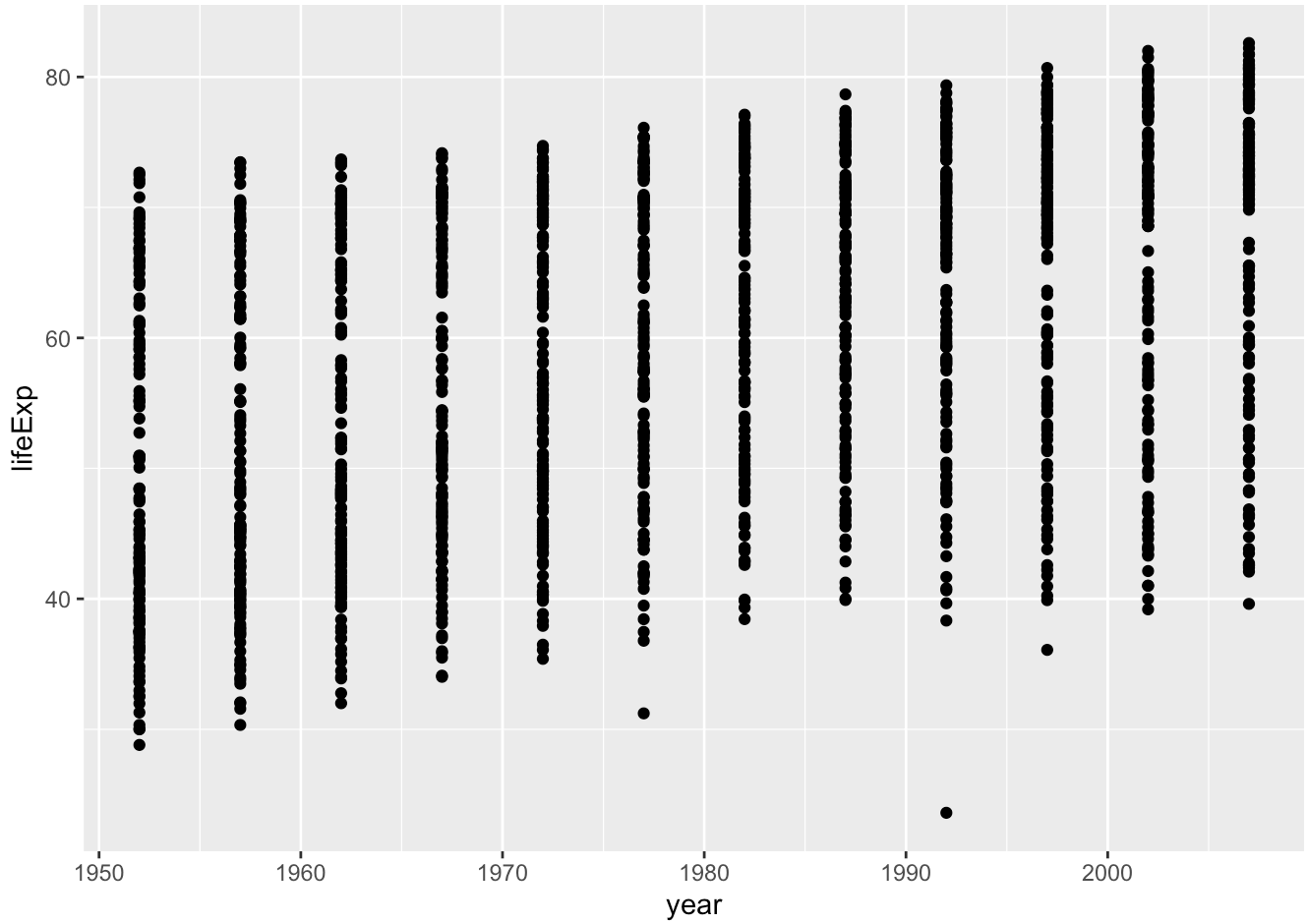
Gapminder tracks economic and social indicators like life expectancy and the
GDP per capita of countries over time

1. Scatterplot

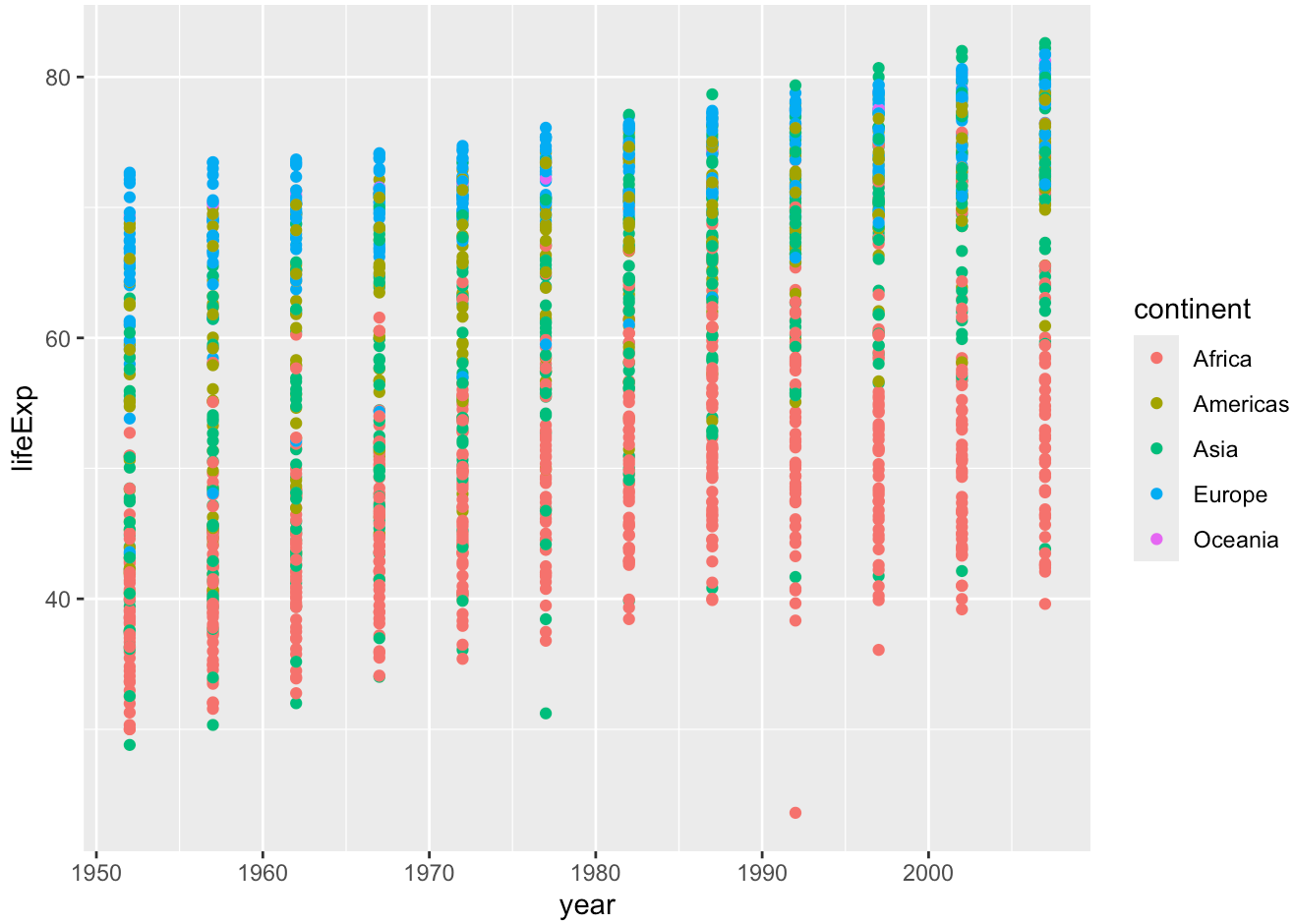
```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```




```
# life expentency over time
ggplot(data = gapminder, mapping = aes(x = year, y = lifeExp)) + geom_point()
```

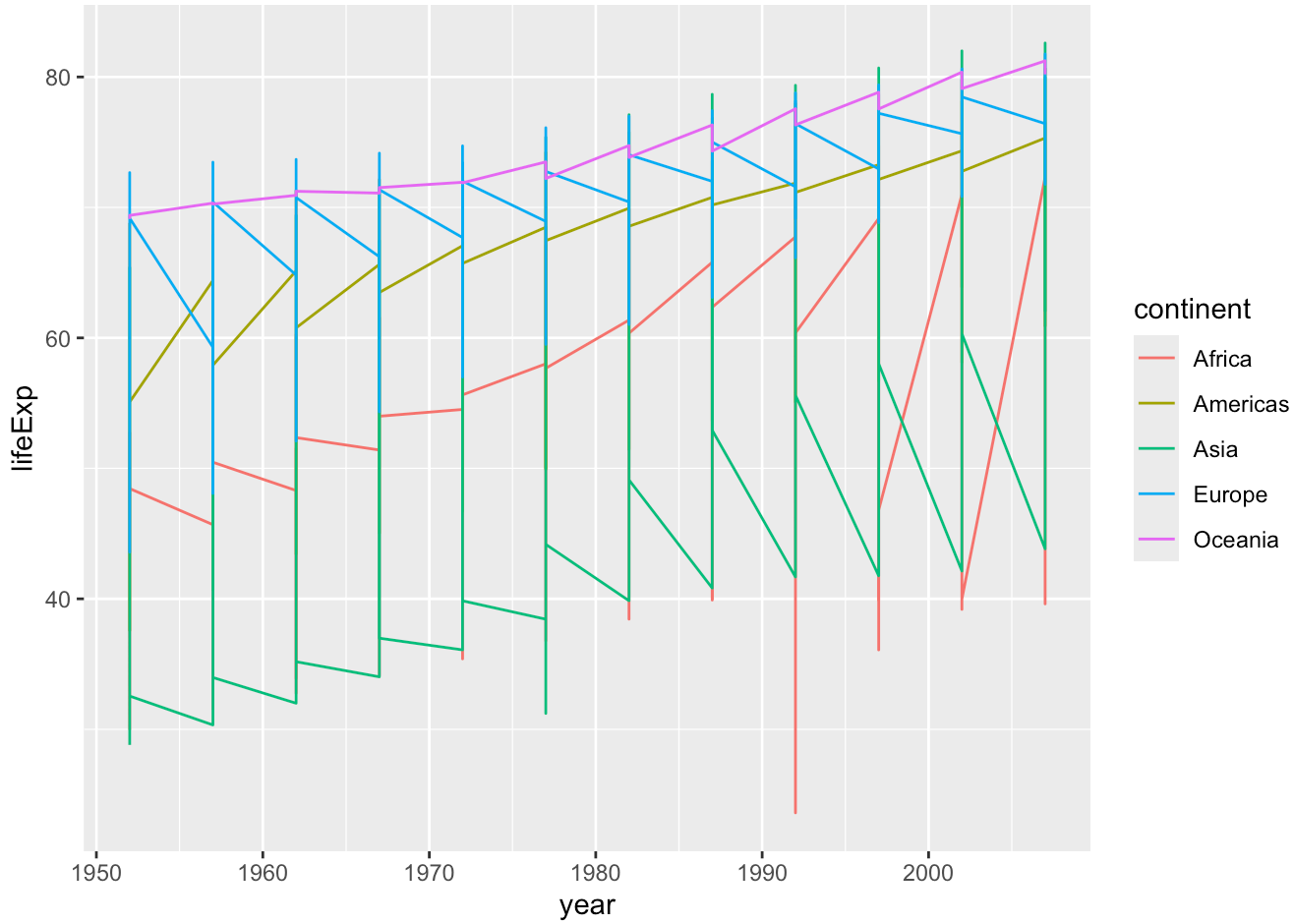



```
# color by continent
ggplot(data = gapminder, mapping = aes(x = year, y = lifeExp, color=continent)) +
  geom_point()
```



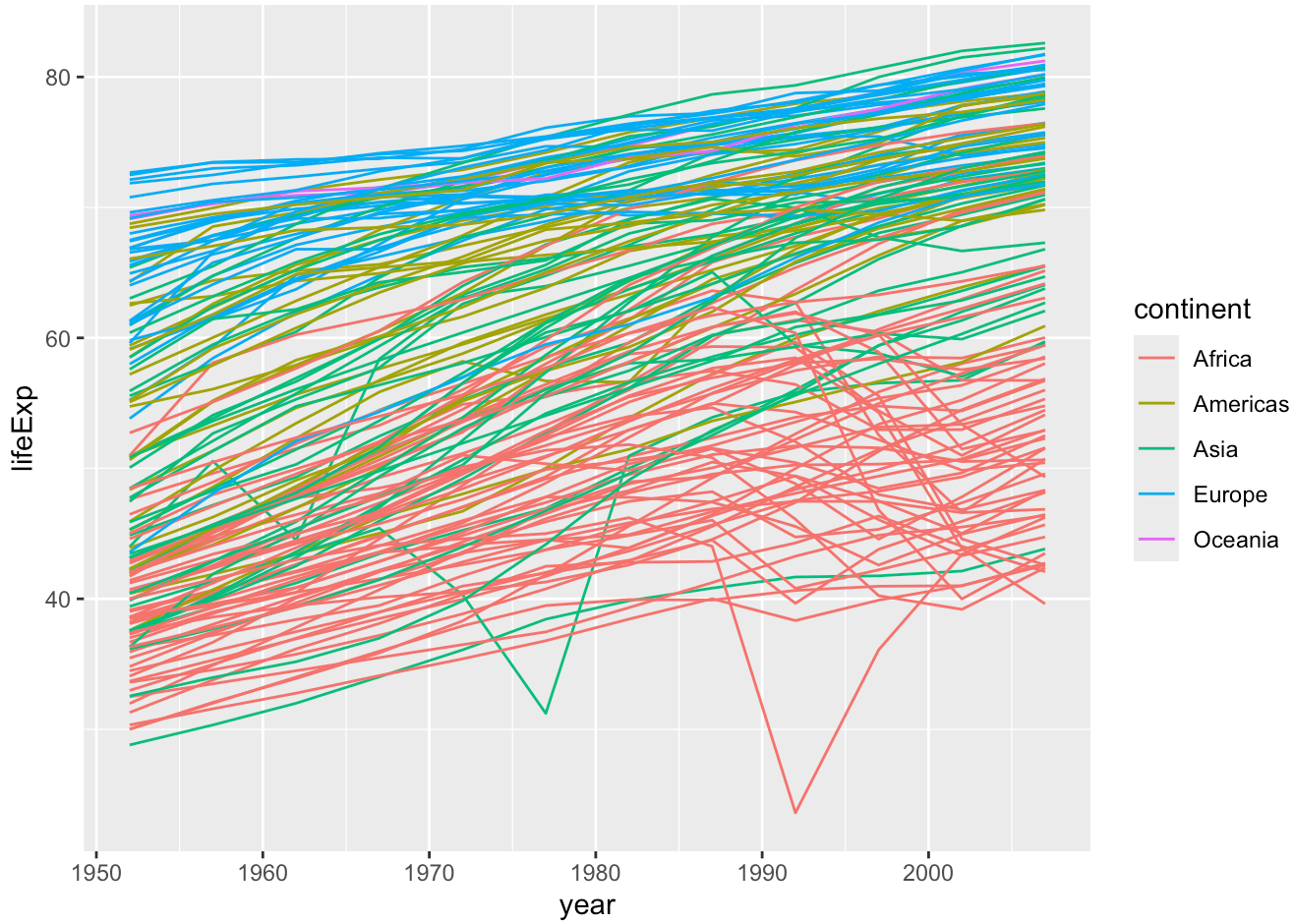
2. Line Plot

```
ggplot(data = gapminder, mapping = aes(x=year, y=lifeExp, color=continent)) +
  geom_line()
```



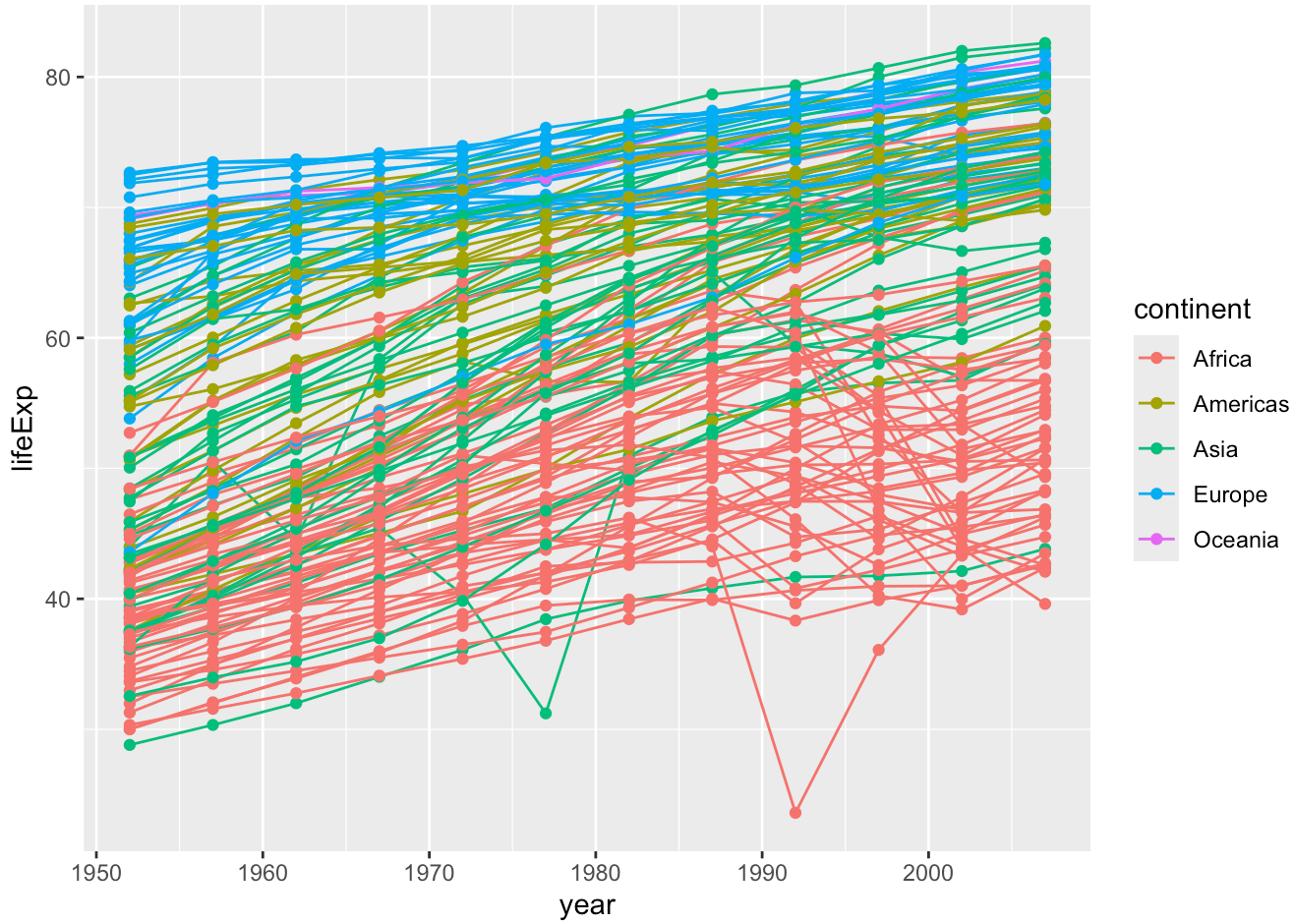
Group by country

```
ggplot(data = gapminder, mapping = aes(x=year, y=lifeExp, group=country, color=continen
t)) +
  geom_line()
```



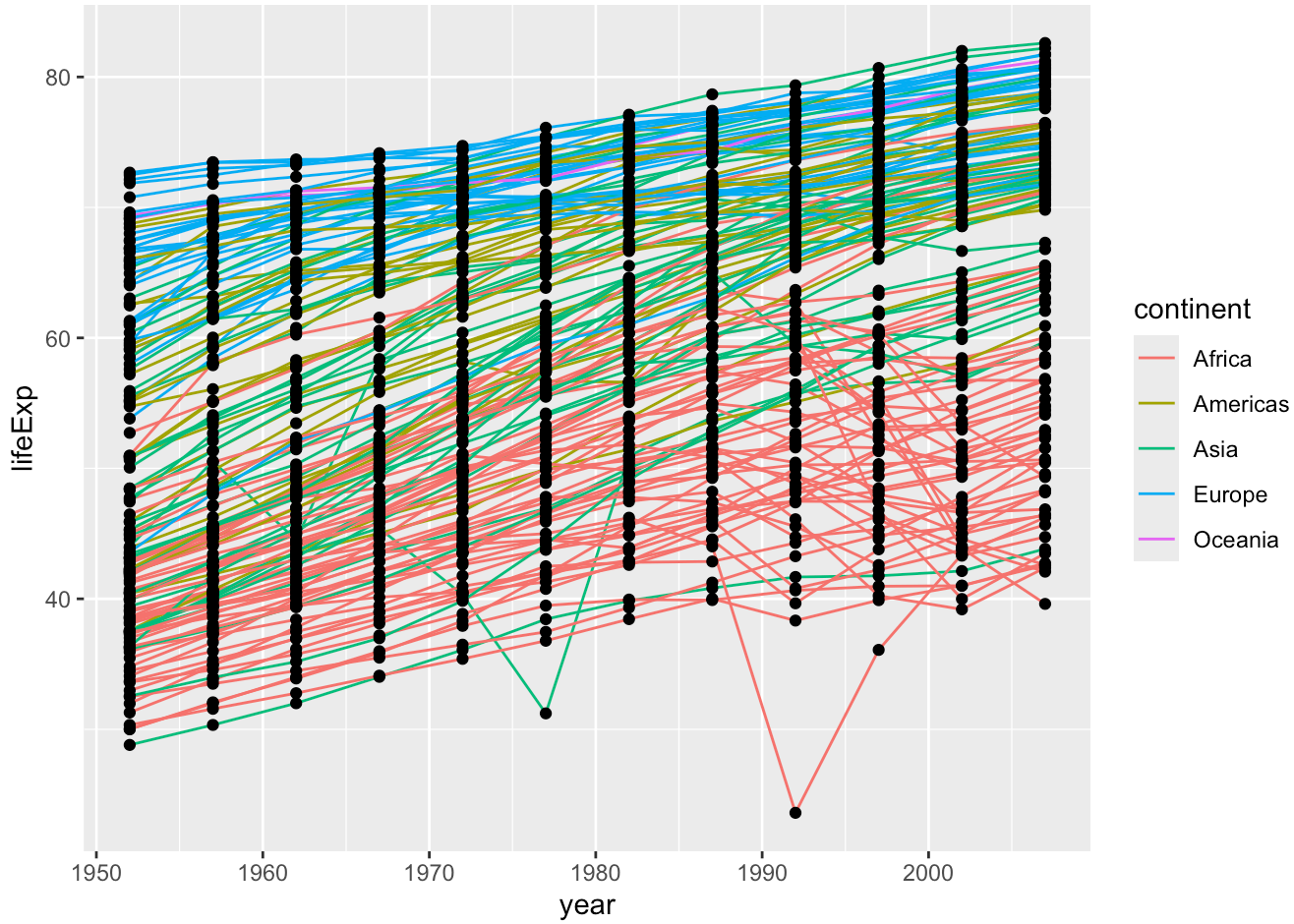
Add layer to visualize both line and scatterplot

```
ggplot(data = gapminder, mapping = aes(x=year, y=lifeExp, group=country, color=continen
t)) +
  geom_line() + geom_point()
```

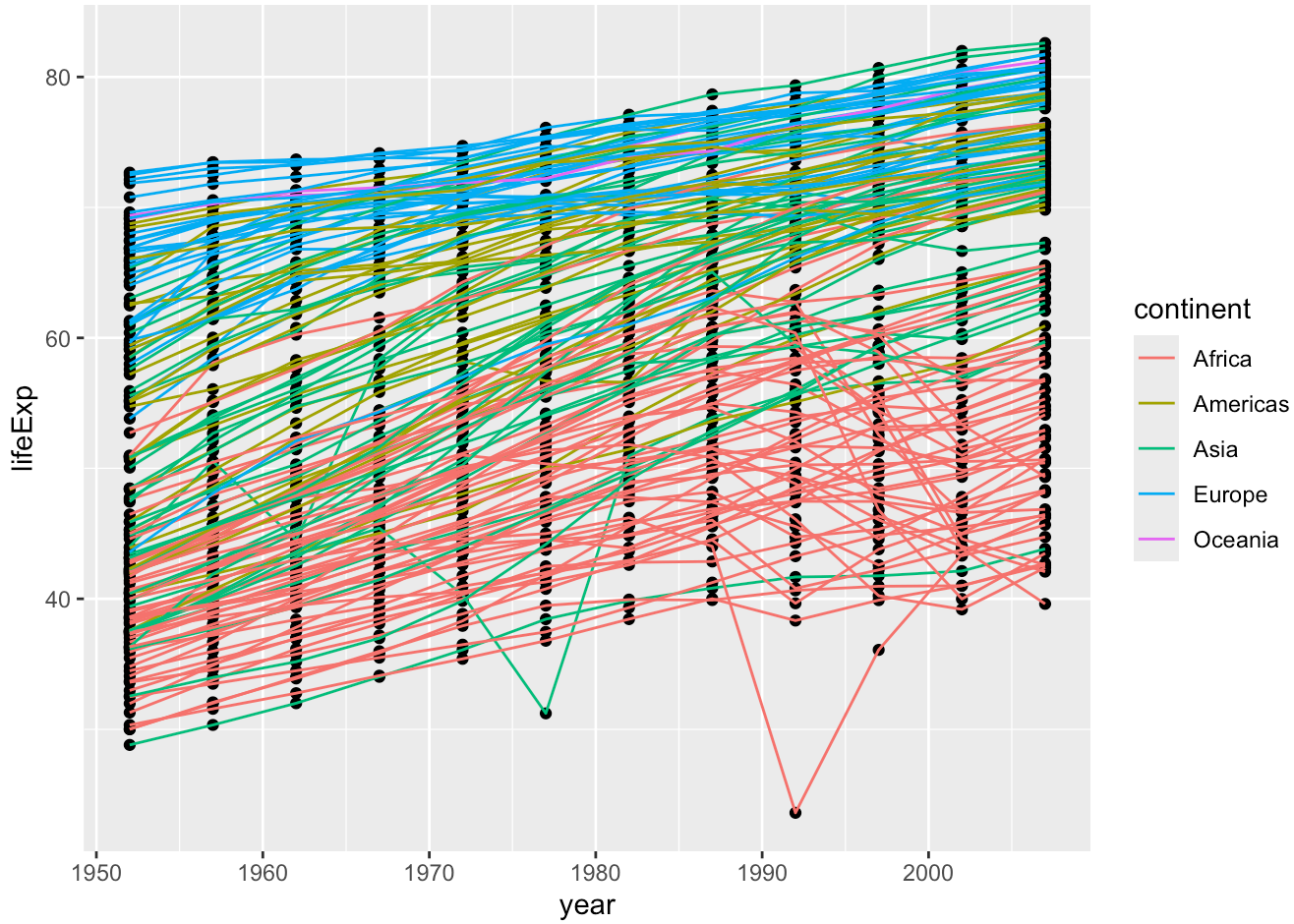


Global plot option to specific layer

```
ggplot(data = gapminder, mapping = aes(x=year, y=lifeExp, group=country)) +
  geom_line(mapping = aes(color=continent)) + geom_point()
```

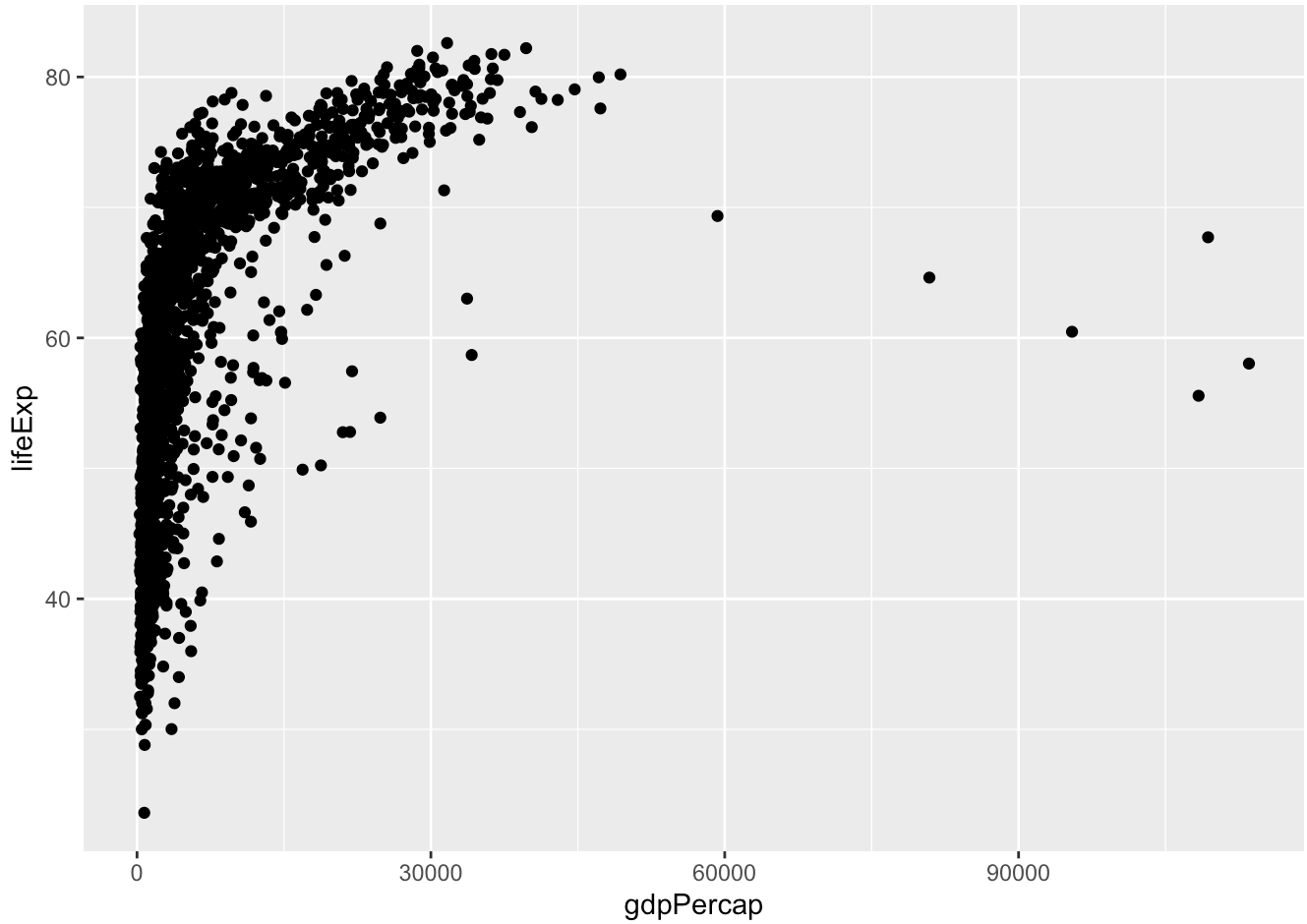



```
ggplot(data = gapminder, mapping = aes(x=year, y=lifeExp, group=country)) +
  geom_point() + geom_line(mapping = aes(color=continent))
```



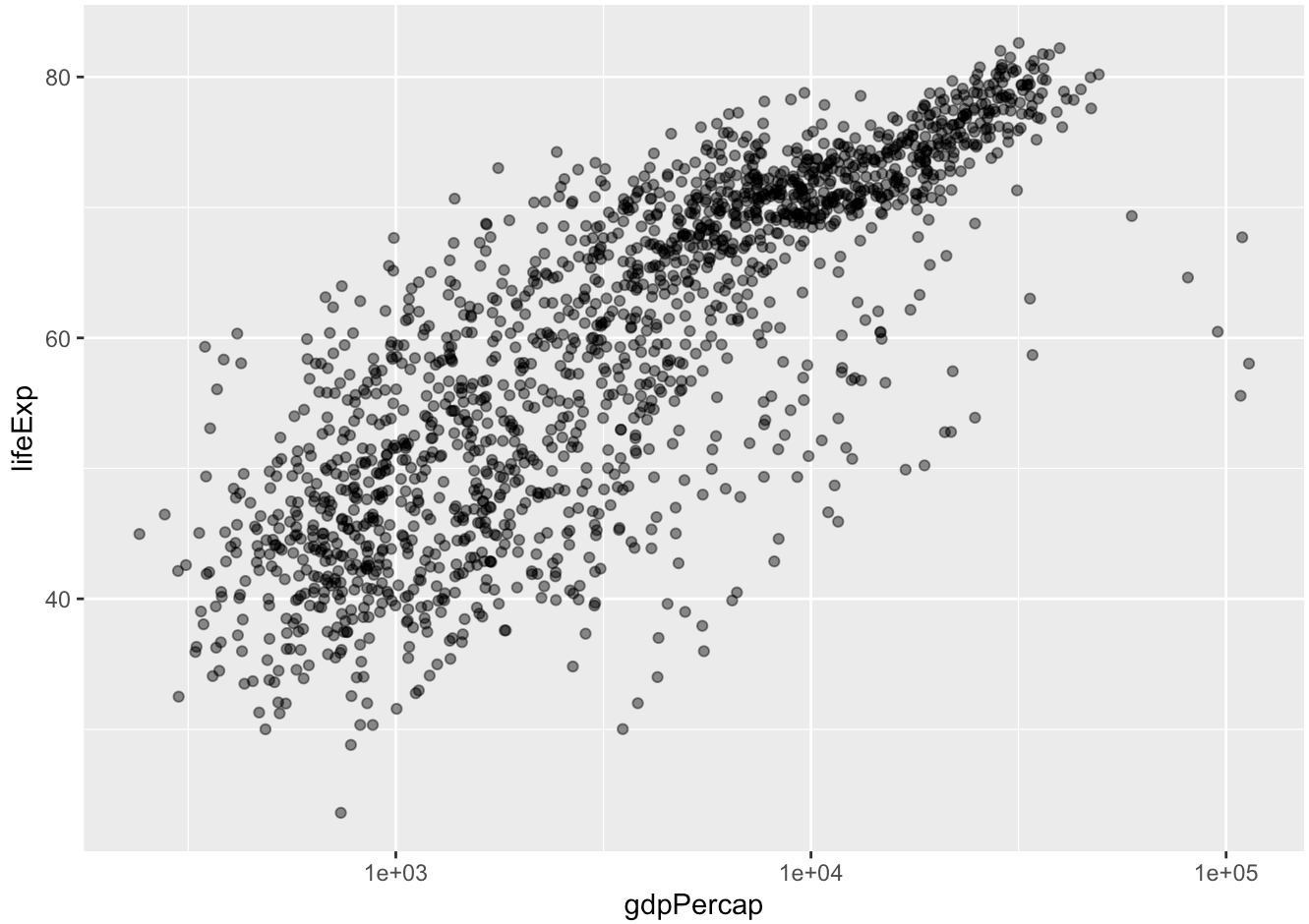
Transformations & Statistics in plots

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



Transform the x-axis scale to a log10

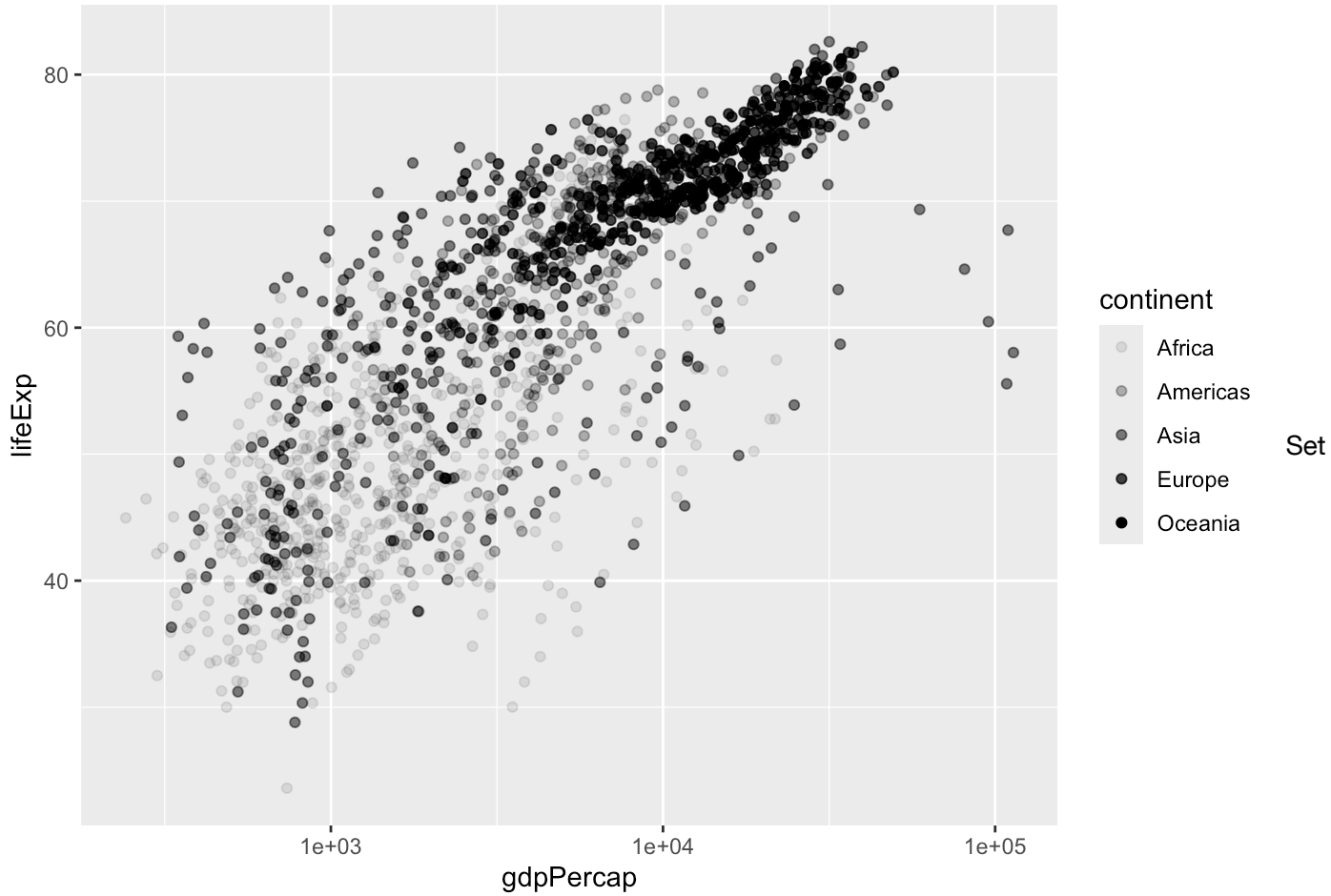
```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(alpha = 0.5) + scale_x_log10()
```



Mapping alpha to a variable

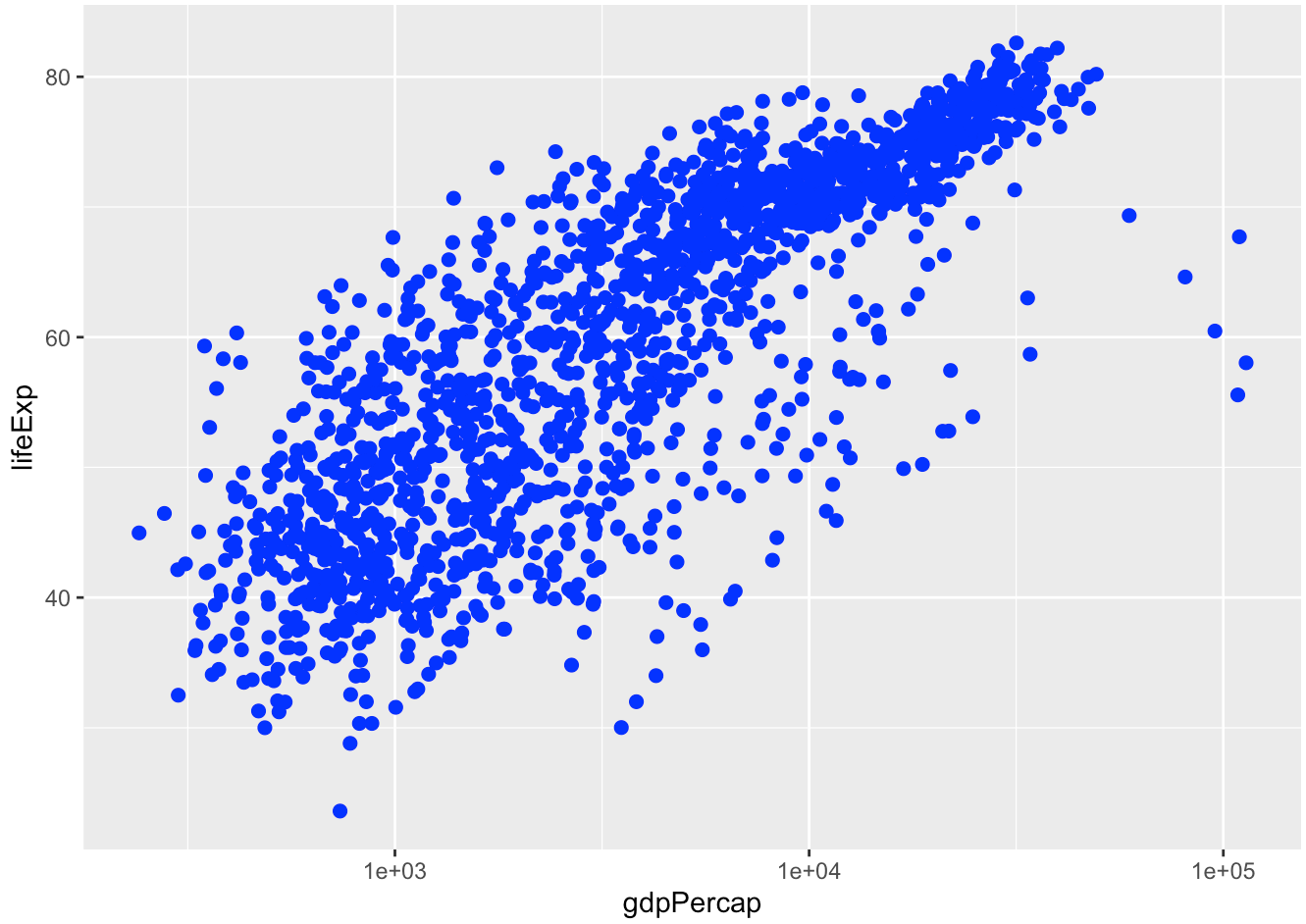
```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(mapping = aes(alpha = continent)) + scale_x_log10()
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



color and size of the points without mapping to a specific variable

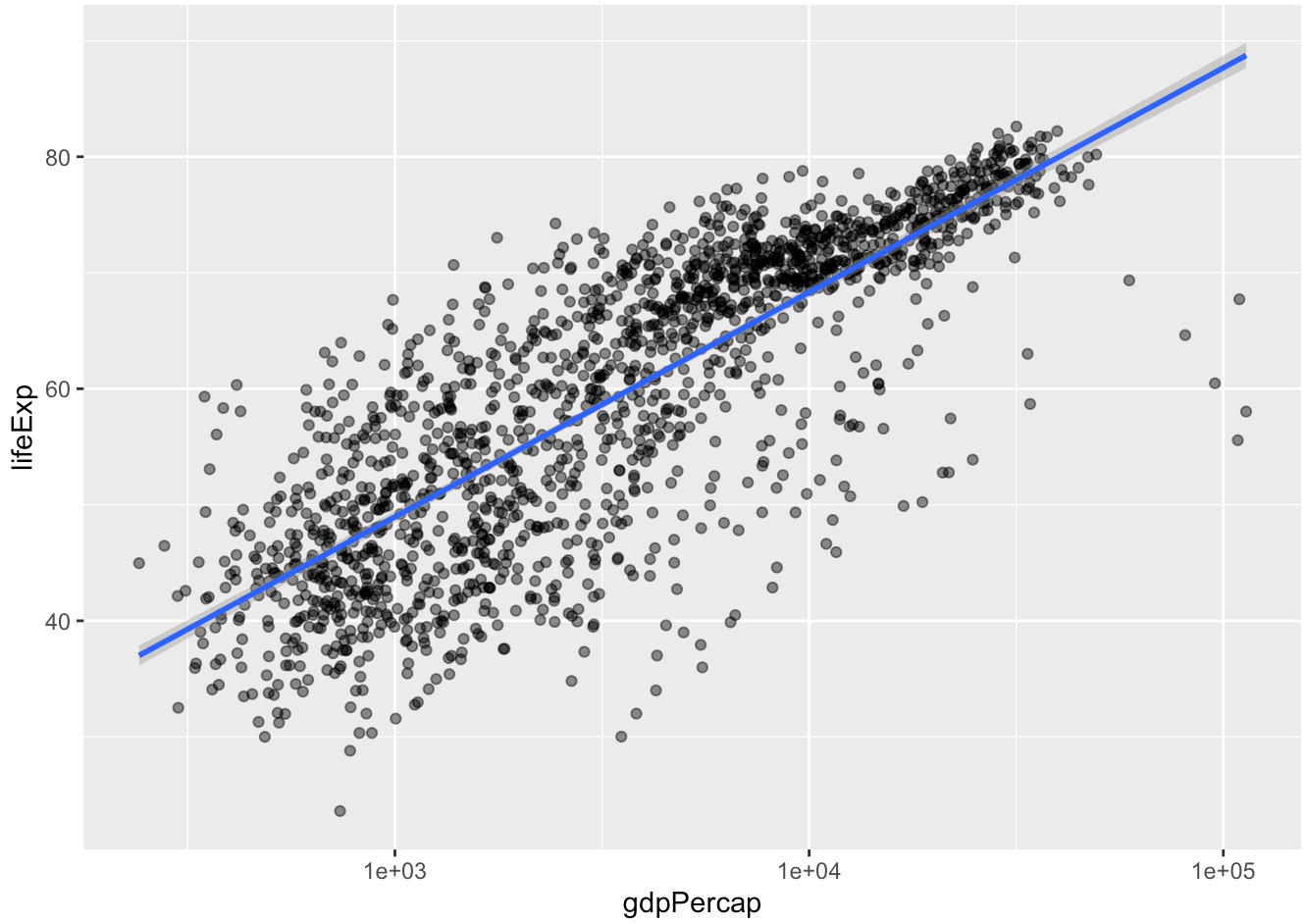
```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(size=2, color="blue") + scale_x_log10()
```



Fit a simple relationship to the data by adding another layer

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(alpha = 0.5) + scale_x_log10() + geom_smooth(method="lm")
```

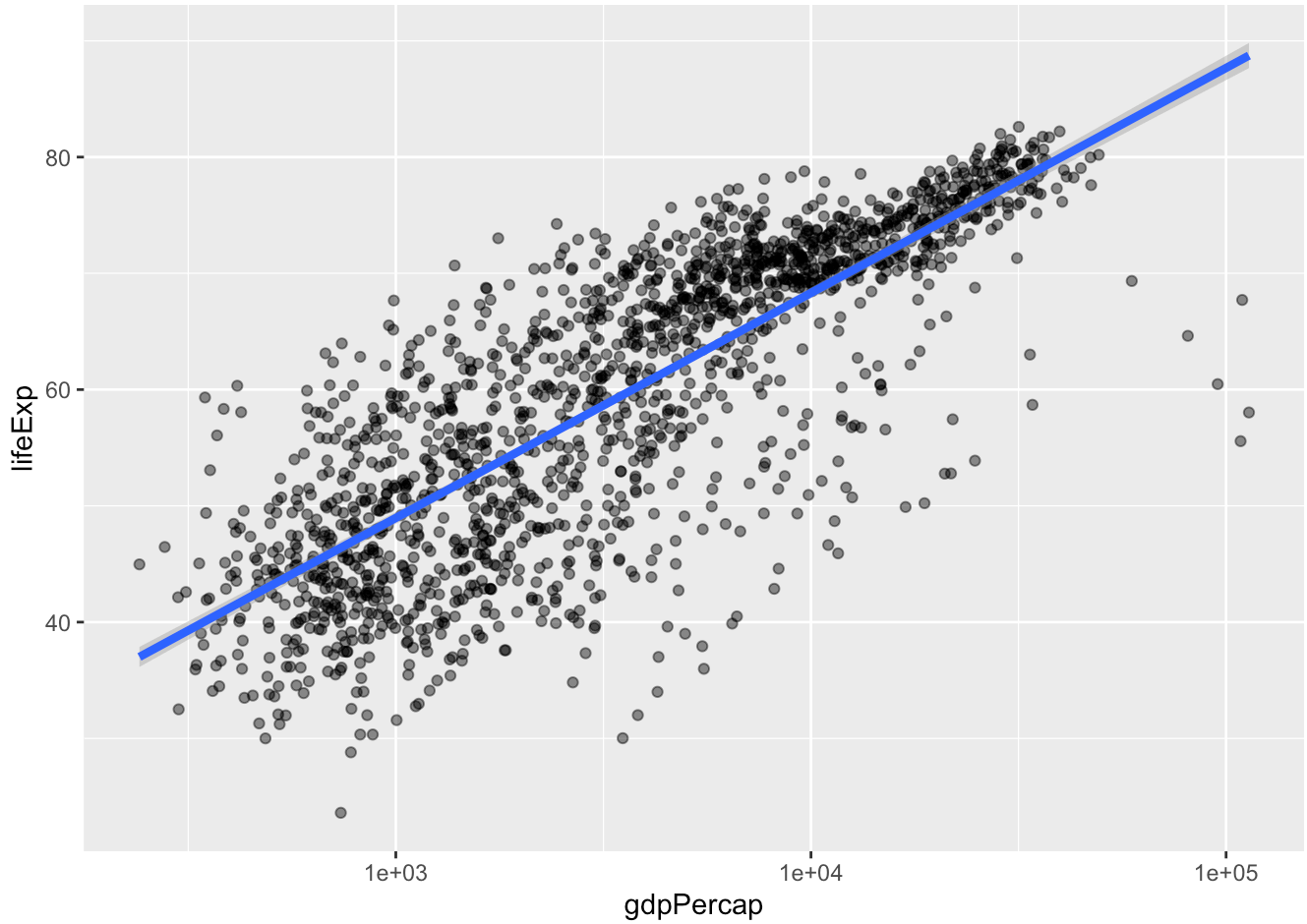
```
## `geom_smooth()` using formula = 'y ~ x'
```



Set a thicker line

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point(alpha = 0.5) + scale_x_log10() + geom_smooth(method="lm", linewidth=1.5)
```

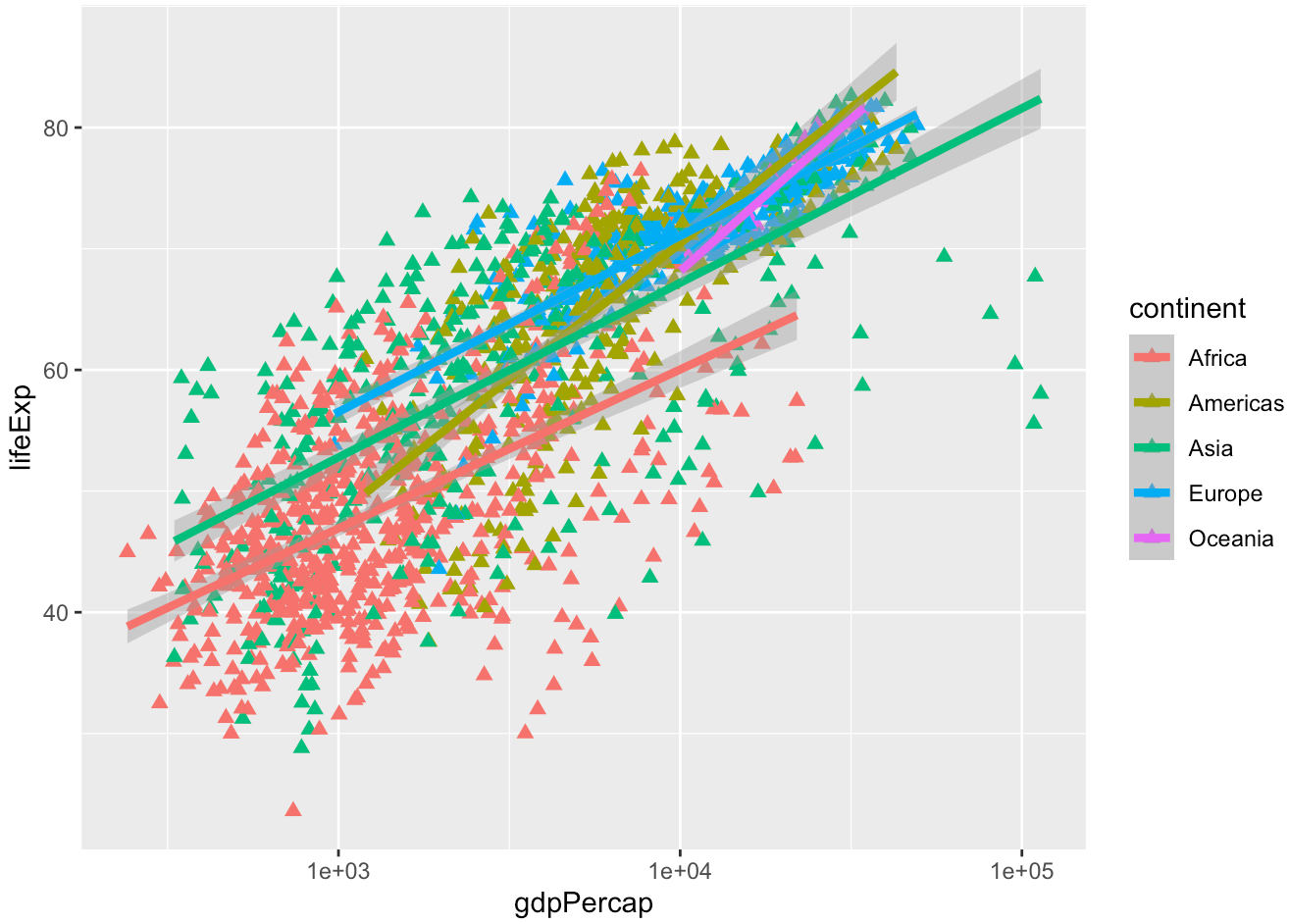
```
## `geom_smooth()` using formula = 'y ~ x'
```



Change color and shape base on the variable ‘continent’

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point(size=2, shape=17) + scale_x_log10() +
  geom_smooth(method="lm", linewidth=1.5)
```

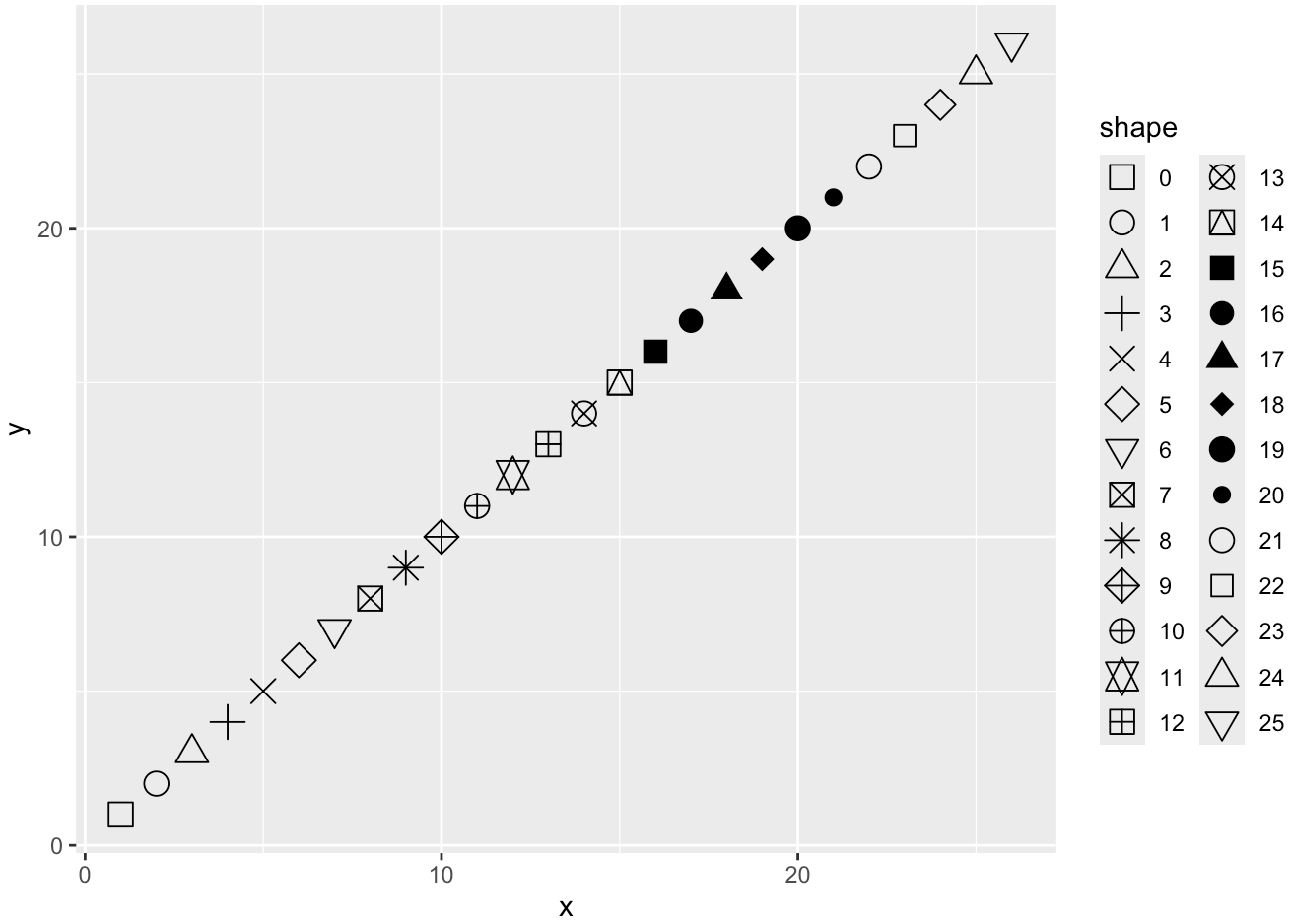
```
## `geom_smooth()` using formula = 'y ~ x'
```



Plotting characters (pch()) == point shape in ggplot

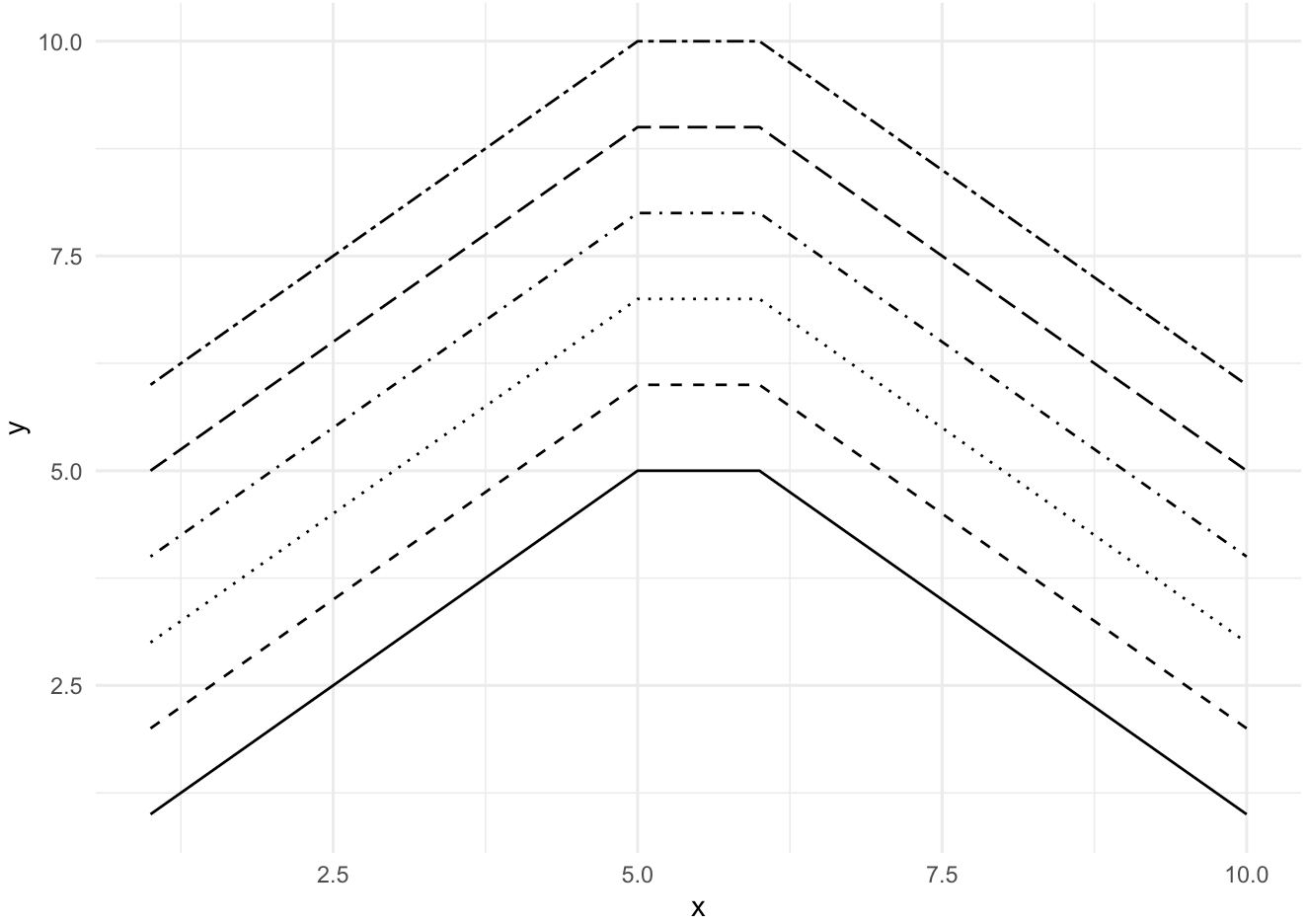
Example of ggplot2 plot with different symbols

```
df <- data.frame(x = 1:26, y = 1:26, shape = factor(0:25))

ggplot(df, aes(x, y, shape = shape)) +
  geom_point(size = 4) +
  scale_shape_manual(values = 0:25)
```



Plotting linetypes in ggplot

```
df <- data.frame(x = 1:10, y = c(1:5, 5:1))

# Plot with different line types
ggplot(df, aes(x, y)) +
  geom_line(linetype = 1) +  # Solid line
  geom_line(aes(y = y + 1), linetype = 2) +  # Dashed line
  geom_line(aes(y = y + 2), linetype = 3) +  # Dotted line
  geom_line(aes(y = y + 3), linetype = 4) +  # Dotdash line
  geom_line(aes(y = y + 4), linetype = 5) +  # Longdash line
  geom_line(aes(y = y + 5), linetype = 6) +  # Twodash line
  theme_minimal()
```



Framingham Dataset

```
df <- read_excel("framingham.xlsx")
str(df) # structure function to display the internal structure of an
```

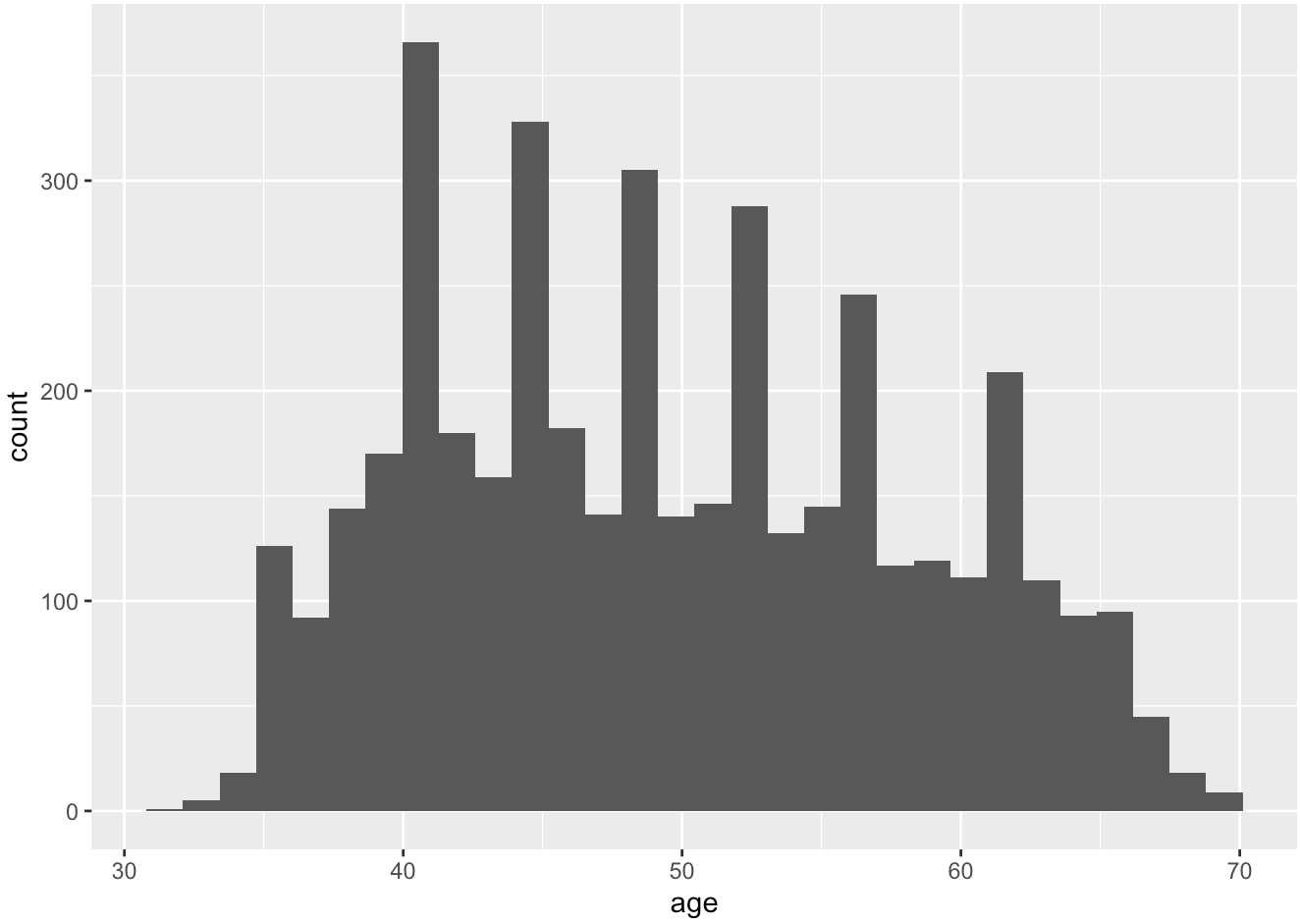
```
## tibble [4,240 × 16] (S3: tbl_df/tbl/data.frame)
##  $ male           : num [1:4240] 1 0 1 0 0 0 0 0 1 1 ...
##  $ age            : num [1:4240] 39 46 48 61 46 43 63 45 52 43 ...
##  $ education      : chr [1:4240] "4" "2" "1" "3" ...
##  $ currentSmoker  : num [1:4240] 0 0 1 1 1 0 0 1 0 1 ...
##  $ cigsPerDay     : chr [1:4240] "0" "0" "20" "30" ...
##  $ BPMeds         : chr [1:4240] "0" "0" "0" "0" ...
##  $ prevalentStroke: num [1:4240] 0 0 0 0 0 0 0 0 0 0 ...
##  $ prevalentHyp   : num [1:4240] 0 0 0 1 0 1 0 0 1 1 ...
##  $ diabetes       : num [1:4240] 0 0 0 0 0 0 0 0 0 0 ...
##  $ totChol        : chr [1:4240] "195" "250" "245" "225" ...
##  $ sysBP          : num [1:4240] 106 121 128 150 130 ...
##  $ diaBP          : num [1:4240] 70 81 80 95 84 110 71 71 89 107 ...
##  $ BMI            : chr [1:4240] "26.97" "28.73" "25.34" "28.58" ...
##  $ heartRate      : chr [1:4240] "80" "95" "75" "65" ...
##  $ glucose        : chr [1:4240] "77" "76" "70" "103" ...
##  $ TenYearCHD     : num [1:4240] 0 0 0 1 0 0 1 0 0 0 ...
```

```
# R object
```

3. Histogram

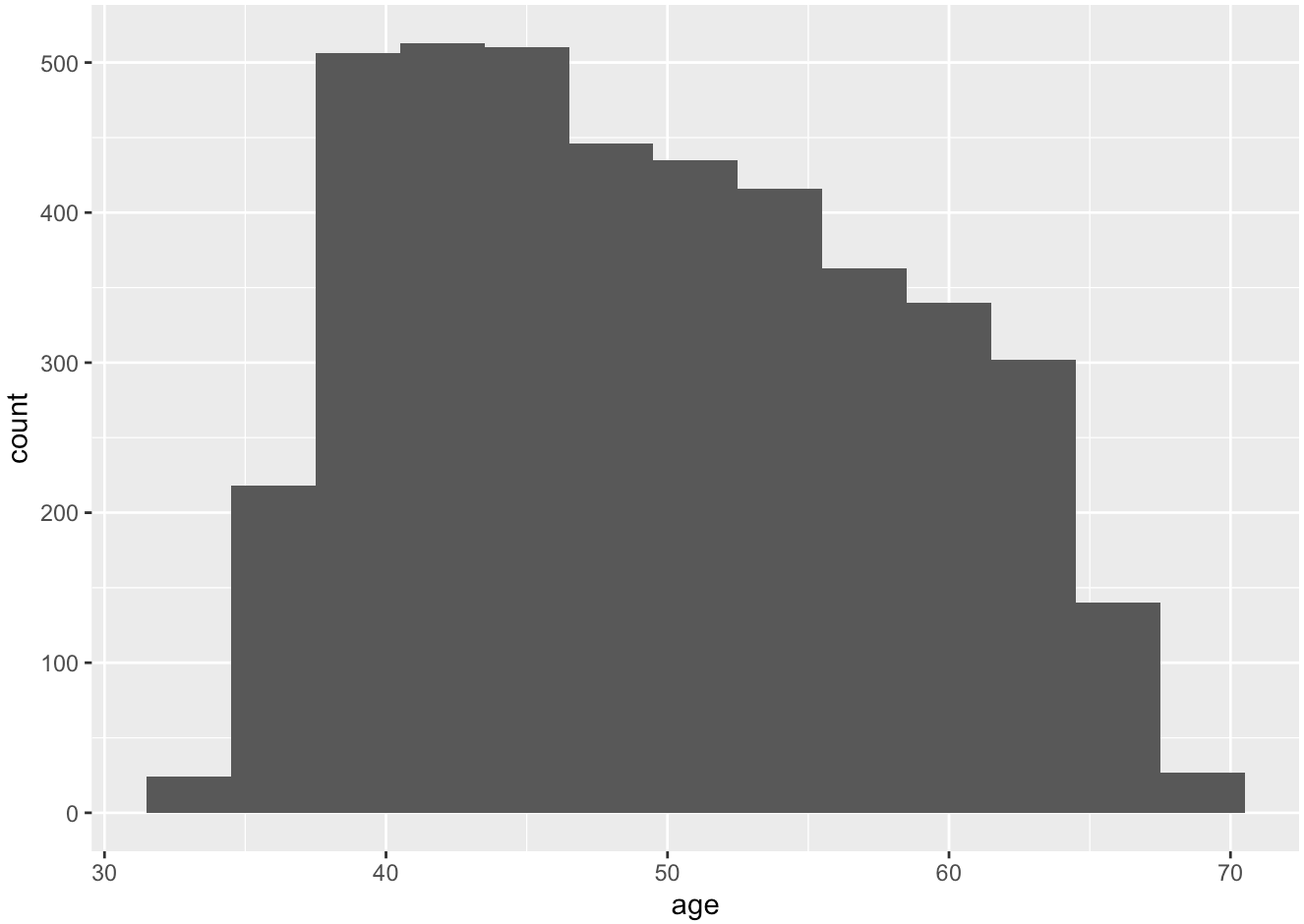
```
# message = FALSE to turn off R console messages

ggplot(df, aes(x=age)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



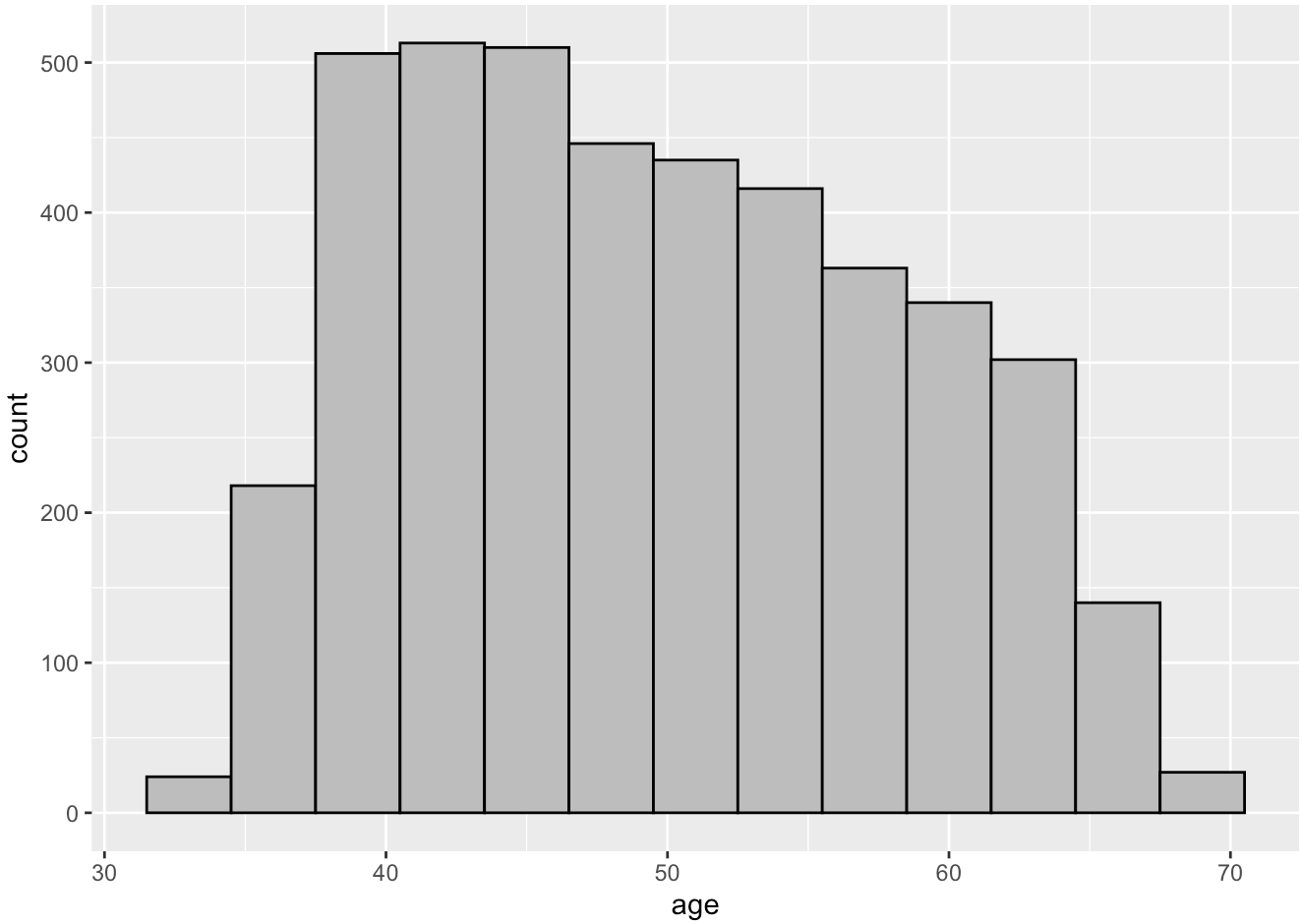
Control the bin size

```
ggplot(df, aes(x=age)) +
  geom_histogram(binwidth=3)
```



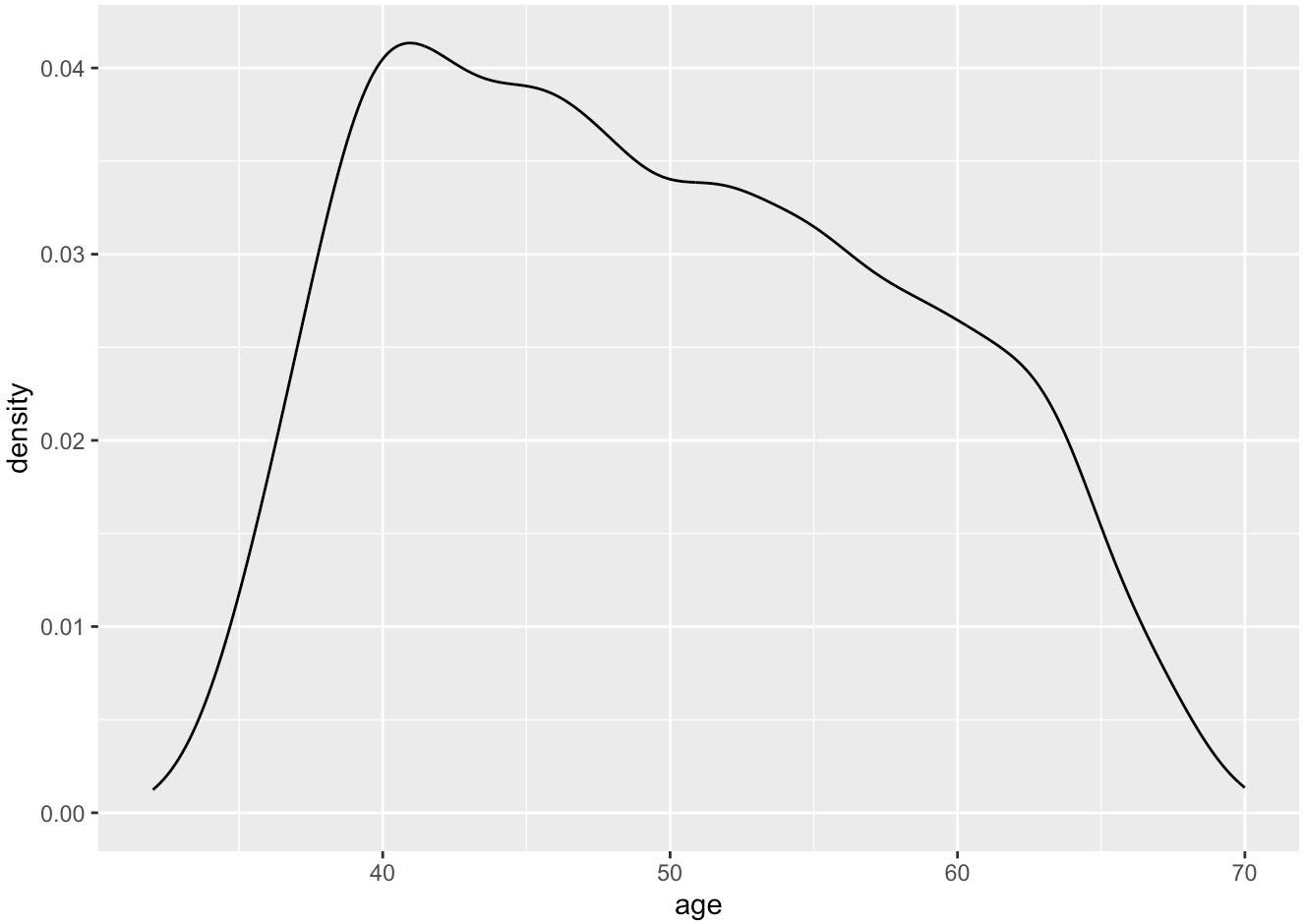
Control the color of the bins

```
ggplot(df, aes(x=age)) +
  geom_histogram(binwidth=3, fill = "gray", color = "black")
```

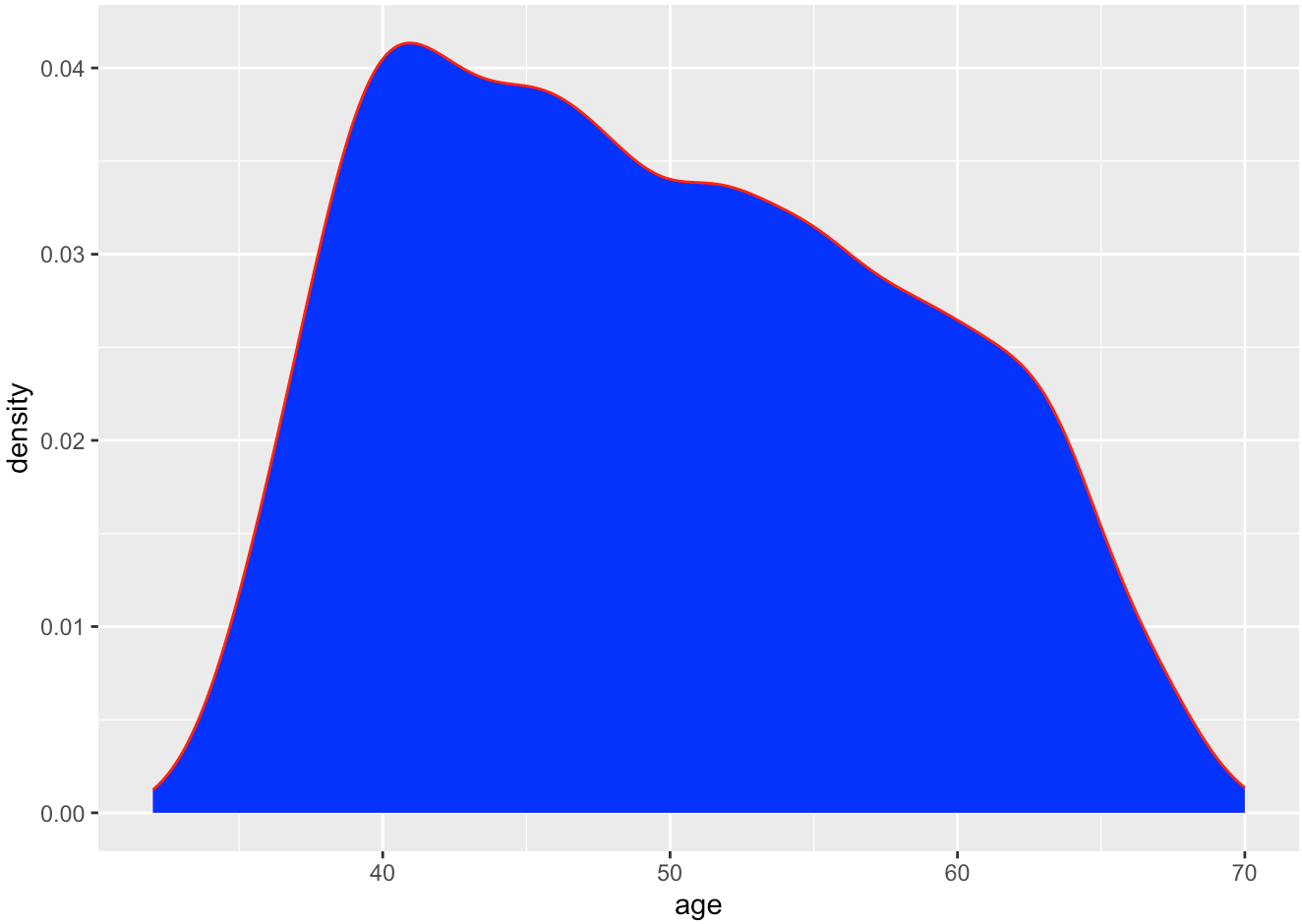


Kernal Density Plot

```
ggplot(df, aes(x=age)) +
  geom_density()
```




```
ggplot(df,aes(x=age)) +
  geom_density(fill="blue", color="red")
```



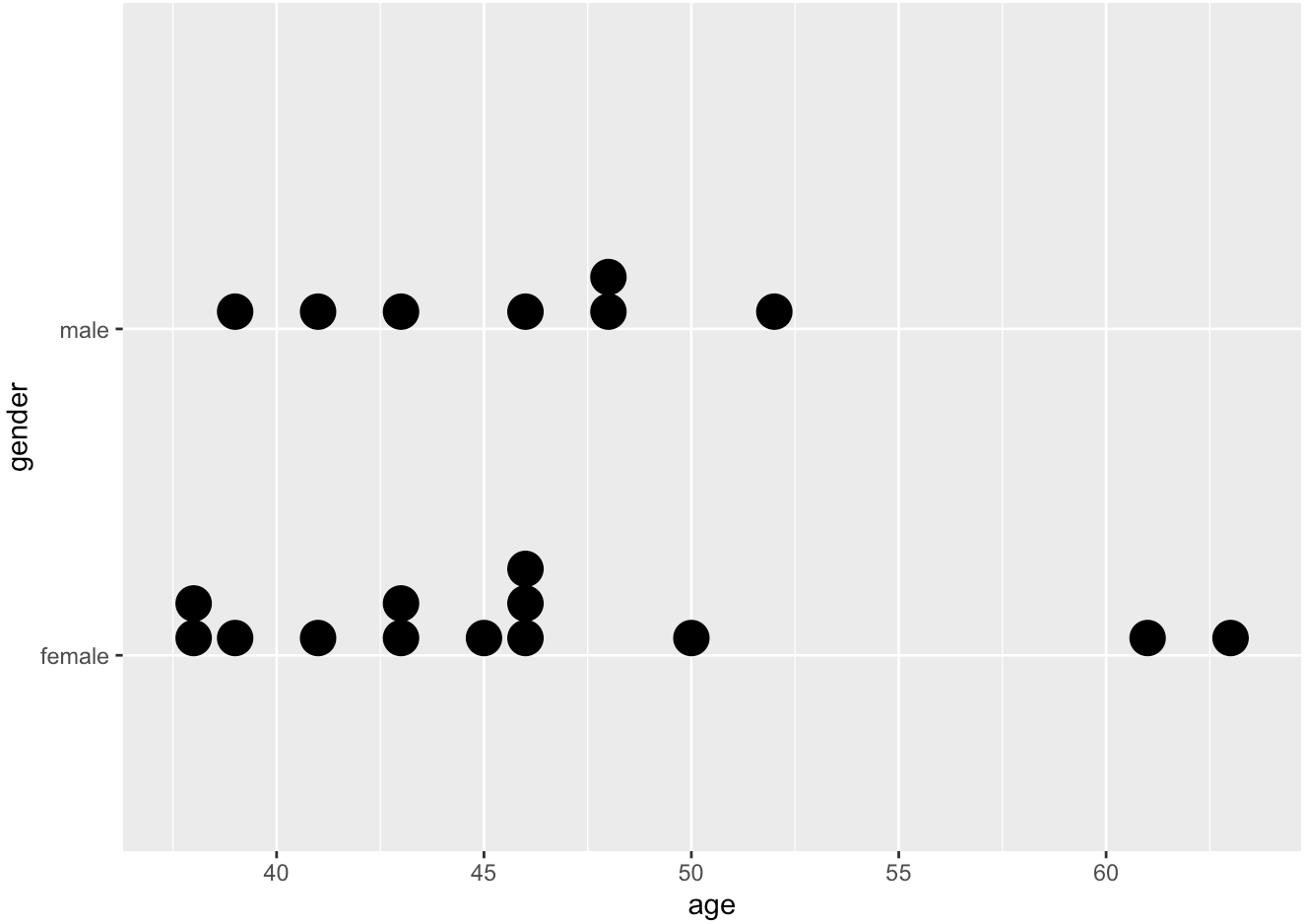
Read in the Framingham subset dataset

```
df_sub <- read.csv("framingham_sub.csv")
```

4. Dotplots

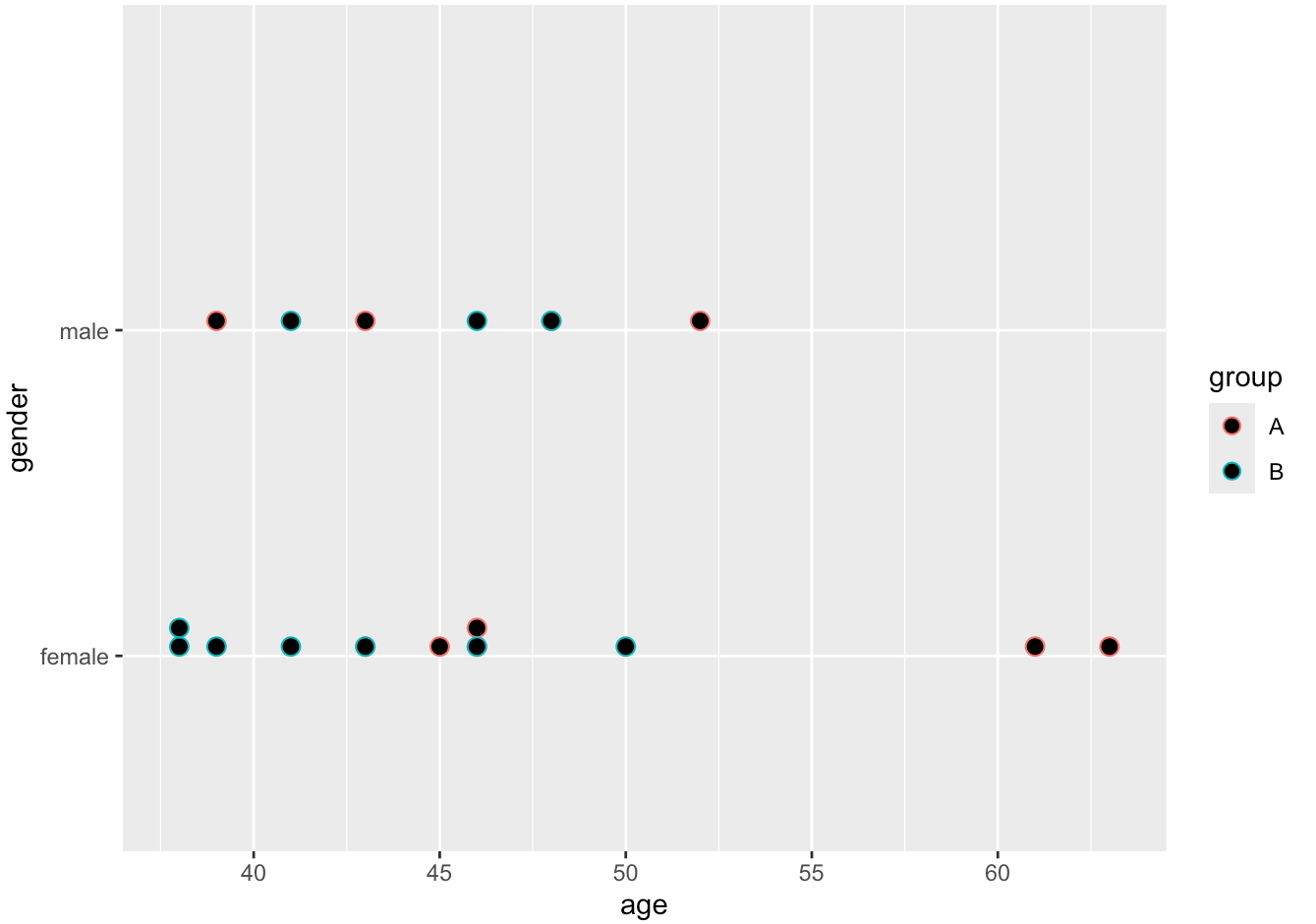
```
ggplot(df_sub, aes(x= age, y = gender)) +
  geom_dotplot()
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.
```



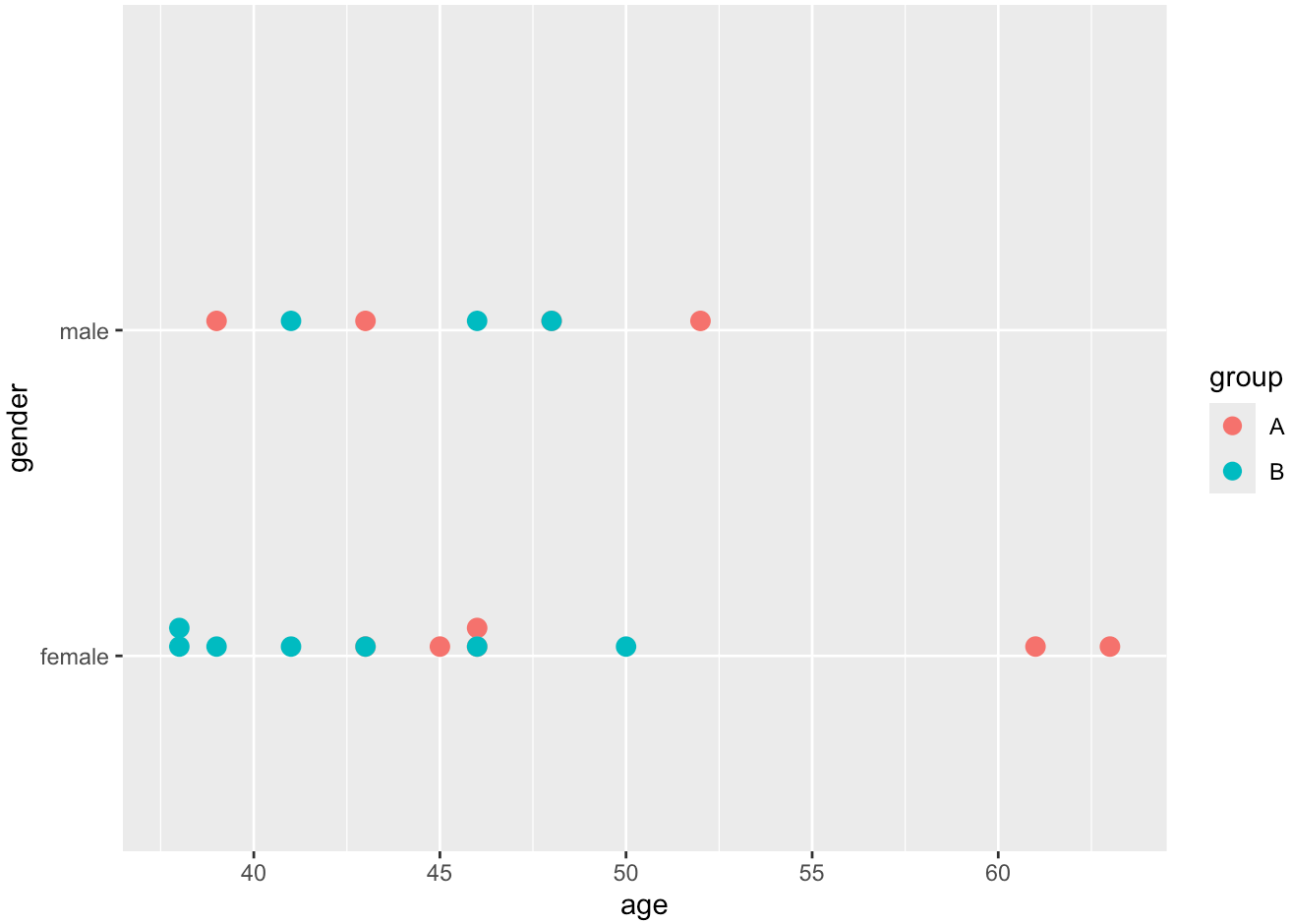
Change bin size

```
ggplot(df_sub, aes(x= age, y = gender, color = group)) +
  geom_dotplot(binwidth = 0.5)
```



Approach 1: - mapping aes() within ggplot (global plot option)

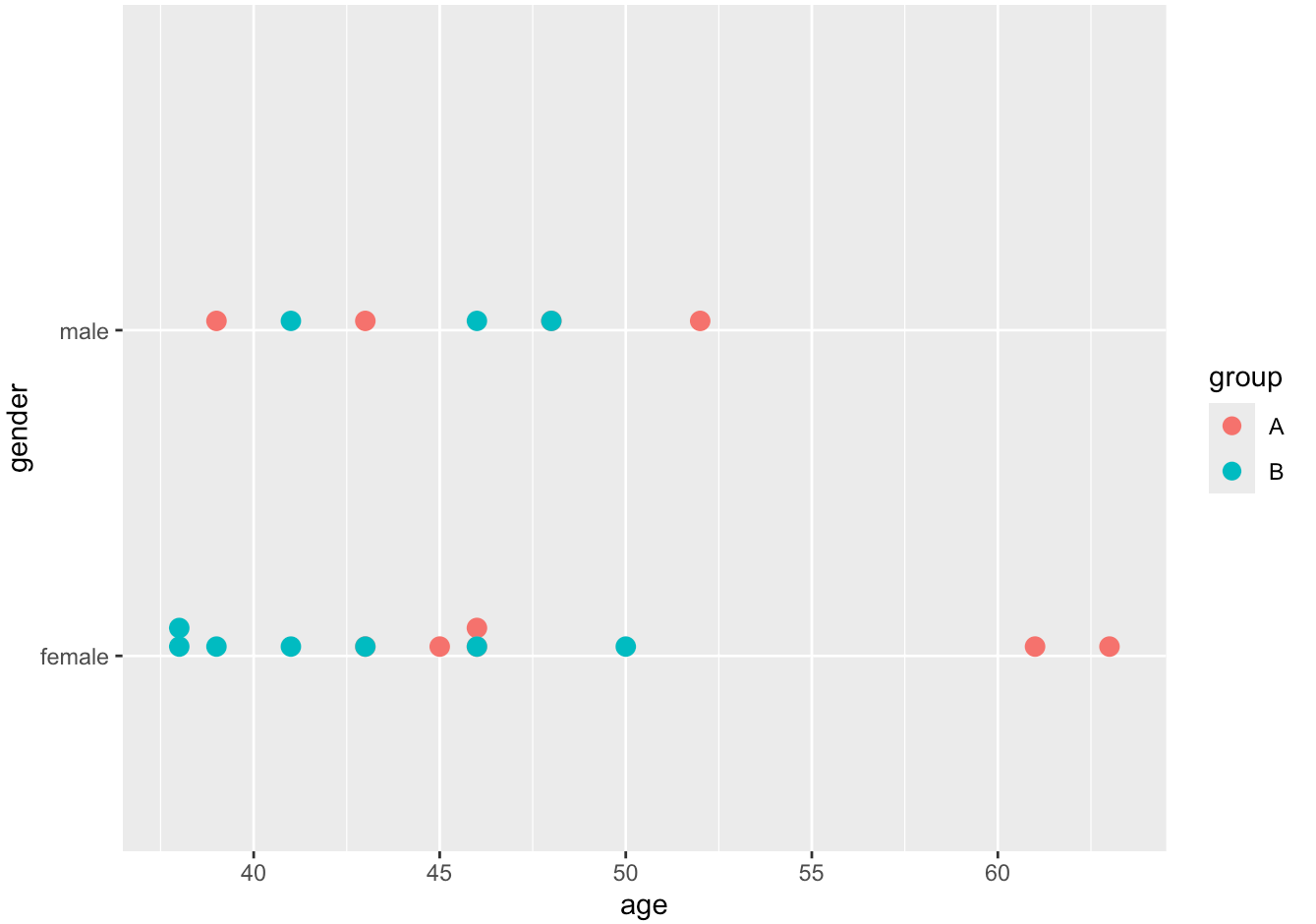
```
ggplot(df_sub, aes(x= age, y = gender, color = group, fill = group)) +
  geom_dotplot(binwidth = 0.5)
```



Approach 2:

- mapping aes() within geom_dotplot

```
ggplot(df_sub, aes(x= age, y = gender)) +
  geom_dotplot(aes(color = group, fill = group),binwidth = 0.5)
```



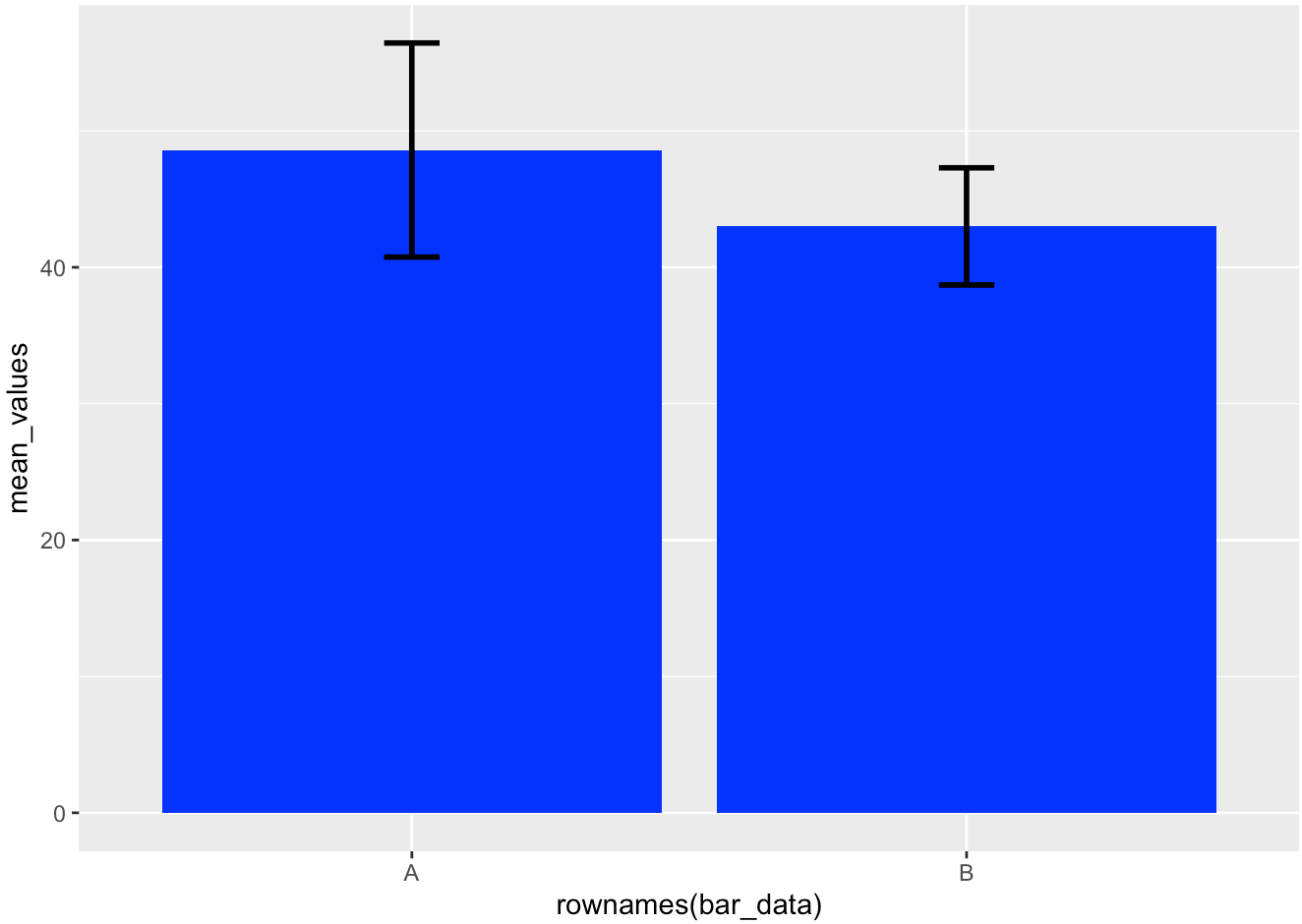
5. Barplots

Needs one numeric and one categorical variable

Simple barplot

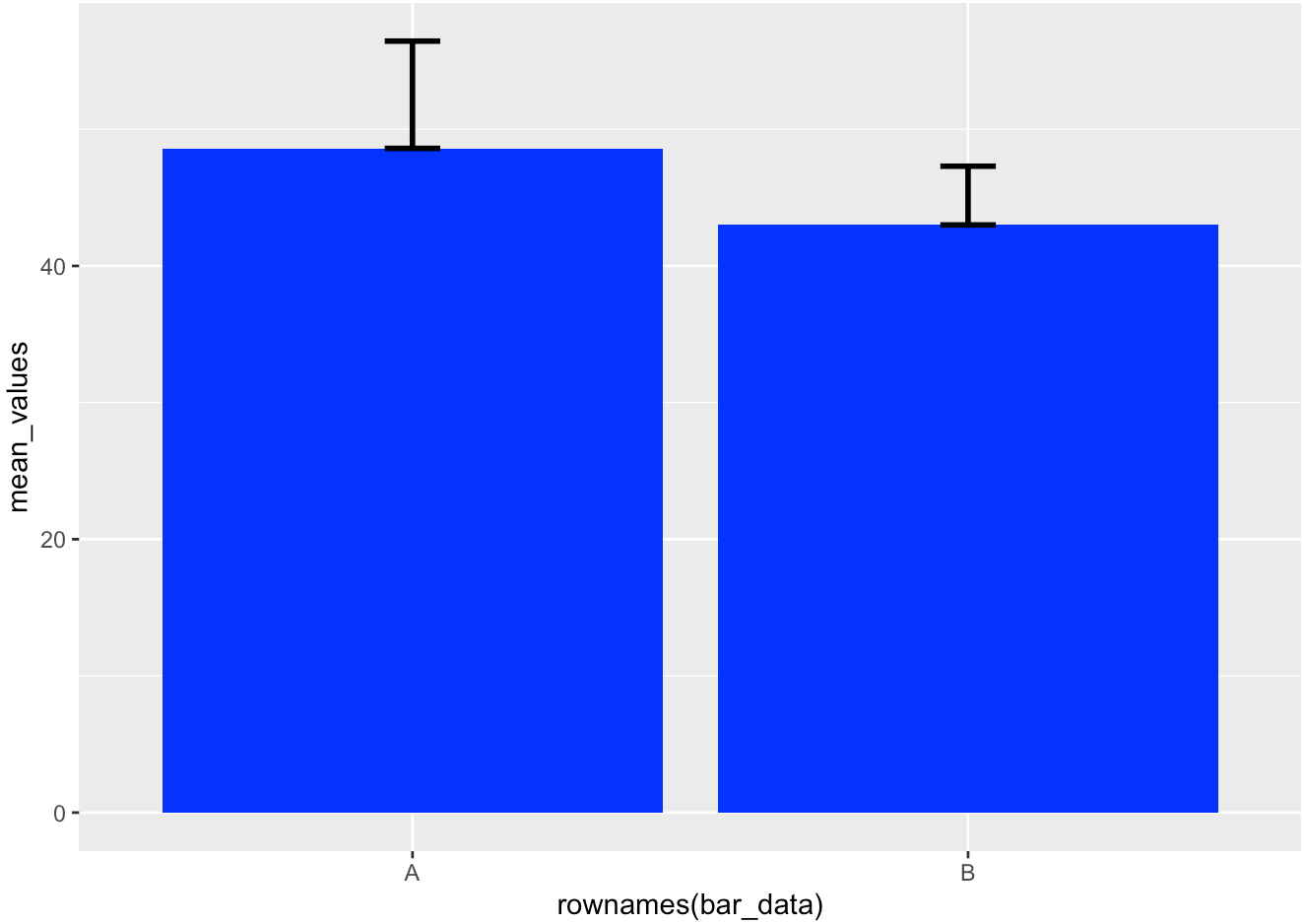
```
# compute the summary data for bar graph
bar_data = data.frame(
  mean_values = tapply(df_sub$age, df_sub$group, mean),
  sd_values = tapply(df_sub$age, df_sub$group, sd),
  sem_values = tapply(df_sub$age, df_sub$group, sd)/sqrt(nrow(df_sub)))

# Barplot with SD
ggplot(bar_data, aes(x=rownames(bar_data), y=mean_values)) +
  geom_bar(stat="identity", fill="blue") +
  geom_errorbar(aes(x=rownames(bar_data), ymin = mean_values - sd_values,
                    ymax=mean_values + sd_values),
                width=0.1, size = 1)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



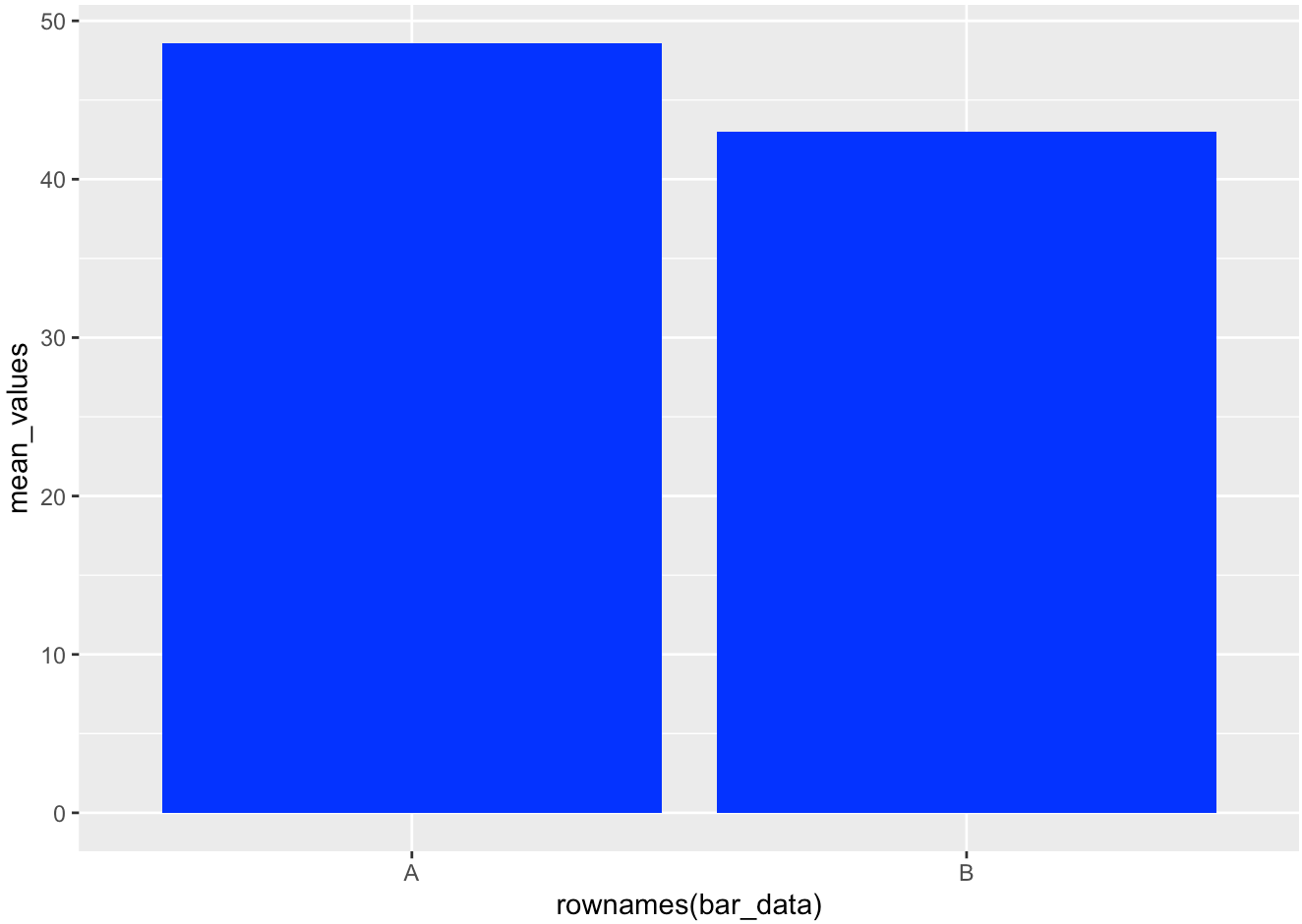
Barplot - one direction for error bars

```
ggplot(bar_data, aes(x=rownames(bar_data), y=mean_values)) +
  geom_bar(stat="identity", fill="blue") +
  geom_errorbar(aes(x=rownames(bar_data), ymin = mean_values,
                    ymax=mean_values + sd_values),
                width=0.1, size = 1)
```



Adding to a ggplot plot

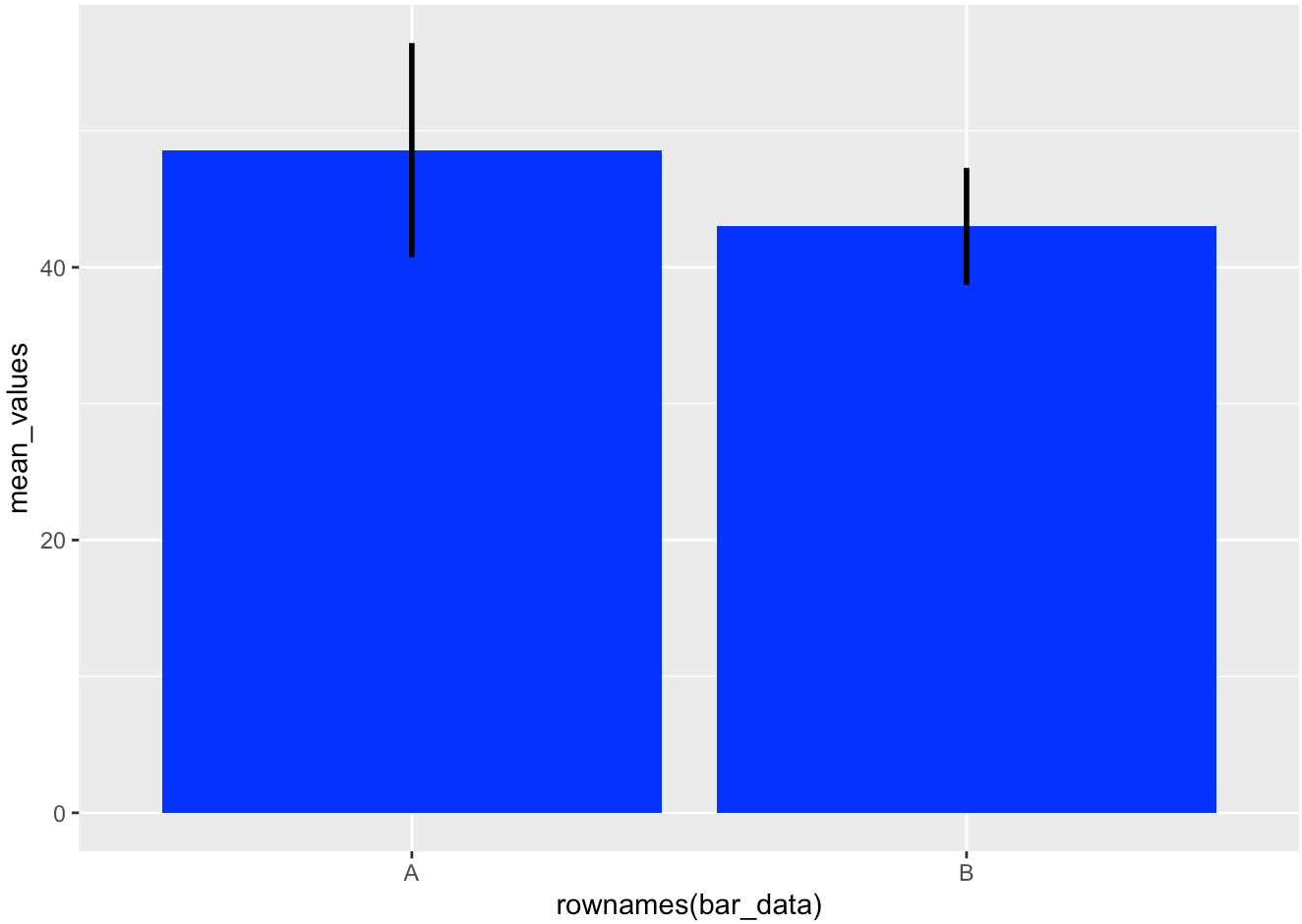
```
p = ggplot(bar_data, aes(x=rownames(bar_data), y=mean_values)) +
  geom_bar(stat="identity", fill="blue")
p
```



Other error bar types:

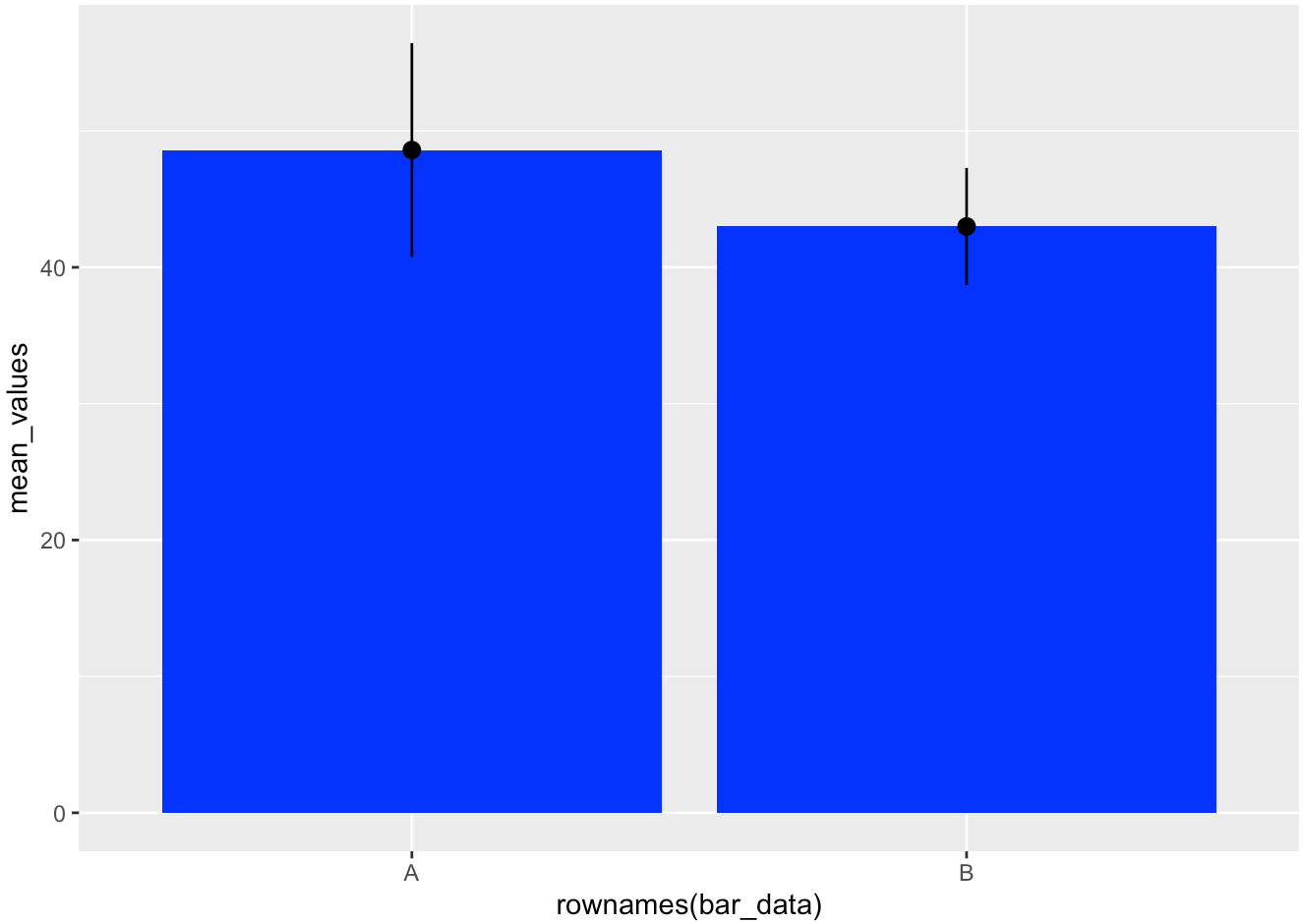
linerange

```
p + geom_linerange(aes(x=rownames(bar_data), ymin = mean_values - sd_values,
                       ymax=mean_values + sd_values), size = 1)
```



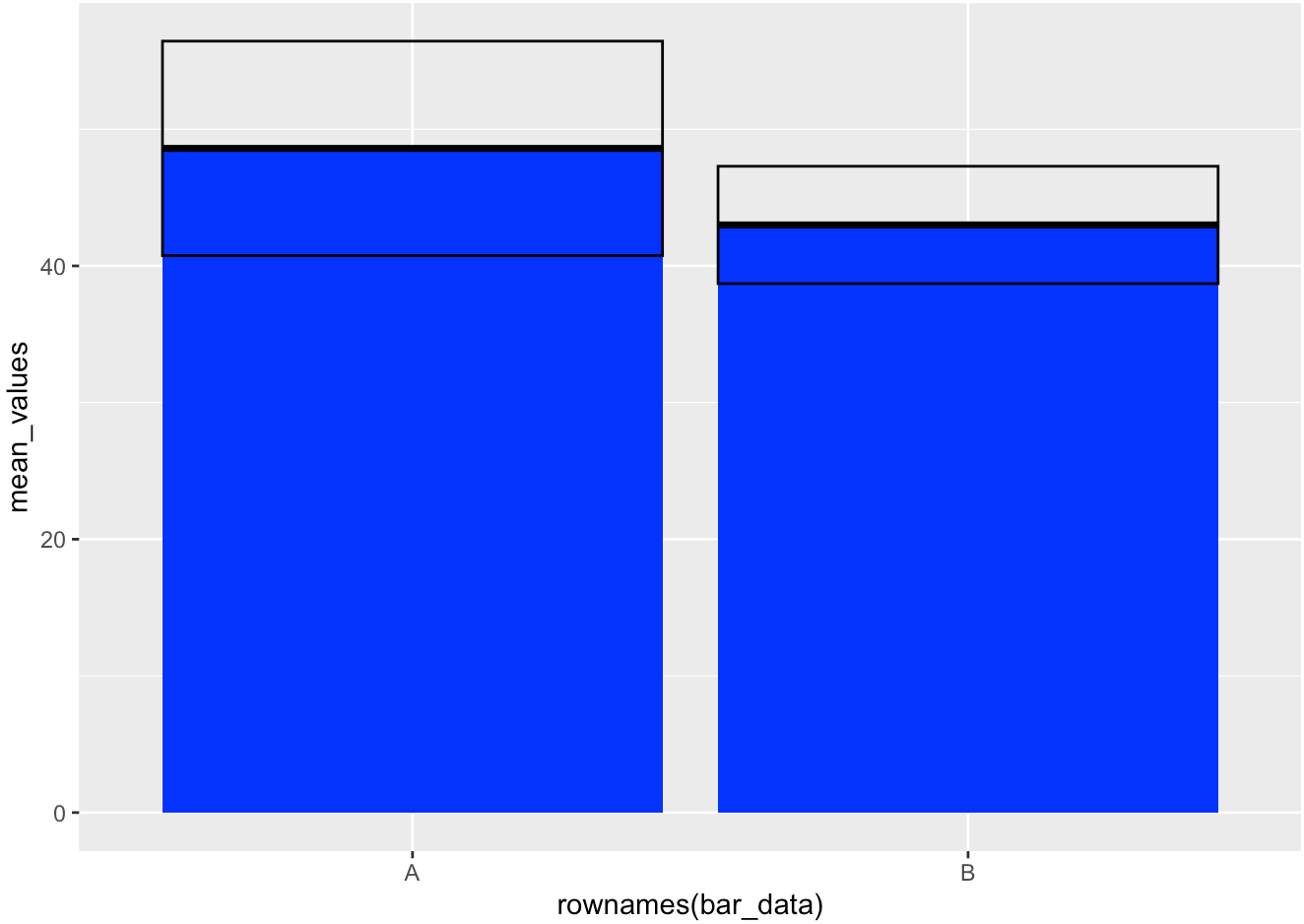
pointrange

```
p + geom_pointrange(aes(ymin = mean_values - sd_values,
                        ymax=mean_values + sd_values))
```



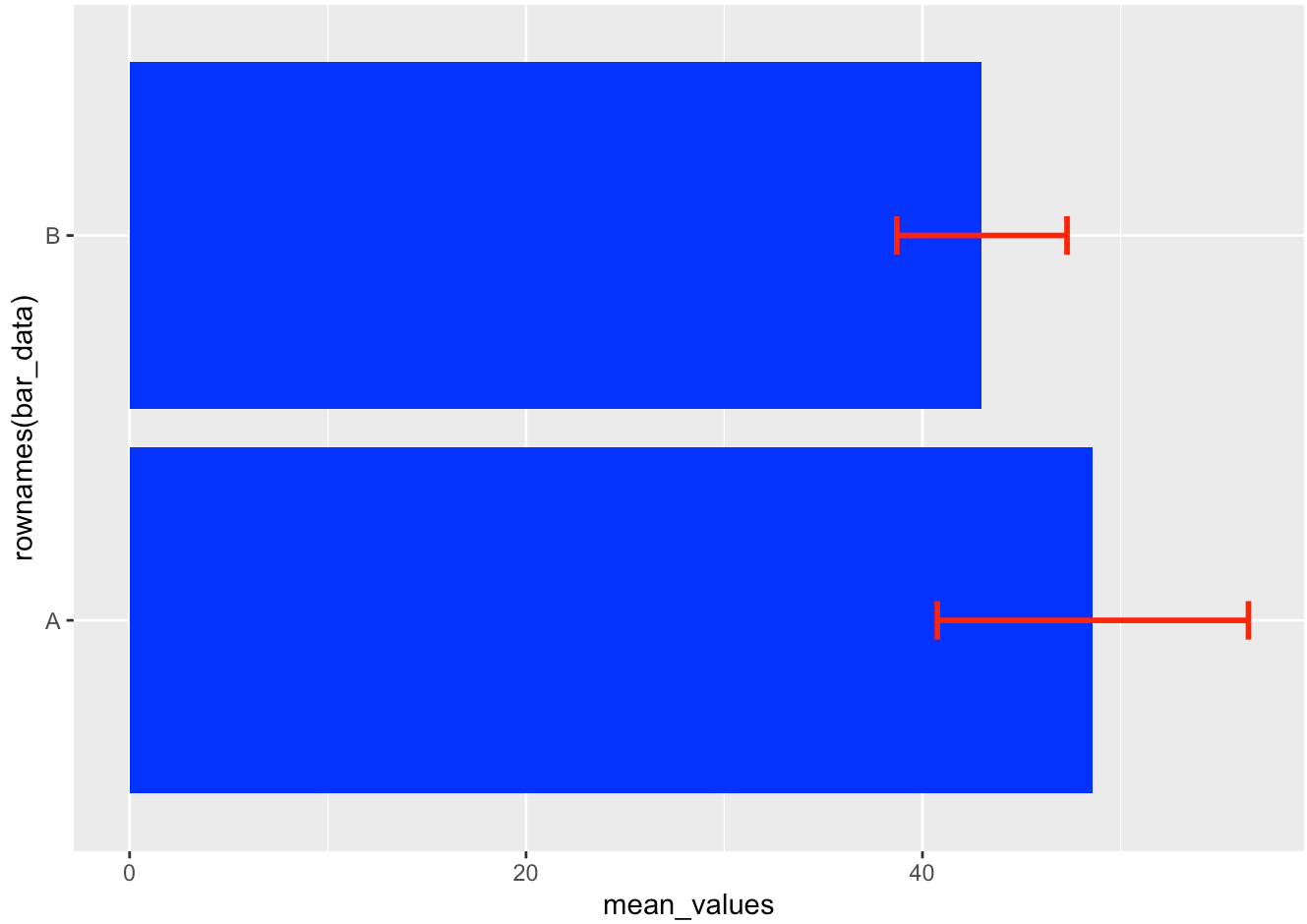
crossbar

```
p + geom_crossbar(aes(ymin = mean_values - sd_values,
                      ymax=mean_values + sd_values))
```



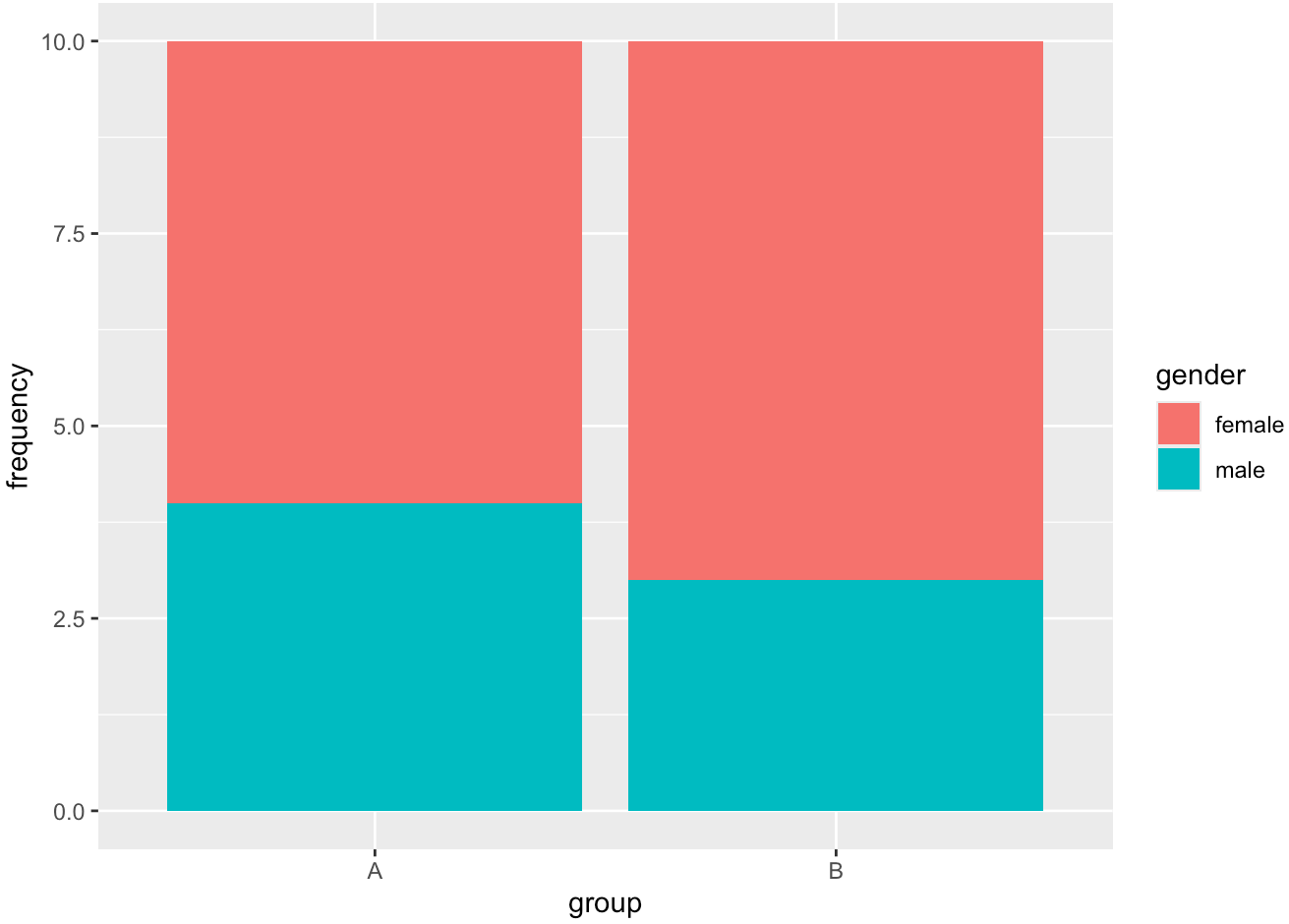
Horizonal barplot with errow bars

```
p + geom_errorbar(aes(ymin = mean_values - sd_values,
                      ymax=mean_values + sd_values),
                  width = 0.1, color = "red", size = 1) +
  coord_flip()
```



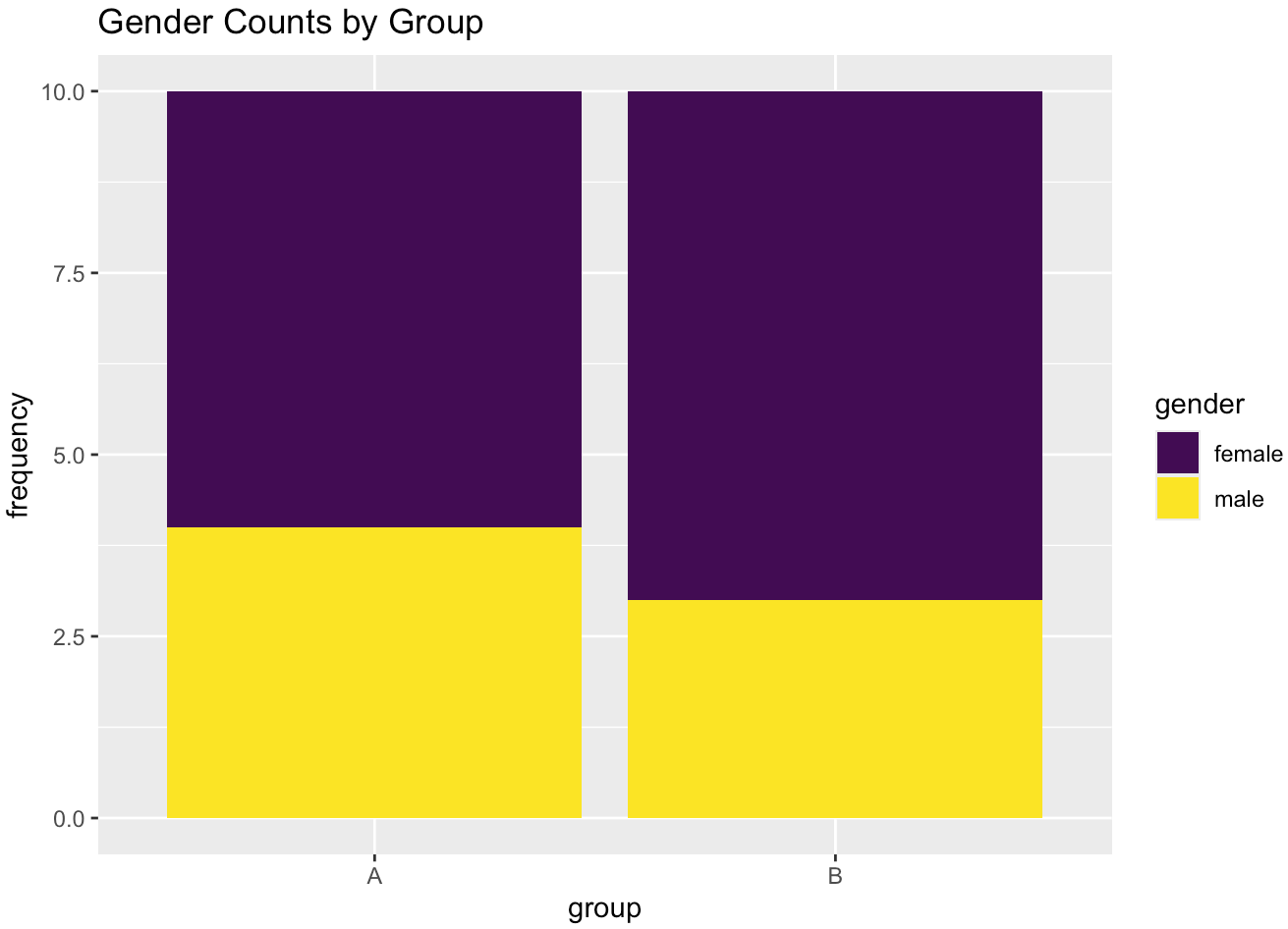
Stacked barplot

```
# get counts of gender data
gender_counts = table(df_sub$gender,df_sub$group)

# create gender counts dataframe
gender_counts = as.data.frame(gender_counts)
colnames(gender_counts) = c("gender", "group", "frequency")

ggplot(gender_counts, aes(x=group, y=frequency, fill = gender)) +
  geom_bar(position="stack", stat="identity")
```



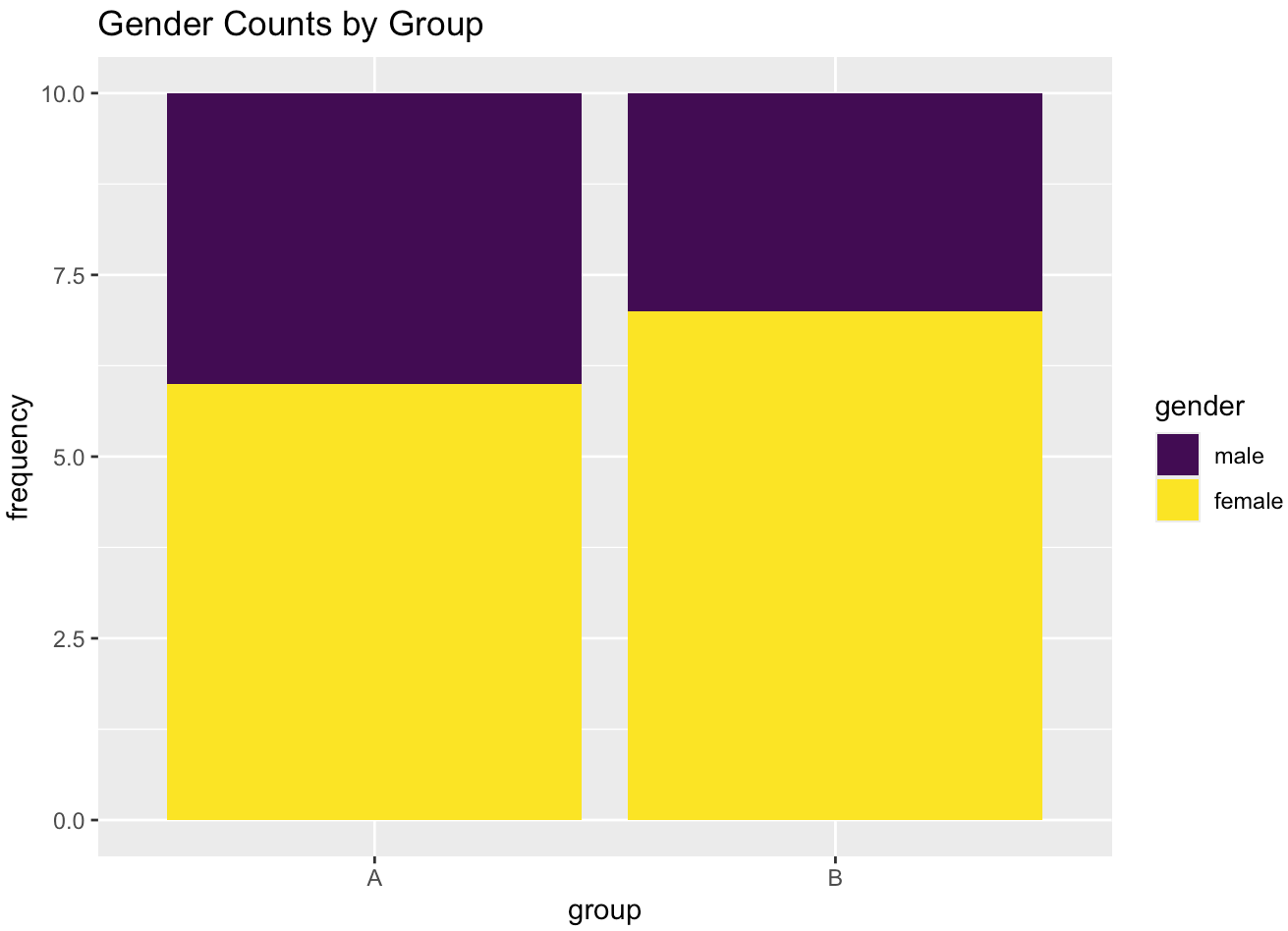
Change color

```
ggplot(gender_counts, aes(x=group, y=frequency, fill = gender)) +
  geom_bar(position="stack", stat="identity") +
  scale_fill_viridis(discrete = T) +
  ggtitle("Gender Counts by Group")
```



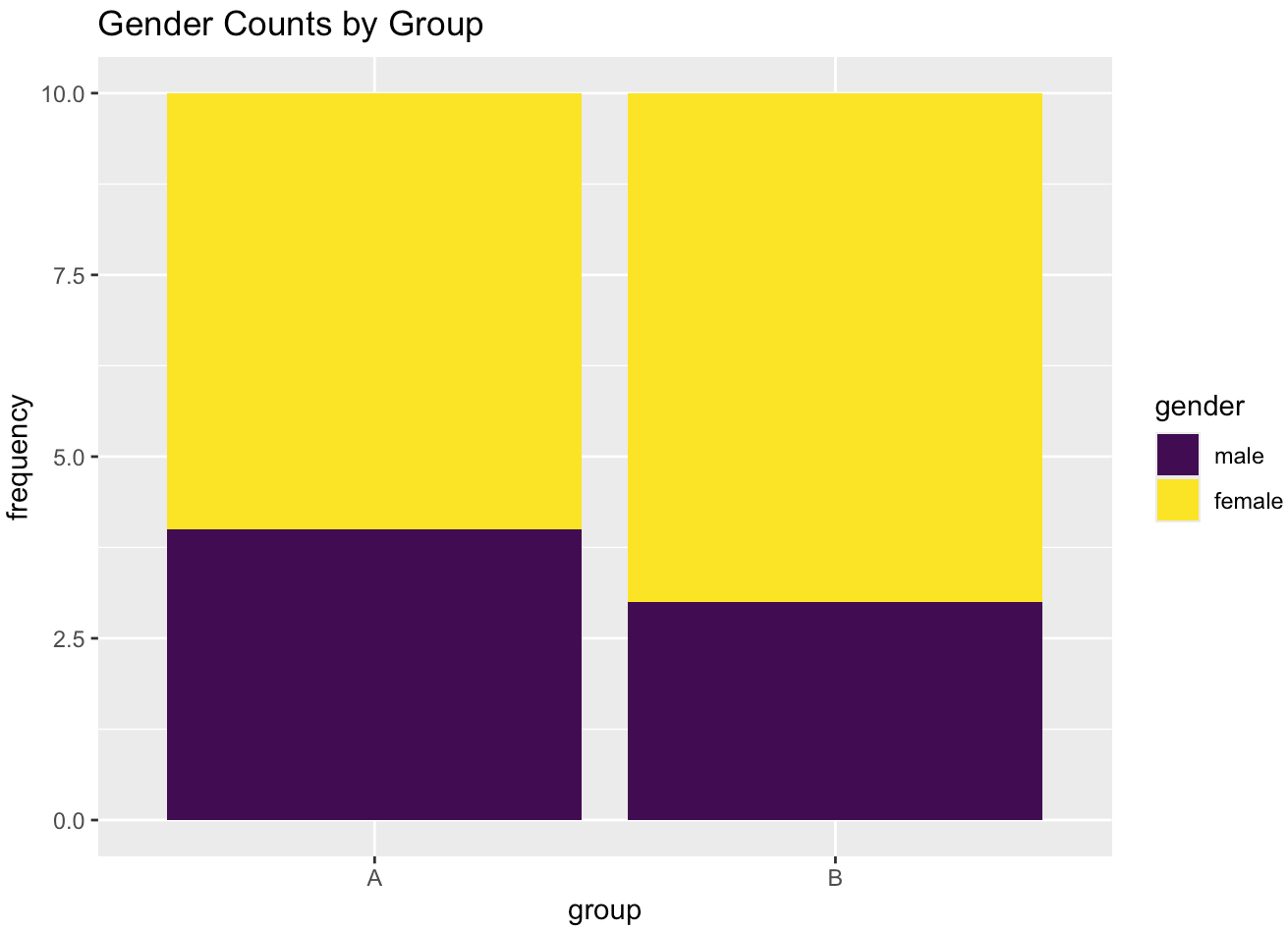
Change gender factor level to make stacked barplot look like Base R stacked plot

```
gender_counts$gender <- factor(gender_counts$gender, levels = c("male", "female"))

ggplot(gender_counts, aes(x=group, y=frequency, fill = gender)) +
  geom_bar(position="stack", stat="identity") +
  scale_fill_viridis(discrete = T) +
  ggtitle("Gender Counts by Group")
```



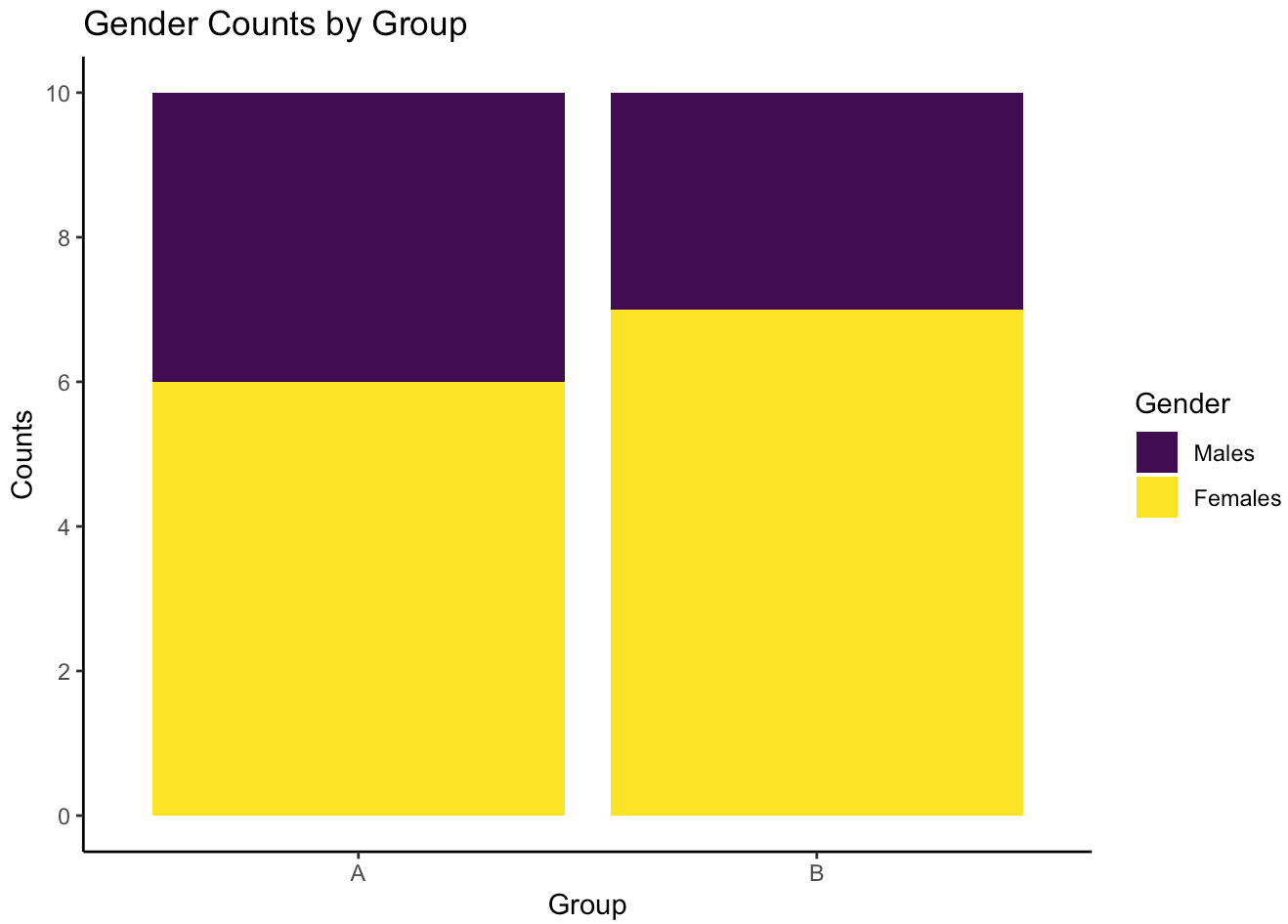
Another way to switch the position of the stacked bars without changing the factor level is to use: position_stack()

```
ggplot(gender_counts, aes(x=group, y=frequency, fill = gender)) +
  geom_bar(position = position_stack(reverse = TRUE), stat="identity") +
  scale_fill_viridis(discrete = T) +
  ggtitle("Gender Counts by Group")
```



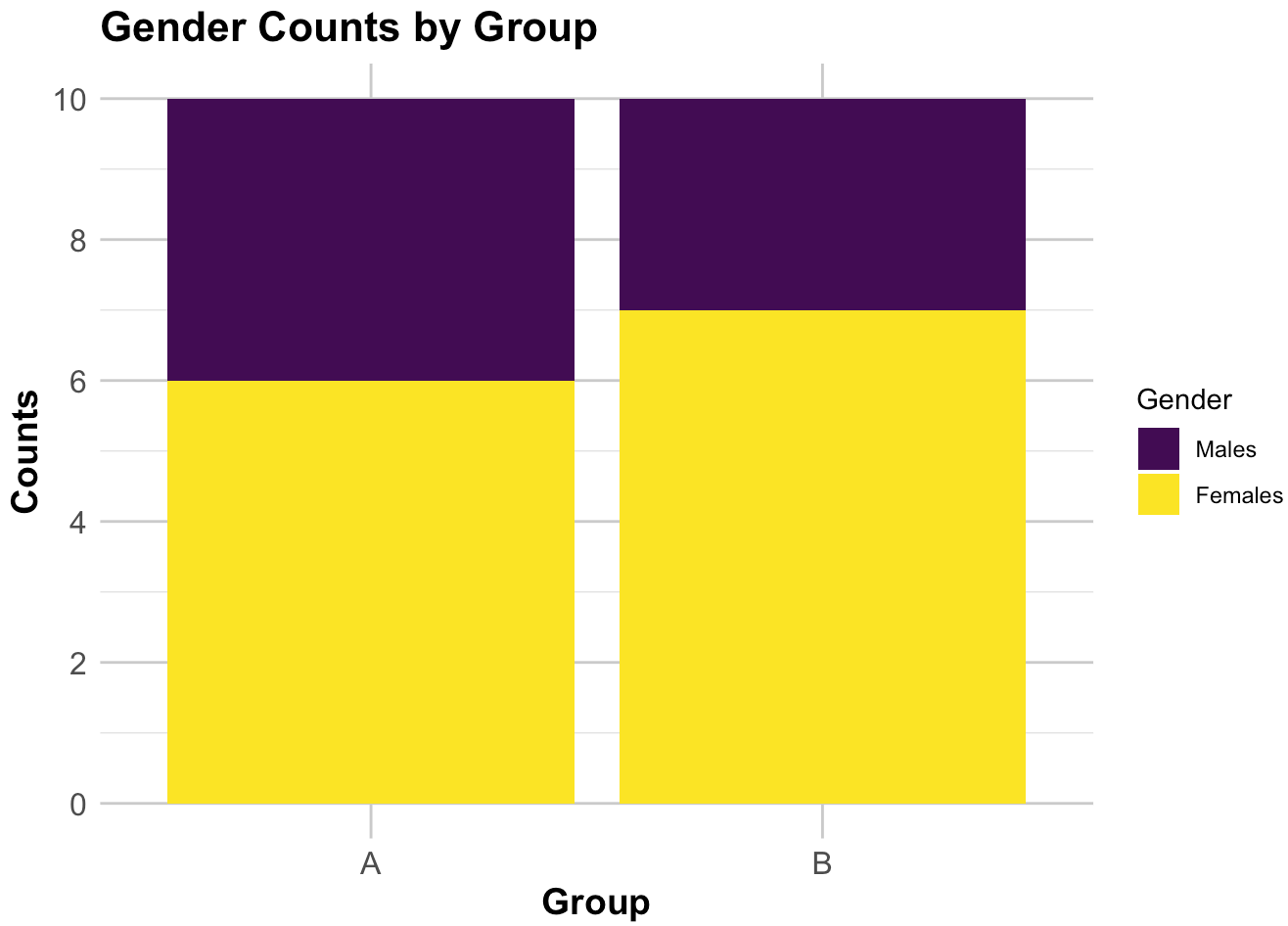
Adding labels, adjusting scales, and changing theme

```
ggplot(gender_counts, aes(x = group, y = frequency, fill = gender)) +
  geom_bar(position = "stack", stat = "identity") +
  scale_y_continuous(breaks = seq(0, sum(gender_counts$frequency), by = 2)) +
  scale_fill_manual(
    values = viridis(2),
    labels = c("Males", "Females")  # Custom legend labels
  ) +
  labs(
    title = "Gender Counts by Group",
    x = "Group",
    y = "Counts",
    fill = "Gender"  # Legend title
  ) +
  theme_classic()
```



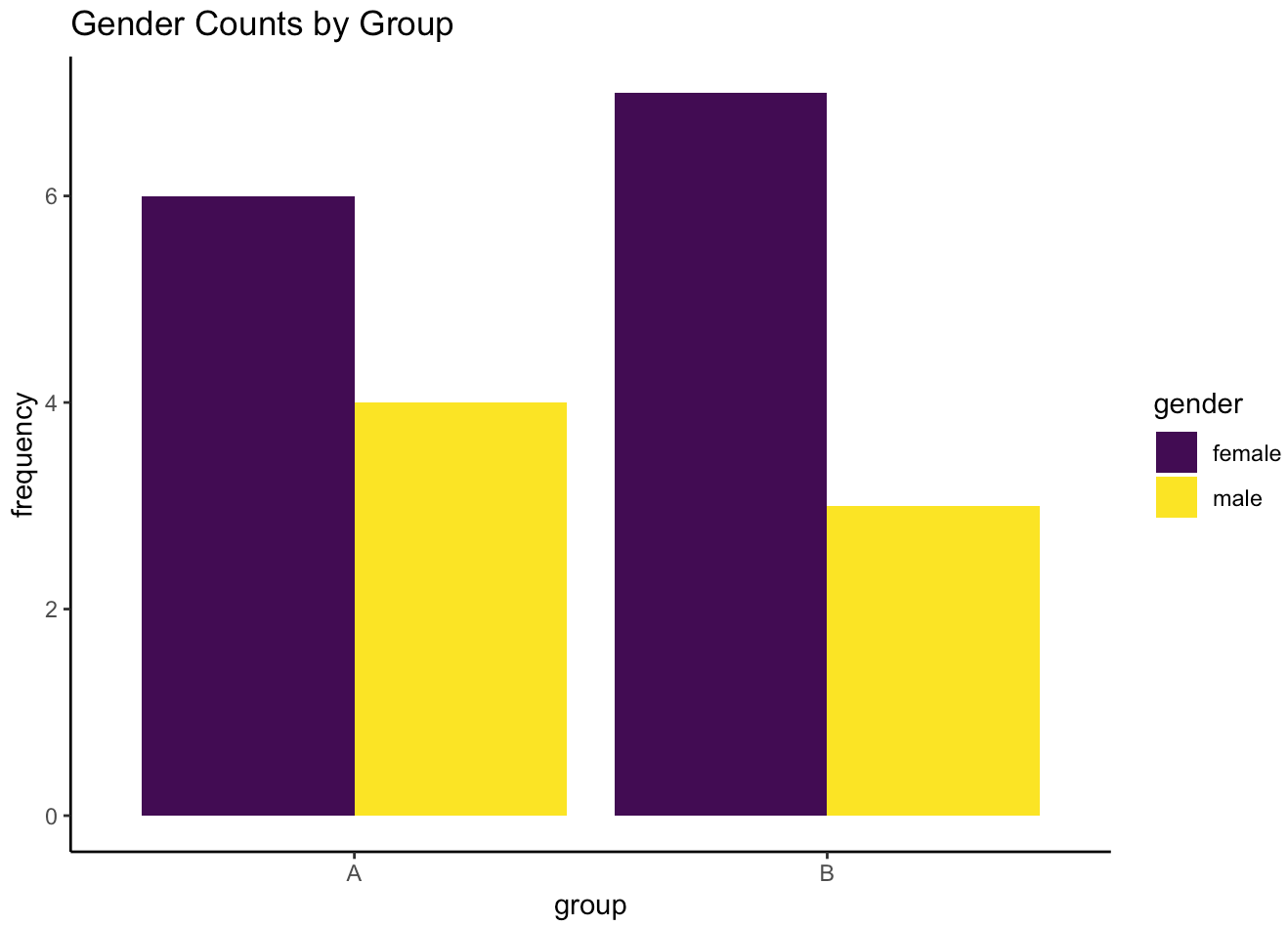
Customizing themes

```
ggplot(gender_counts, aes(x = group, y = frequency, fill = gender)) +
  geom_bar(position = "stack", stat = "identity") +
  scale_y_continuous(breaks = seq(0, sum(gender_counts$frequency), by = 2)) +
  scale_fill_manual(
    values = viridis(2),
    labels = c("Males", "Females")  # Custom legend labels
  ) +
  labs(
    title = "Gender Counts by Group",
    x = "Group",
    y = "Counts",
    fill = "Gender"  # Legend title
  ) +
  theme_minimal() +
  theme(
  plot.title = element_text(size = 16, face = "bold"),
  axis.text = element_text(size = 12),
  axis.title = element_text(size = 14),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold"),
  panel.grid.major = element_line(color = "gray80"),
  panel.grid.minor = element_line(color = "gray90"),)
```



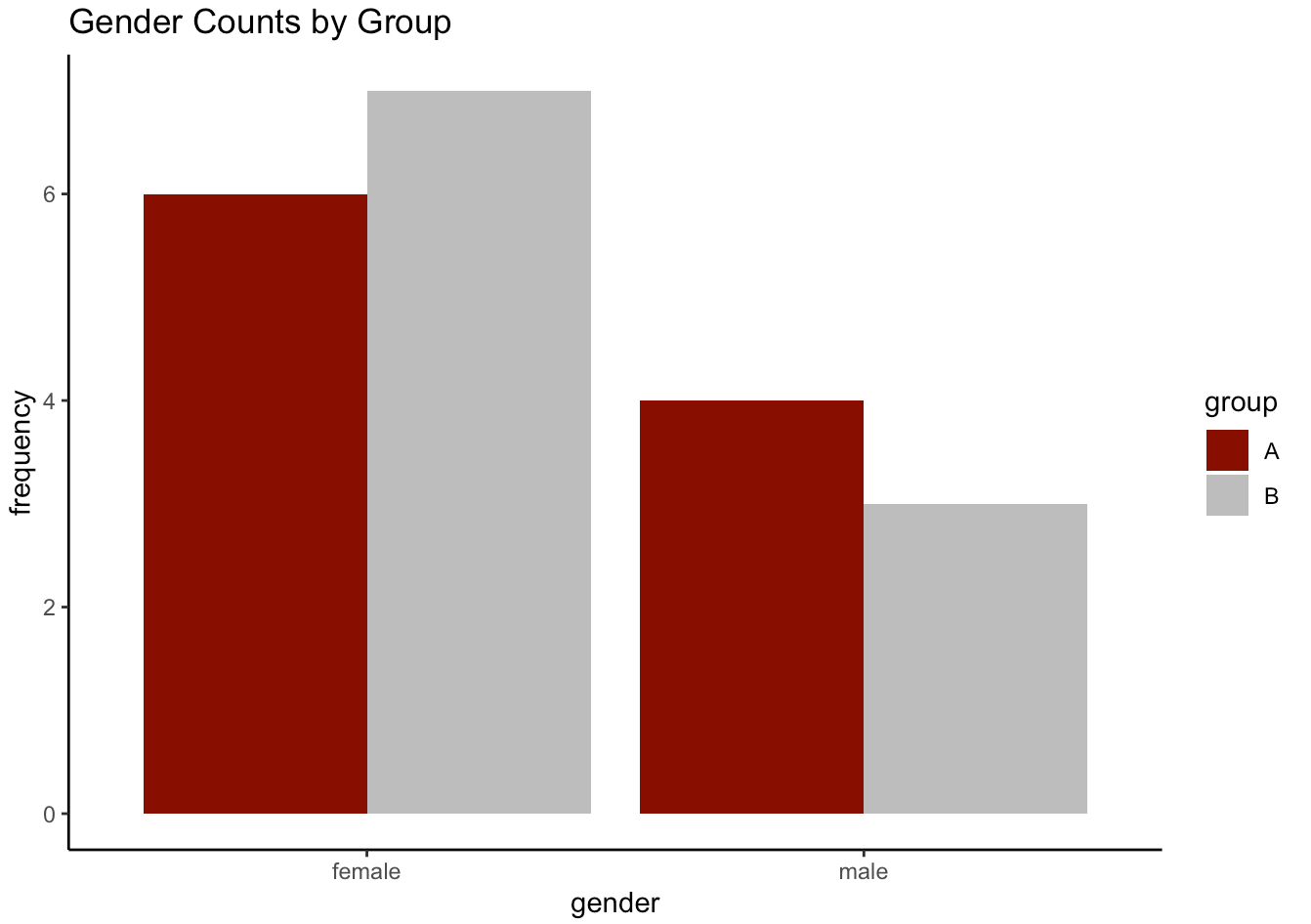
Grouped barplot

```
gender_counts$gender <- factor(gender_counts$gender, levels = c("female", "male"))

ggplot(gender_counts, aes(x=group, y=frequency, fill = gender)) +
  geom_bar(position = "dodge", stat="identity") +
  scale_fill_viridis(discrete = T) +
  ggtitle("Gender Counts by Group") +
  theme_classic()
```



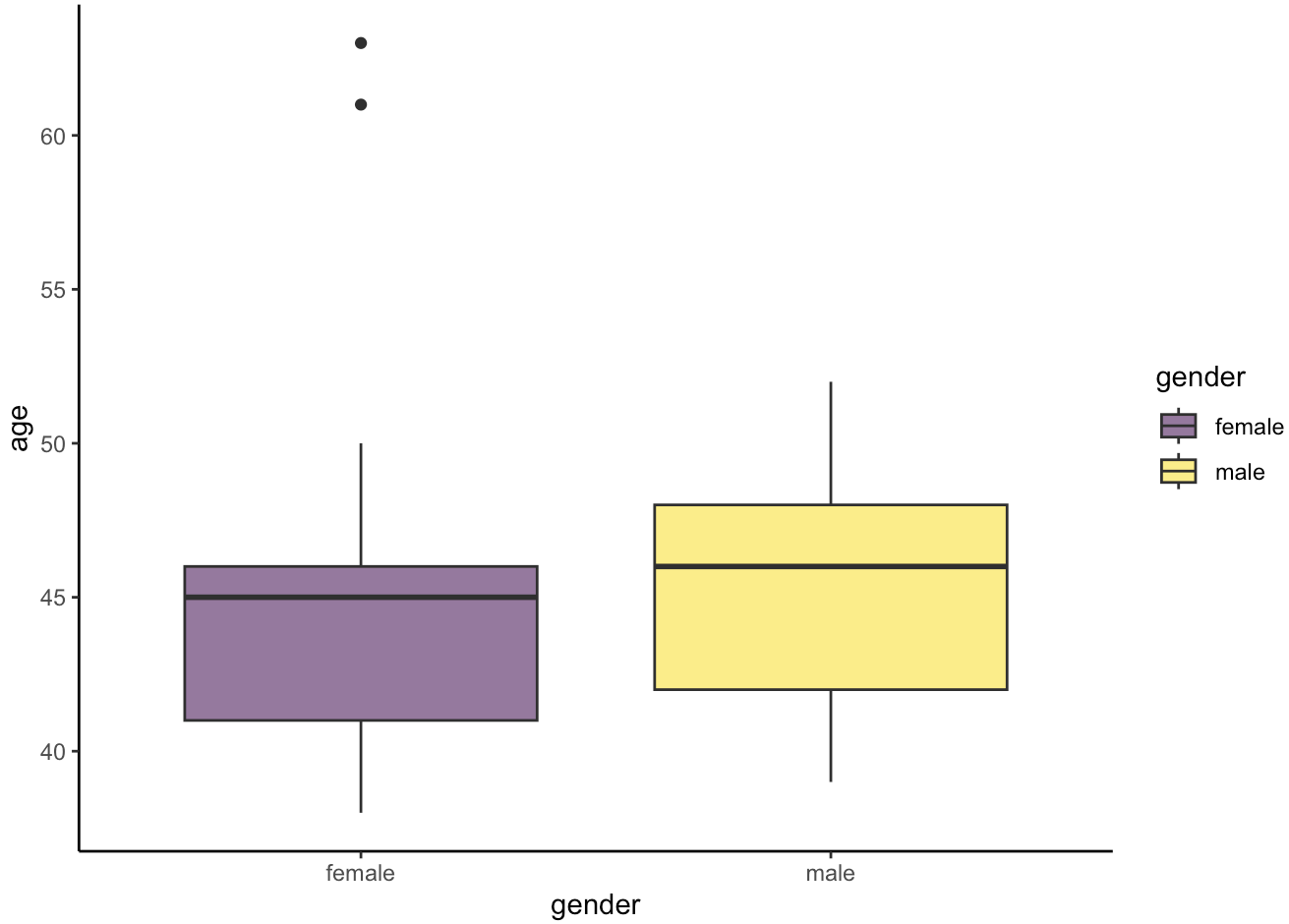
Change bar color to group instead of gender

```
ggplot(gender_counts, aes(x=gender, y=frequency, fill = group)) +
  geom_bar(position = "dodge", stat="identity") +
  scale_fill_manual(values = c("darkred", "gray")) +
  ggtitle("Gender Counts by Group") +
  theme_classic()
```



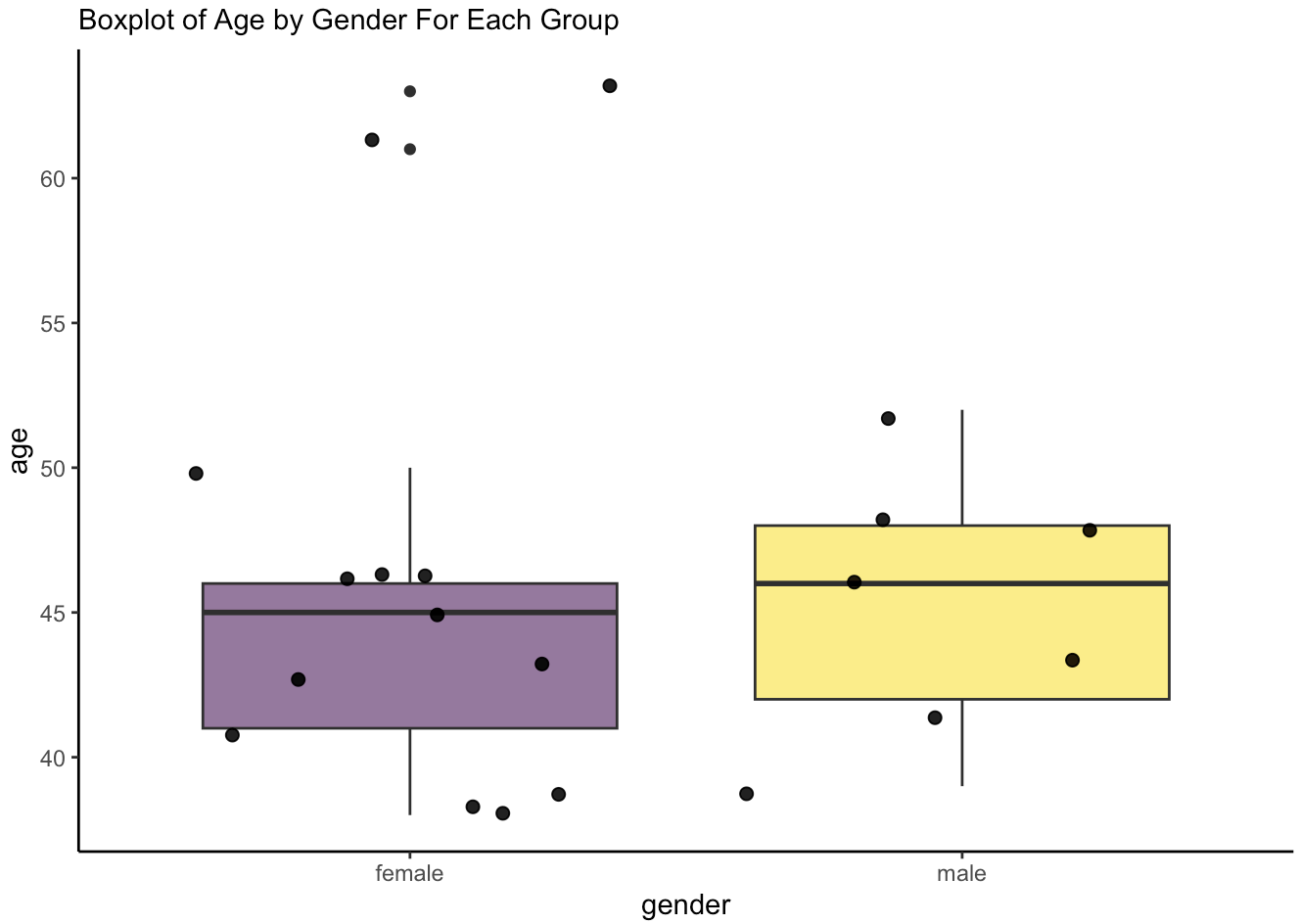
6. Boxplots

```
df_sub %>%
  ggplot(aes(x=gender, y=age, fill = gender)) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha=0.6) +
  theme_classic()
```



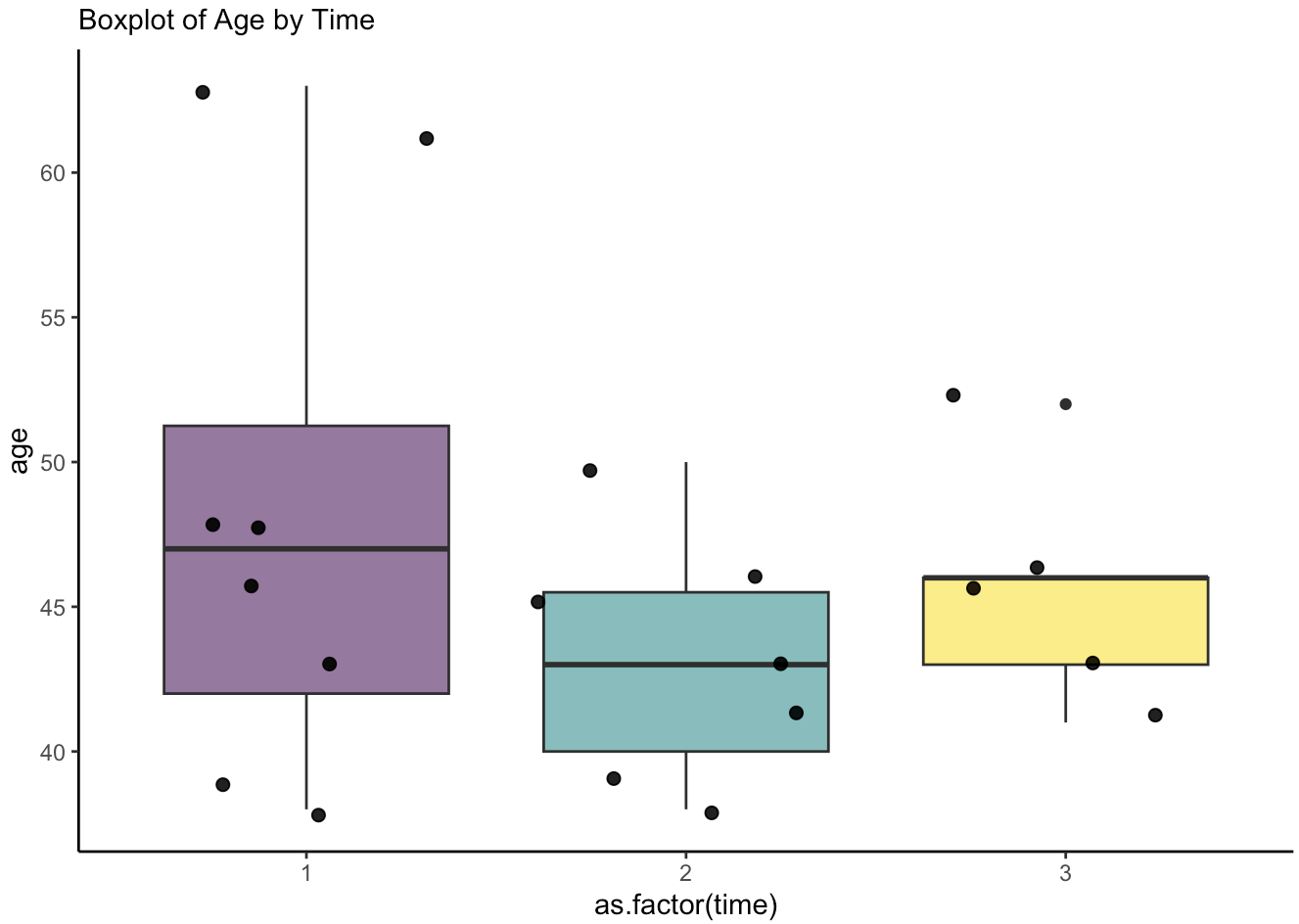
Boxplot with dots

```
df_sub %>%
  ggplot(aes(x=gender, y=age, fill=gender)) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha=0.6) +
  geom_jitter(color="black", size=2, alpha=0.9) +
  theme_classic() +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)) +
  ggtitle("Boxplot of Age by Gender For Each Group")
```



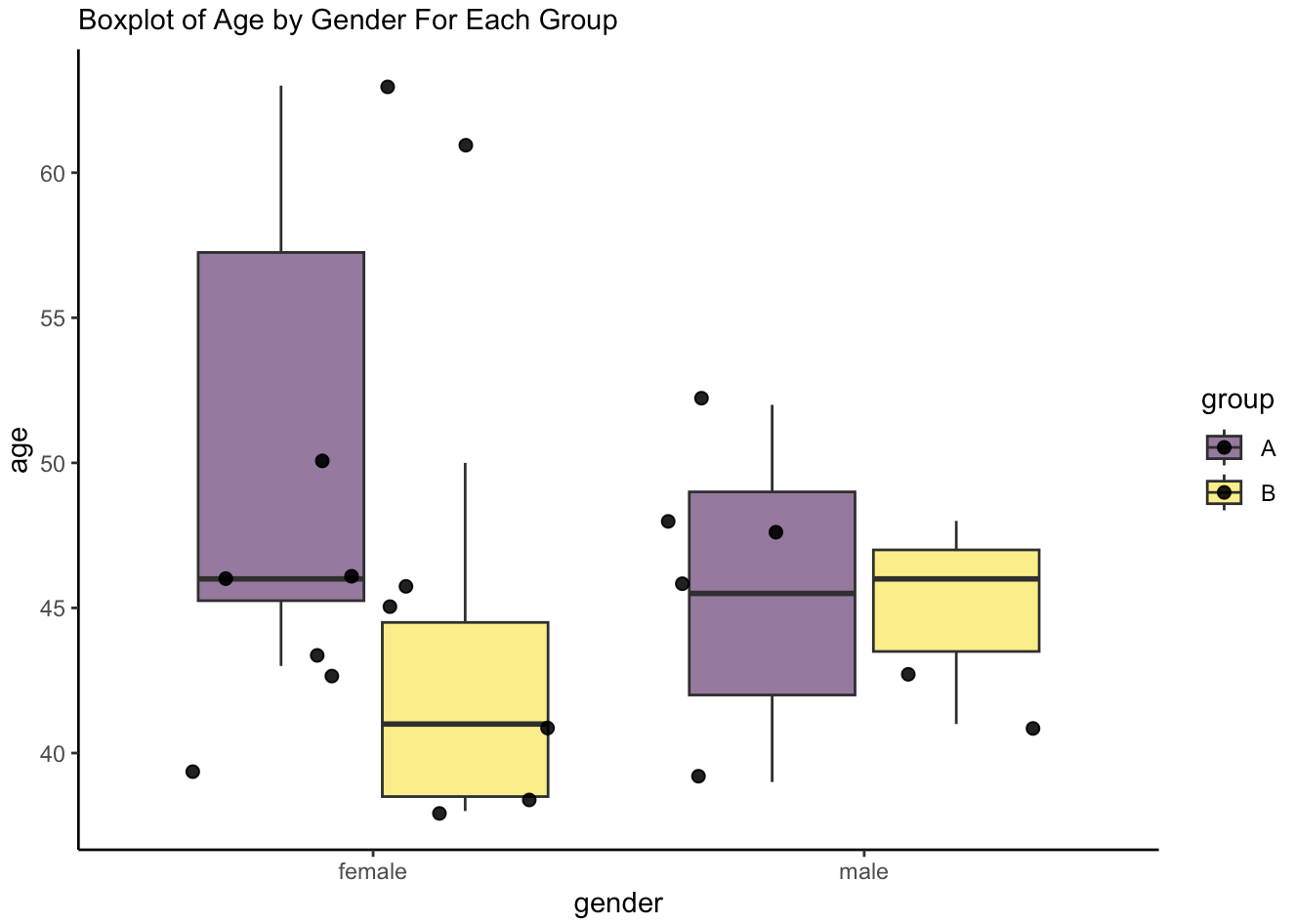
Boxplot of age by time

```
df_sub %>%
  ggplot(aes(x=as.factor(time), y=age, fill=as.factor(time))) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha=0.6) +
  geom_jitter(color="black", size=2, alpha=0.9) +
  theme_classic() +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)) +
  ggtitle("Boxplot of Age by Time")
```



Age by gender for each group

```
df_sub %>%
  ggplot(aes(x=gender, y=age, fill=group)) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha=0.6) +
  geom_jitter(color="black", size=2, alpha=0.9) +
  theme_classic() +
  theme(plot.title = element_text(size=11)) +
  ggtitle("Boxplot of Age by Gender For Each Group")
```



7. Violin plots

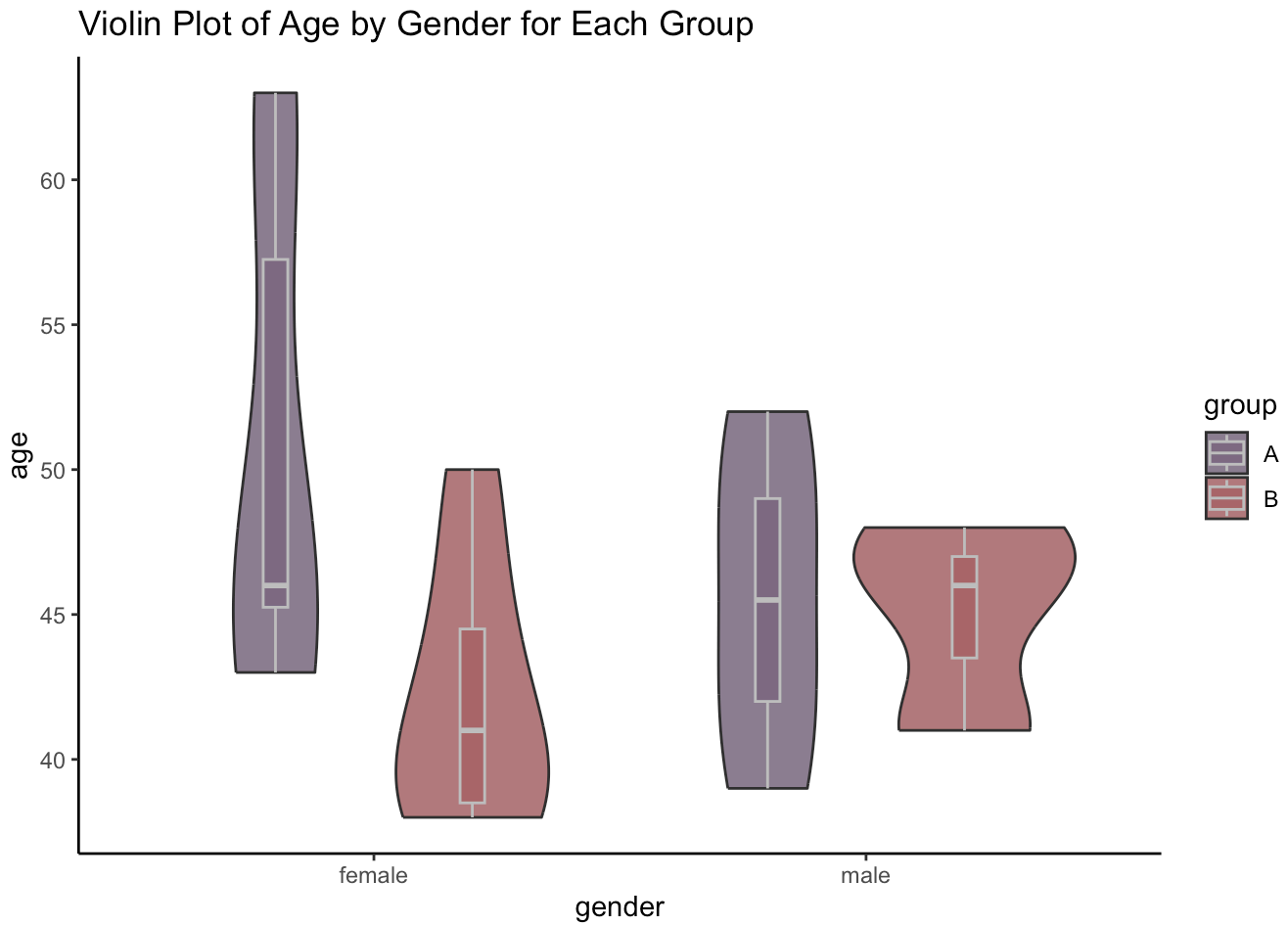
```
df_sub %>%
  ggplot(aes(x=gender, y=age, fill=group)) +
  geom_violin() +
  scale_fill_viridis(discrete = TRUE, alpha = 0.6, option="H") +
  theme_classic() +
  ggtitle("Violin Plot of Age by Gender For Each Group")
```

Plot violin and small boxplot

```
df_sub %>%
  ggplot(aes(x = gender, y = age, fill = group)) +
  geom_violin(alpha = 0.6) +
  geom_boxplot(width = 0.1, color = "gray", alpha = 0.2) +
  scale_fill_viridis(discrete = TRUE, option = "H") +
  theme_classic() +
  ggtitle("Violin Plot of Age by Gender for Each Group")
```

Fix the boxplot positions

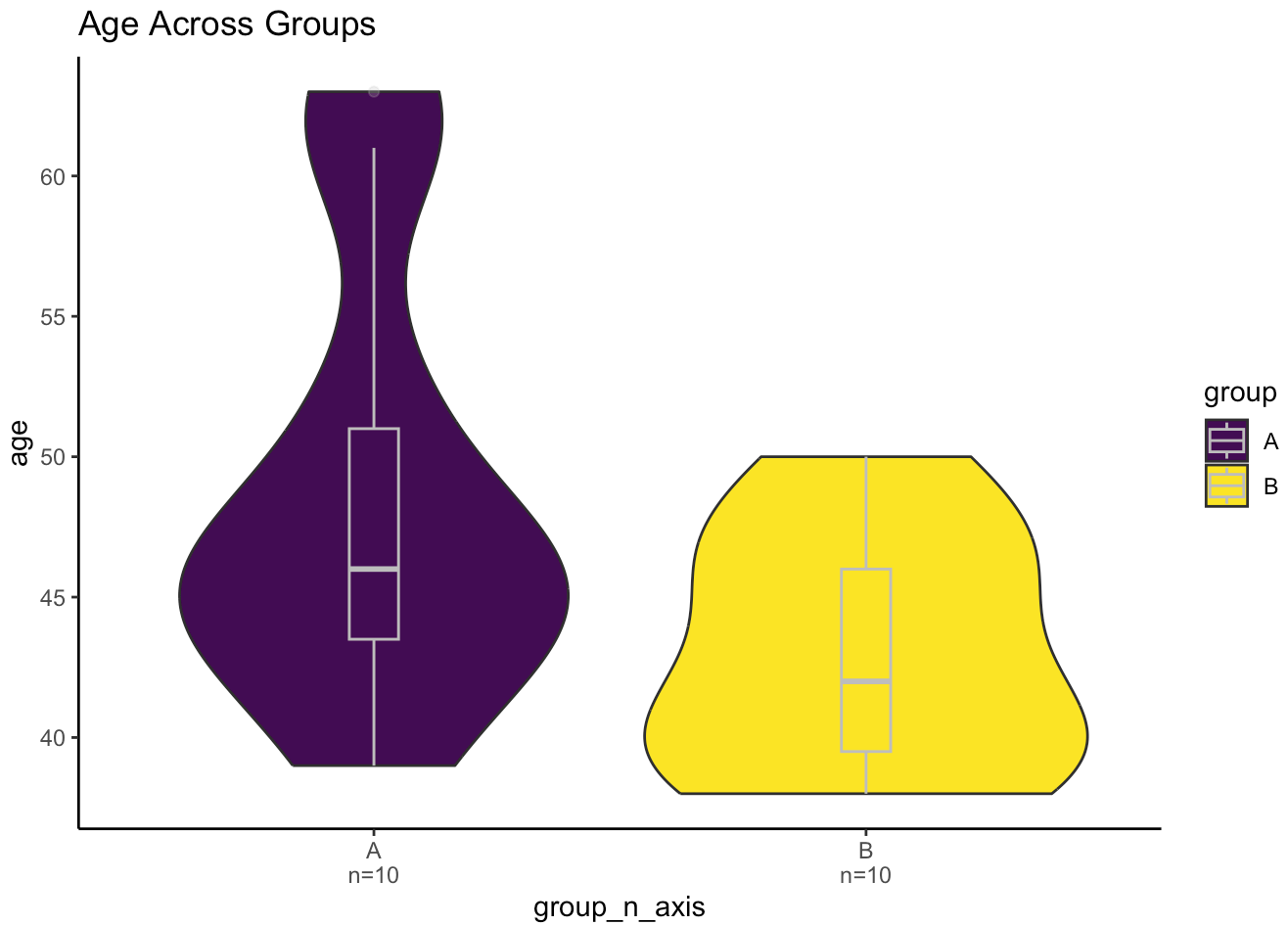
```
df_sub %>%
  ggplot(aes(x = gender, y = age, fill = group)) +
  geom_violin(position = position_dodge(width = 0.80), alpha = 0.6) +
  geom_boxplot(position = position_dodge(width = 0.80), width = 0.1, color = "gray", alp
ha = 0.2) +
  scale_fill_viridis(discrete = TRUE, option = "H") +
  theme_classic() +
  ggtitle("Violin Plot of Age by Gender for Each Group")
```



Making axis labels with n numbers

```
sample_size = df_sub %>% group_by(group) %>% summarize(num=n())

df_sub %>%
  left_join(sample_size) %>%
  mutate(group_n_axis = paste0(group, "\n", "n=", num)) %>%
  ggplot(aes(x=group_n_axis, y=age, fill = group)) +
  geom_violin() +
  geom_boxplot(width=0.1, color="gray", alpha=0.2) +
  scale_fill_viridis(discrete = TRUE, option = "D") +
  theme_classic() +
  ggtitle("Age Across Groups")
```

```
## Joining with `by = join_by(group)`
```



Create label n per group and color by gender

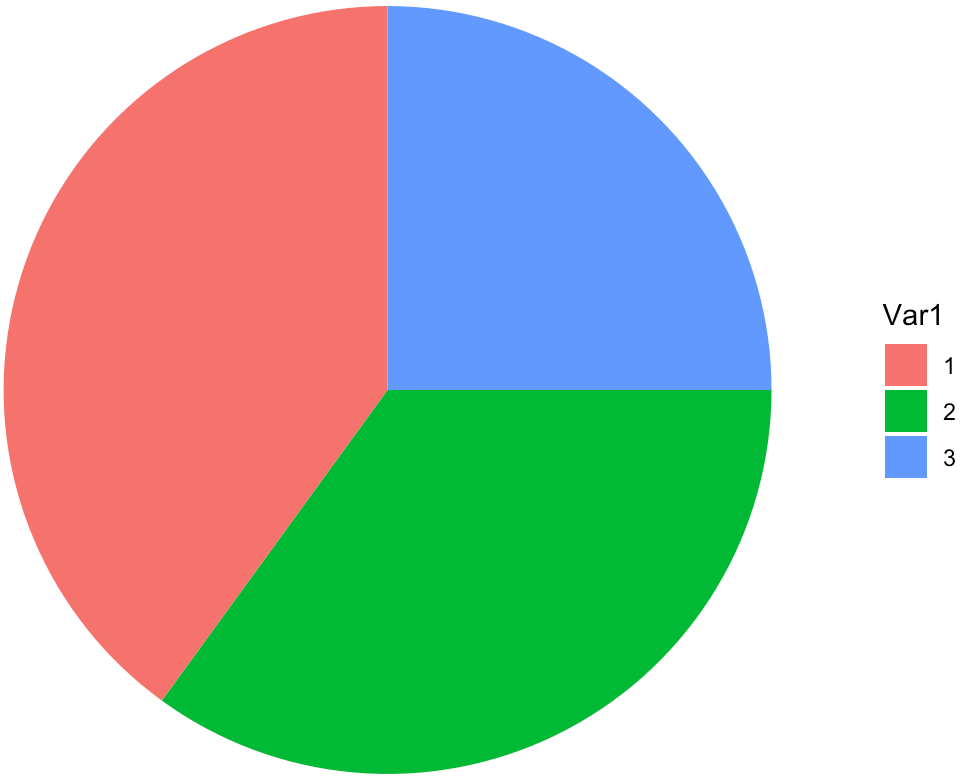
```
sample_size = df_sub %>% group_by(group, gender) %>% summarize(num=n())
```

```
## `summarise()` has grouped output by 'group'. You can override using the
## `.groups` argument.
```

```
df_sub_size = df_sub %>%
  left_join(sample_size, by = c("group", "gender")) %>%
  mutate(myaxis = paste0(group, "\n", "n=", num))

df_sub %>%
  left_join(sample_size, by = c("group", "gender")) %>%
  mutate(group_n_axis = paste0(group, "\n", "n=", num)) %>%
  ggplot(aes(x = group_n_axis, y = age, fill = gender)) +
  geom_violin(width = 1, alpha = 0.6) +
  geom_boxplot(width = 0.1, color = "gray", alpha = 0.2) +
  scale_fill_viridis(discrete = TRUE, option = "H") +
  theme_classic() +
  ggtitle("Violin Plot of Age by Gender and Group") +  # Set title
  theme(plot.title = element_text(size=11, face = "bold"))
```

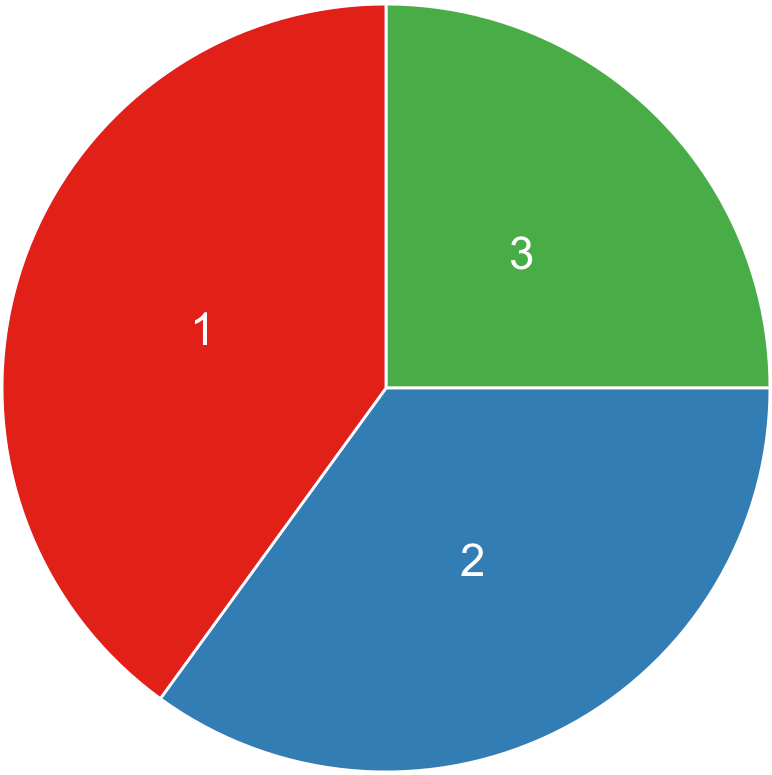
8. Pie charts

```
# Create a table from the data
time_counts <- data.frame(table(df_sub$time))

ggplot(time_counts, aes(x="", y=Freq, fill=Var1)) +
  geom_bar(stat="identity", width=1) +
  coord_polar("y", start=0) +
  theme_void() # removes background, grid, and numeric labels
```



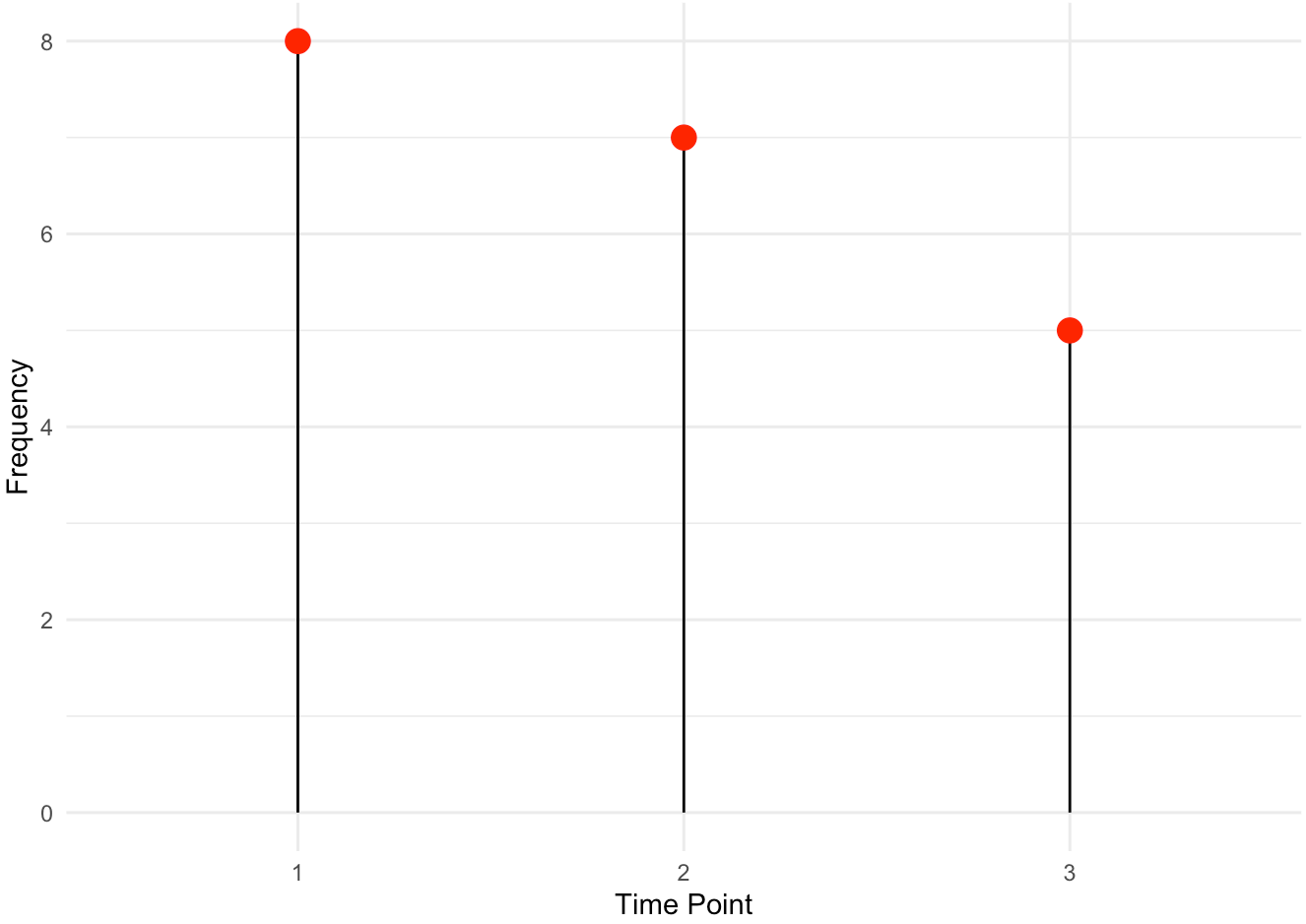
Change color Colorbrewer (https://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3)

```
# change color and add labels inside the pie chart
time_counts <- time_counts %>%
  arrange(desc(Var1)) %>%
  mutate(prop = Freq / sum(time_counts$Freq) *100) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop )

ggplot(time_counts, aes(x="", y=prop, fill=Var1)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +

  geom_text(aes(y = ypos, label = Var1), color = "white", size=6) +
  scale_fill_brewer(palette="Set1") # ColorBrewer package
```



9. Lollipop plot

```
ggplot(time_counts, aes(x=Var1, y=Freq)) +
  geom_segment(aes(x=Var1, xend=Var1, y=0, yend=Freq), color="black") +
  geom_point(color="red", size=4) +
  theme_minimal() +
  xlab("Time Point") +
  ylab("Frequency")
```



ggplot themes

ggplot themes = several built-in themes that you can use to customize the appearance of your plots

```
# Sample data
example_df <- data.frame(
  x = 1:10,
  y = rnorm(10))

# Define a list of theme functions
themes <- list(
  theme_gray = theme_gray(),
  theme_bw = theme_bw(),
  theme_linedraw = theme_linedraw(),
  theme_light = theme_light(),
  theme_dark = theme_dark(),
  theme_minimal = theme_minimal(),
  theme_classic = theme_classic(),
  theme_void = theme_void(),
  theme_test = theme_test())

# Define descriptive names for each theme
theme_names <- c(
  "Gray",
  "Black & White",
  "Linedraw",
  "Light",
  "Dark",
  "Minimal",
  "Classic",
  "Void",
  "Test")

# Create a plot with each theme and store them in a list
plots <- lapply(seq_along(themes), function(i) {
  ggplot(example_df, aes(x, y)) +
    geom_point() +
    themes[[i]] +
    ggtitle(paste("Theme:", theme_names[i]))})
```

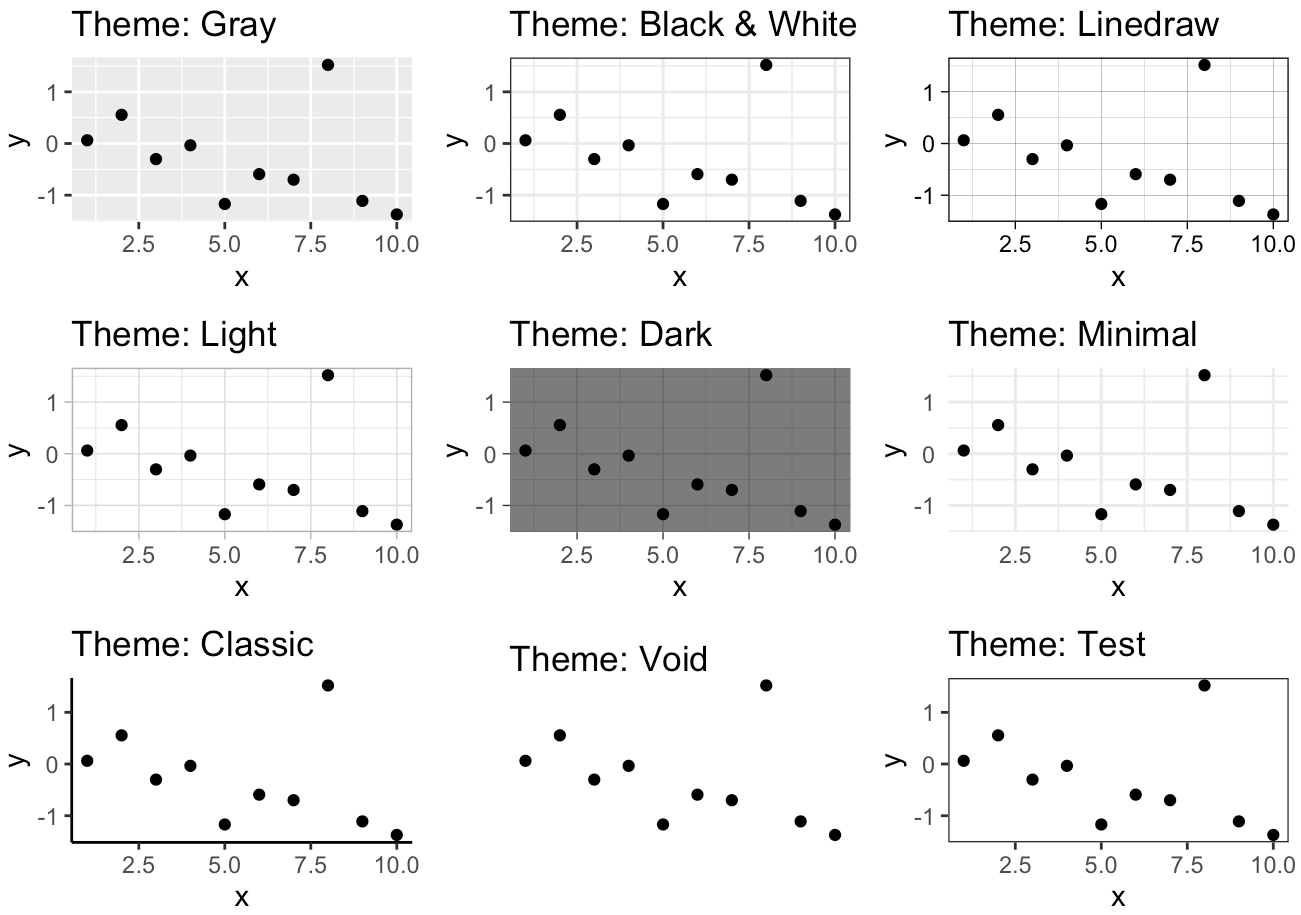
Save each plot using ggsave in a loop

```
for (i in seq_along(plots)) {
  ggsave(filename = paste0("plot_with_", theme_names[i], ".tiff"), plot = plots[[i]])}
```

```
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
## Saving 7 x 5 in image
```

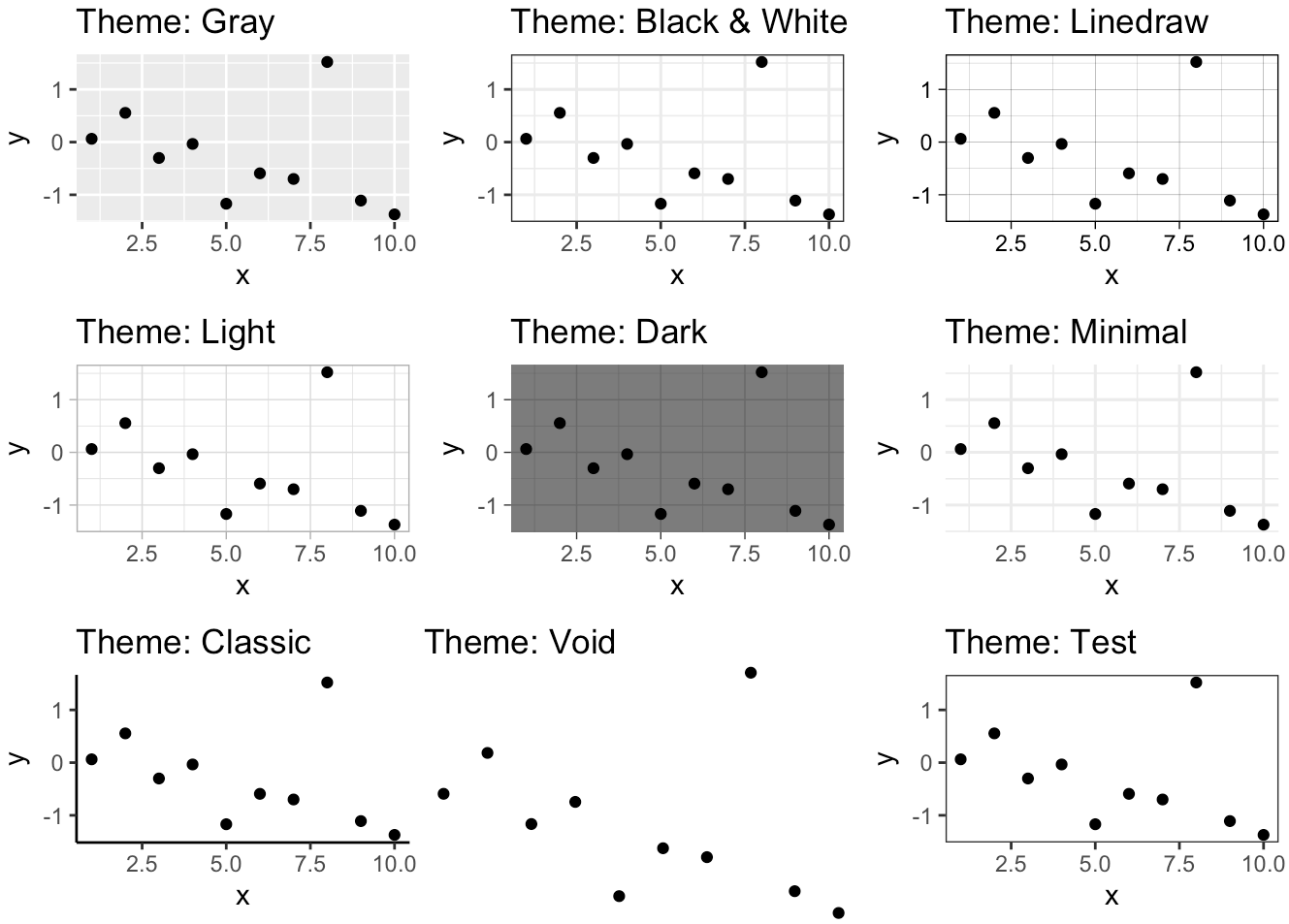
Wrapping multiple plots using patchwork

```
# Combine plots using patchwork
wrap_plots(plots, ncol = 3)
```



wrapping multiple plots using cowplots

```
plot_grid(plotlist = plots, ncol = 3)
```

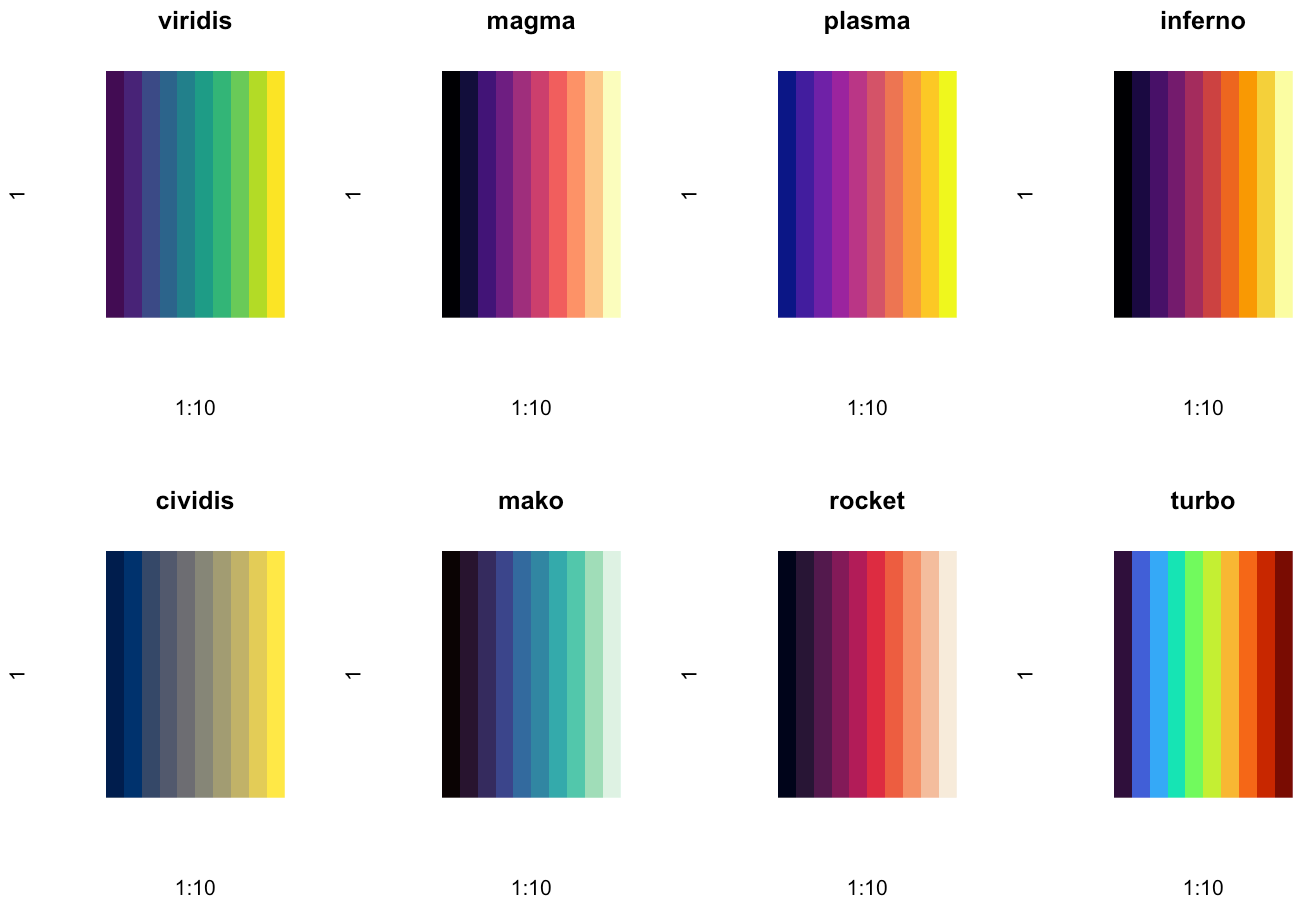


Color

Viridis = created as a new default colormap for matplotlib in python
Viridis = Latin for green, named after a snake
Viridis (https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html)
SciPy2015 (https://www.youtube.com/watch?v=xAoljeRJ3lU)

Visualize all viridis options

```
par(mfrow = c(2, 4))
image(1:10, 1, as.matrix(1:10), col = viridis(10), main = "viridis", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = magma(10), main = "magma", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = plasma(10), main = "plasma", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = inferno(10), main = "inferno", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = cividis(10), main = "cividis", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = mako(10), main = "mako", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = rocket(10), main = "rocket", axes = FALSE)
image(1:10, 1, as.matrix(1:10), col = turbo(10), main = "turbo", axes = FALSE)
```



Practice Session

1. How to change the y-axis scale for the gender grouped barplot to increment by 1?
2. For the lollipop plot, how would you change the line width thickness and flip the plot horizontally?
3. Create a graph with custom x-axis label to reflect the gender and n on the x-axis (female = 6, etc.) and color by group?
4. Fix the ordering of the x-axis labels so the groups are (A, B, A, B)