Tidying

Saneesh

Contents

1 Shortcuts

`alt` + `-` will add `<-` an assignment operator `Shift` + `ctrl` + `c` to add `#` in front of a line
`----` or four dashes for a header, so it is easy to navigate through the script
`command`/`Ctrl` + `Shift` + `m` for pipe `%>%`
`Ctrl`+ `Alt` + `i` for new code chunk

2 Rmd syntax

Plain text
End a line with two spaces to start a new paragraph.
For *italics* `*text*` or `_text_` (without gap `*`text`*`)
For **bold** `**text**`(without gap `**`text`**`)
superscript^2 superscript^2^
Strikethrough `~Strikethrough~`

More about R Markdown
More about PDF document

Adding web link to a text: ConservationPlus to my web page e.g., `[text] and without gap (paste link with https://conservationplus.weebly.com/)`

3 Logical operations

1==1 # equal
1!=3 # unequal
13<14 # 13 smaller than 14
14>13 # 14 bigger than 13
12>=0 # 12 greater or equal to zero
12<=3 # 12 smaller or equal to zero

4 Creating data.frame

i.e. family

```
name <- c("saneesh", "sanusha", "appu", "kishan")
weight <- c(63, 48, 20, NA)
height <- c(164, 150, NA, 75)
family <- data.frame(name, weight, height)
family %>%
    as_tibble()
```

```
## # A tibble: 4 x 3
##   name    weight height
##   <chr>    <dbl>  <dbl>
## 1 saneesh     63    164
## 2 sanusha     48    150
## 3 appu        20     NA
## 4 kishan      NA     75
```

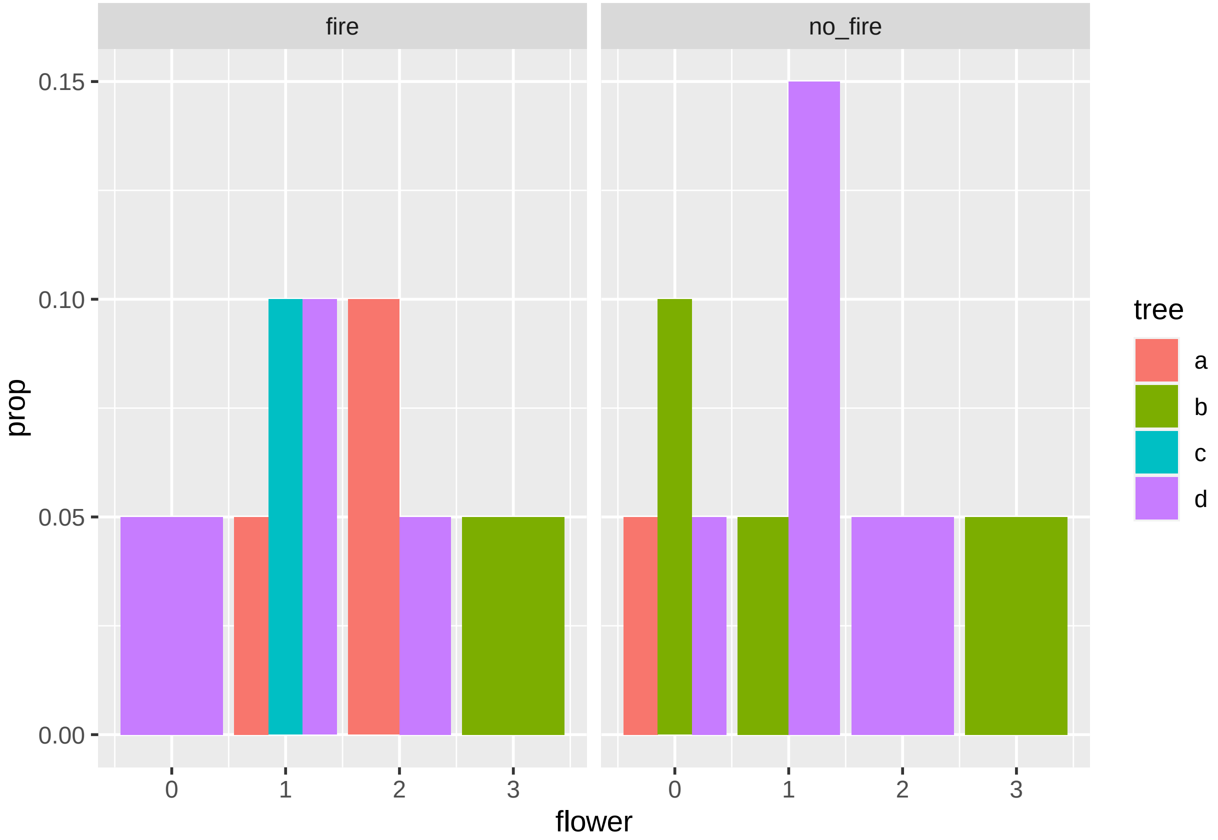
```
same.family <- data.frame(name = c("saneesh", "sanusha", "appu",
    "kishan"), weight = c(63, 48, 20, NA), height = c(164, 150,
    NA, 75))
```

4.1 Abundance

```
Community <- c(rep("A", 3), rep("B", 3))
Species <- rep(c("X", "Y", "Z"), 2)
Count <- c(100, 0, 50, 50, 30, 40)

df <- data.frame(Community, Species, Count)

# abundance refers to the total number of individuals of
# different species within each community. It represents
# the quantity or total count of individuals present.

abundance <- df %>%
    group_by(Community) %>%
    summarise(Total_abundance = sum(Count))

# Species richness, on the other hand, refers to the total
# number of unique species present in each community. It
# represents the diversity of species within a community.

richness <- df %>%
    group_by(Community) %>%
    filter(Count > 0) %>%
    distinct(Species) %>%
    summarise(Richness = n())
```

4.2 Proportion

```
tree <- c("a", "b", "c", "d")
treatment <- c("fire", "no_fire")


data.frame(tree = sample(tree, 20, replace = T), treatment = sample(treatment,
    20, replace = T), flower = rbinom(20, 3, prob = 0.3)) %>%
    group_by(tree, treatment, flower) %>%
    summarise(count = n(), .groups = "drop") %>%
    mutate(prop = count/sum(count)) %>%
    ggplot(aes(x = flower, y = prop, fill = tree)) + geom_bar(stat = "identity",
    position = "dodge") + facet_wrap(~treatment)
```



4.3 Zero count

```r
library(dplyr)

df <- data.frame(tree = c(rep("a", 4), rep("b", 4)), seeds = c(0,
    0, 0, 1, 2, 3, 0, 0))

zero_counts <- df %>%
    group_by(tree) %>%
    summarise(zero_count = sum(seeds == 0))

print(zero_counts)
```

```
## # A tibble: 2 x 2
##   tree  zero_count
##   <chr>      <int>
## 1 a              3
## 2 b              2
```

4.4 Data frame with unequal values 10 and 8

```r
library(tidyverse)
data <- data.frame(sex = c(rep("female", 10), rep("male", 8)),
    score = c(rnorm(n = 10, mean = 7.56, sd = 1.978), rnorm(n = 8,
        mean = 7.75, sd = 1.631)))

data %>%
    head(5)
```

```
##      sex    score
## 1 female 9.127767
## 2 female 5.643989
## 3 female 7.807600
## 4 female 9.099800
## 5 female 7.659015
```

```r
data %>%
    group_by(sex) %>%
    summarise(score = n()) %>%
    mutate(freq = score/sum(score) * 100)
```

```
## # A tibble: 2 x 3
##   sex    score  freq
##   <chr>  <int> <dbl>
## 1 female    10  55.6
## 2 male       8  44.4
```

4.5 Name the unnamed first column of a data.frame

```r
# newdf <- rownames_to_column(df, var = 'name to an
# unnamed')
```

5 Creating a tibble

```r
library(tidyverse)
years <- tribble(~Location, ~Year, ~Month, ~Day, ~Lenght, "Sydney",
    2000, 9, 15, 12.1213, "Athens", 2004, 8, 13, 12.1212, "Beijing",
    2008, 8, 8, 13.212, "London", 2012, 7, 27, 13.1212, "Rio de Janeiro",
    2016, 8, 5, 65)

# write.csv(years, file = 'years.csv', row.names = FALSE) #
# without index use row.names = FALSE
```

5.1 tabyl

tabyl

5.2 mutate round

```r
# run previous code chunk
library(gt)
years %>%
    gt()
```

Location	Year	Month	Day	Lenght
Sydney	2000	9	15	12.1213
Athens	2004	8	13	12.1212
Beijing	2008	8	8	13.2120
London	2012	7	27	13.1212
Rio de Janeiro	2016	8	5	65.0000

```r
years %>%
    mutate(Lenght = round(Lenght, 2)) %>%
    gt() %>%
    tab_options(column_labels.font.size = 11, column_labels.font.weight = "bold",
        table.font.size = 10, ) %>%
    opt_table_outline(style = "solid", width = px(2))
```

Location	Year	Month	Day	Lenght
Sydney	2000	9	15	12.12

Athens	2004	8	13	12.12
Beijing	2008	8	8	13.21
London	2012	7	27	13.12
Rio de Janeiro	2016	8	5	65.00

```
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
data <- data.frame(HairEyeColor)

data %>%
    tabyl(Hair, Eye) %>%
    adorn_percentages("row") %>%
    adorn_pct_formatting(digits = 2) %>%
    adorn_ns() %>%
    knitr::kable()
```

Hair	Brown	Blue	Hazel	Green
Black	25.00% (2)	25.00% (2)	25.00% (2)	25.00% (2)
Brown	25.00% (2)	25.00% (2)	25.00% (2)	25.00% (2)
Red	25.00% (2)	25.00% (2)	25.00% (2)	25.00% (2)
Blond	25.00% (2)	25.00% (2)	25.00% (2)	25.00% (2)

6 Data cleaning

6.1 Find NAs

```
# identify location of NAs in vector
which(is.na(family))
```

```
## [1]  8 11
```

```
colSums(is.na(family))
```

```
##   name weight height
##      0      1      1
```

6.2 Replace na

```
mat <- matrix(sample(c(NA, 1:5), 50, replace = TRUE), 5)
df <- as.data.frame(mat)
df %>%
    replace(is.na(.), 0) %>%
    View()
```

6.3 Drop na

see spread & gather

6.4 Clean names

```
# install.packages('janitor')
library(janitor)

id <- (c(1, 1, 2, 2, 3, 3))
Country <- c("Angola", "Angola", "Botswana", "Botswana", "Zimbabwe",
    "Zimbabwe")
year <- c("2006", "2007", "2008", "2009", "2010", "2006")
bank.ratio <- c(24, 25, 38, 34, 42, 49)
Reserve.ratio <- c(77, 59, 64, 65, 57, 86)
broad.money <- c(163, 188, 317, 361, 150, 288)


bank <- data.frame(id, Country, year, bank.ratio, Reserve.ratio,
    broad.money)

bank <- bank %>%
    clean_names()  # replaced . with _

glimpse(bank)
```

```
## Rows: 6
## Columns: 6
## $ id            <dbl> 1, 1, 2, 2, 3, 3
## $ country       <chr> "Angola", "Angola", "Botswana", "Botswana", "Zimbabwe", ~
## $ year          <chr> "2006", "2007", "2008", "2009", "2010", "2006"
## $ bank_ratio    <dbl> 24, 25, 38, 34, 42, 49
## $ reserve_ratio <dbl> 77, 59, 64, 65, 57, 86
## $ broad_money   <dbl> 163, 188, 317, 361, 150, 288
```

6.5 Filter

filter bank data frame below such that it retains a country if a given id is satisfied e.g. filtering a data frame that has countries with id 1 and 2 only

```
bank %>%
    filter(id %in% c(1, 2)) %>%
    as_tibble()
```

```
## # A tibble: 4 x 6
##      id country  year  bank_ratio reserve_ratio broad_money
##   <dbl> <chr>    <chr>      <dbl>         <dbl>       <dbl>
## 1     1 Angola   2006          24            77         163
## 2     1 Angola   2007          25            59         188
## 3     2 Botswana 2008          38            64         317
## 4     2 Botswana 2009          34            65         361
```

summarise fund available with each countries

```
bank %>%
    group_by(country) %>%
    summarise(fund = sum(broad_money)) %>%
    as_tibble()
```

```
## # A tibble: 3 x 2
##   country   fund
##   <chr>    <dbl>
## 1 Angola     351
## 2 Botswana   678
## 3 Zimbabwe   438
```

6.6 Rename column

column: new name= old name

```
iris %>%
    rename(S.len = Sepal.Length, Sp. = Species) %>%
    head(3)
```

```
##   S.len Sepal.Width Petal.Length Petal.Width    Sp.
## 1   5.1         3.5          1.4         0.2 setosa
## 2   4.9         3.0          1.4         0.2 setosa
## 3   4.7         3.2          1.3         0.2 setosa
```

6.7 Rename to lower

```
iris %>%
    rename_with(tolower) %>%
    head(3)
```

```
##   sepal.length sepal.width petal.length petal.width species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

6.8 Rename to lower specific columns

```
iris %>%
    select_at(vars(Species, Petal.Length), tolower) %>%
    head(3)
```

```
##   species petal.length
## 1  setosa          1.4
## 2  setosa          1.4
## 3  setosa          1.3
```

6.9 Add name to a nameless column

```
library(tidyverse)
mtcars <- mtcars %>%
    as_tibble(rownames = "cars")
```

6.10 Add column

```
library(tibble)
iris %>%
    add_column(ob_no = 1:150) %>%
    head(5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species ob_no
## 1          5.1         3.5          1.4         0.2  setosa     1
## 2          4.9         3.0          1.4         0.2  setosa     2
## 3          4.7         3.2          1.3         0.2  setosa     3
## 4          4.6         3.1          1.5         0.2  setosa     4
## 5          5.0         3.6          1.4         0.2  setosa     5
```

```
iris %>%
    as_tibble() %>%
    head(3)
```

```
## # A tibble: 3 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
```

```
library(gapminder)
summary(gapminder)
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

```
str(gapminder)
```

```
## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 163
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

6.11 Re-code observation (recode)

change name of observation— mutate (variable=recode (variable, ‘old name’=‘new name’)))

```
gapminder %>%
    mutate(country = recode(country, India = "IND")) %>%
    filter(country == "IND") %>%
    head(3)
```

```
## # A tibble: 3 x 6
##   country continent  year lifeExp       pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
## 1 IND     Asia       1952    37.4 372000000      547.
## 2 IND     Asia       1957    40.2 409000000      590.
## 3 IND     Asia       1962    43.6 454000000      658.
```

6.12 Convert numeric values to a binary (Yes/No)

To convert all non-zero numeric values to “Yes” to convert zero values to “No”

```
df <- data.frame(name = c("saneesh", "sanusha", "appu", "jaru"),
    sex = c(2, 0, 5, 8))
df
```

```
##      name sex
## 1 saneesh   2
## 2 sanusha   0
## 3    appu   5
## 4    jaru   8
```

```
# convert numeric values to 'Yes'
df %>%
    mutate(sex1 = ifelse(sex != 0, "Yes", "No"))
```

```
##      name sex sex1
## 1 saneesh   2  Yes
## 2 sanusha   0   No
## 3    appu   5  Yes
## 4    jaru   8  Yes
```

```
df %>%
    mutate(sex1 = ifelse(sex != 0, "Male", "Female"))
```

```
##      name sex   sex1
## 1 saneesh   2   Male
## 2 sanusha   0 Female
## 3    appu   5   Male
## 4    jaru   8   Male
```

The `ifelse()` function is used to check whether each value in the "sex" column is non-zero. If it is, the value is replaced with "Yes". If not, the value is replaced with "No".

6.13 Select

```
gapminder %>%
    select(year, country, gdpPercap) %>%
    head(3)
```

```
## # A tibble: 3 x 3
##    year country     gdpPercap
##   <int> <fct>           <dbl>
## 1  1952 Afghanistan      779.
## 2  1957 Afghanistan      821.
## 3  1962 Afghanistan      853.
```

```
msleep %>%
    select(starts_with("sleep")) %>%
    head(3)
```

```
## # A tibble: 3 x 3
##   sleep_total sleep_rem sleep_cycle
##         <dbl>     <dbl>       <dbl>
## 1        12.1      NA            NA
## 2        17         1.8          NA
## 3        14.4       2.4          NA
```

6.14 Do not select

```
iris %>%
    select(-Sepal.Length, -Species) %>%
    head(3)
```

```
##   Sepal.Width Petal.Length Petal.Width
## 1         3.5          1.4         0.2
## 2         3.0          1.4         0.2
## 3         3.2          1.3         0.2
```

```
iris %>%
    select(-c(Sepal.Length)) %>%
    head(3)
```

```
##   Sepal.Width Petal.Length Petal.Width Species
## 1         3.5          1.4         0.2  setosa
## 2         3.0          1.4         0.2  setosa
## 3         3.2          1.3         0.2  setosa
```

```
iris %>%
    select(!Sepal.Length) %>%
    head(3)
```

```
##   Sepal.Width Petal.Length Petal.Width Species
## 1         3.5          1.4         0.2  setosa
## 2         3.0          1.4         0.2  setosa
## 3         3.2          1.3         0.2  setosa
```

6.15 ends_with

```
iris %>%
    select(ends_with("length")) %>%
    head(3)
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
```

6.16 starts_with

```
iris %>%
    select(starts_with("Sepal")) %>%
    head(3)
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
```

6.17 Filter

```
gapminder %>%
    select(year, country, lifeExp) %>%
    filter(country == "Eritrea", year > 1950) %>%
    head(3)
```

```
## # A tibble: 3 x 3
##    year country lifeExp
##   <int> <fct>     <dbl>
## 1  1952 Eritrea    35.9
## 2  1957 Eritrea    38.0
## 3  1962 Eritrea    40.2
```

```
gapminder %>%
    filter(country == "Canada") %>%
    head(3)  # from gapminder data filter country Canada and show only 2 observations
```

```
## # A tibble: 3 x 6
##   country continent  year lifeExp      pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Canada  Americas   1952    68.8 14785584    11367.
## 2 Canada  Americas   1957    70.0 17010154    12490.
## 3 Canada  Americas   1962    71.3 18985849    13462.
```

6.18 Except

```
gapminder %>%
    filter(country != "Oman") %>%
    head(3)  # from gapminder data filter all the other countries except Oman
```

```
## # A tibble: 3 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
```

6.19 Omit

```
iris %>%
    filter(Species != "setosa") %>%
    glimpse()
```

```
## Rows: 100
## Columns: 5
## $ Sepal.Length <dbl> 7.0, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5.0, 5.~
## $ Sepal.Width  <dbl> 3.2, 3.2, 3.1, 2.3, 2.8, 2.8, 3.3, 2.4, 2.9, 2.7, 2.0, 3.~
## $ Petal.Length <dbl> 4.7, 4.5, 4.9, 4.0, 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.~
## $ Petal.Width  <dbl> 1.4, 1.5, 1.5, 1.3, 1.5, 1.3, 1.6, 1.0, 1.3, 1.4, 1.0, 1.~
## $ Species      <fct> versicolor, versicolor, versicolor, versicolor, versicolo~
```

6.20 Filter multiple

```
iris %>%
    select(Species) %>%
    distinct(Species) %>%
    filter(Species %in% c("setosa", "versicolor")) %>%
    head(3)
```

```
##      Species
## 1     setosa
## 2 versicolor
```

using a vector, save the names as a vector and give it to `%in%`

```
target <- c("Hungary", "Iceland", "Mongolia")
gapminder %>%
    filter(country %in% target) %>%
    head(3)
```

```
## # A tibble: 3 x 6
##   country continent  year lifeExp      pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Hungary Europe     1952    64.0  9504000     5264.
## 2 Hungary Europe     1957    66.4  9839000     6040.
## 3 Hungary Europe     1962    68.0 10063000     7550.
```

```
friends <- data.frame(Names = c("Saneesh", "Appu", "Shruti",
    "Aradhana", "Arathi", "James Bond"), age = c(40, 9, 25, 25,
    25, 50))
# data frame is friends columns in friends are Names, Age,
# Height, etc.  Column Name have 'Saneesh', 'Appu',
# 'Shruti', 'Aradhana', 'Arathi', 'James Bond' We want to
# filter information related to Sanees and James Bond only,
# so we created a vector with these names in it.

target <- c("Appu", "James Bond")  #and then

friends %>%
    filter(Names %in% target)
```

```
##        Names age
## 1       Appu   9
## 2 James Bond  50
```

```
# or
friends %>%
    filter(Names == "Appu" | Names == "James Bond")
```

```
##        Names age
## 1       Appu   9
## 2 James Bond  50
```

```
# or
friends %>%
    filter(Names %in% c("Appu", "James Bond"))
```

```
##        Names age
## 1       Appu   9
## 2 James Bond  50
```

6.21 omit multiple

```
iris %>%
    filter(!Species %in% c("setosa", "versicolor")) %>%
    glimpse()
```

```
## Rows: 50
## Columns: 5
## $ Sepal.Length <dbl> 6.3, 5.8, 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.~
## $ Sepal.Width  <dbl> 3.3, 2.7, 3.0, 2.9, 3.0, 3.0, 2.5, 2.9, 2.5, 3.6, 3.2, 2.~
## $ Petal.Length <dbl> 6.0, 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1, 5.1, 5.~
## $ Petal.Width  <dbl> 2.5, 1.9, 2.1, 1.8, 2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2.0, 1.~
## $ Species      <fct> virginica, virginica, virginica, virginica, virginica, vi~
```

6.22 filter between

```
iris %>%
    filter(Petal.Width >= 2 & Petal.Width <= 5) %>%
    glimpse()
```

```
## Rows: 29
## Columns: 5
## $ Sepal.Length <dbl> 6.3, 7.1, 6.5, 7.6, 7.2, 6.5, 6.8, 5.7, 5.8, 6.4, 7.7, 7.~
## $ Sepal.Width  <dbl> 3.3, 3.0, 3.0, 3.0, 3.6, 3.2, 3.0, 2.5, 2.8, 3.2, 3.8, 2.~
## $ Petal.Length <dbl> 6.0, 5.9, 5.8, 6.6, 6.1, 5.1, 5.5, 5.0, 5.1, 5.3, 6.7, 6.~
## $ Petal.Width  <dbl> 2.5, 2.1, 2.2, 2.1, 2.5, 2.0, 2.1, 2.0, 2.4, 2.3, 2.2, 2.~
## $ Species      <fct> virginica, virginica, virginica, virginica, virginica, vi~
```

6.23 filter matching

```
library(tidyverse)
library(dplyr)
mtcars <- mtcars %>%
    rownames_to_column
mtcars %>%
    filter(str_detect(rowname, "Merc")) %>%
    head(3)  # filter only 'Merc'
```

```
## # A tibble: 0 x 13
## # i 13 variables: rowname <chr>, cars <chr>, mpg <dbl>, cyl <dbl>, disp <dbl>,
## #   hp <dbl>, drat <dbl>, wt <dbl>, qsec <dbl>, vs <dbl>, am <dbl>, gear <dbl>,
## #   carb <dbl>
```

```
mtcars %>%
    filter(!str_detect(rowname, "Merc")) %>%
    head(3)  # filter everything except 'Merc'
```

```
## # A tibble: 3 x 13
##   rowname cars         mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear
##   <chr>   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1       Mazda RX4   21       6   160   110  3.9   2.62  16.5     0     1     4
## 2 2       Mazda RX4~  21       6   160   110  3.9   2.88  17.0     0     1     4
## 3 3       Datsun 710  22.8     4   108    93  3.85  2.32  18.6     1     1     4
## # i 1 more variable: carb <dbl>
```

6.24 filter distinct

To remove or exclude all entries in the “name” column of your data frame that have 1 in the “pref” column, you can use the `filter()` and `distinct()` functions from the dplyr

```
df <- data.frame(name = c("a", "a", "b", "c", "d", "a", "d"),
    pref = c(1, 2, 2, 1, 3, 4, 1))

df
```

```
##   name pref
## 1    a    1
## 2    a    2
## 3    b    2
## 4    c    1
## 5    d    3
## 6    a    4
## 7    d    1
```

```
df %>%
    group_by(name) %>%
    filter(!any(pref == 1)) %>%
    ungroup()
```

```
## # A tibble: 1 x 2
##   name   pref
##   <chr> <dbl>
## 1 b         2
```

or, if you have multiple rows with the same name but different values in the “pref” column, the code above will remove all rows with that name if any of them have 1 in the “pref” column. If you want to remove only the rows with 1 in the “pref” column, but keep the other rows with the same name, you can modify the code as follows:

```
df %>%
    group_by(name) %>%
    filter(!any(pref == 1)) %>%
    ungroup()
```

```
## # A tibble: 1 x 2
##   name   pref
##   <chr> <dbl>
## 1 b         2
```

6.25 Pull

```
iris %>%
    pull(Species) %>%
    head(3)  # returns vector values
```

```
## [1] setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
iris %>%
    select(Species) %>%
    head(3)  # returns a table with one column
```

```
##   Species
## 1  setosa
## 2  setosa
## 3  setosa
```

```
iris %>%
    select(everything()) %>%
    head(3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

6.26 multiple conditions

```r
gapminder %>%
    filter(country == "Oman" & year > 1980 & year <= 2000) %>%
    head(4)
```

```
## # A tibble: 4 x 6
##   country continent  year lifeExp     pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>   <int>     <dbl>
## 1 Oman    Asia       1982    62.7 1301048    12955.
## 2 Oman    Asia       1987    67.7 1593882    18115.
## 3 Oman    Asia       1992    71.2 1915208    18617.
## 4 Oman    Asia       1997    72.5 2283635    19702.
```

```r
gapminder %>%
    select(country, year) %>%
    filter(year >= 1980, country == "India" | country == "Oman" |
        country == "Canada") %>%
    head(4)
```

```
## # A tibble: 4 x 2
##   country  year
##   <fct>   <int>
## 1 Canada   1982
## 2 Canada   1987
## 3 Canada   1992
## 4 Canada   1997
```

```r
gapminder %>%
    filter(country != "Oman") %>%
    head(3)  # from gapminder data filter all the other countires exept Oman
```

```
## # A tibble: 3 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
```

6.27 drop

```r
gapminder %>%
    select(-year, -pop) %>%
    head(5)
```

```
## # A tibble: 5 x 4
##   country     continent lifeExp gdpPercap
##   <fct>       <fct>       <dbl>     <dbl>
## 1 Afghanistan Asia         28.8      779.
## 2 Afghanistan Asia         30.3      821.
## 3 Afghanistan Asia         32.0      853.
## 4 Afghanistan Asia         34.0      836.
## 5 Afghanistan Asia         36.1      740.
```

6.28 group by & summarise

```
gapminder %>%
    filter(year == 2007) %>%
    group_by(country) %>%
    summarise(meanLE = mean(lifeExp)) %>%
    arrange(meanLE, decreasing = TRUE) %>%
    head(3)
```

```
## # A tibble: 3 x 2
##   country    meanLE
##   <fct>       <dbl>
## 1 Swaziland    39.6
## 2 Mozambique   42.1
## 3 Zambia       42.4
```

```
gapminder %>%
    group_by(country) %>%
    summarise(minLE = min(lifeExp)) %>%
    arrange(minLE, decreasing = FALSE) %>%
    head(3)
```

```
## # A tibble: 3 x 2
##   country     minLE
##   <fct>       <dbl>
## 1 Rwanda       23.6
## 2 Afghanistan  28.8
## 3 Gambia       30
```

grouped by continent, then summarise two things, first `n=n()` number of rows in which each continent are or the size of each group, then the mean of the mean of the lifeExp variable.

```
gapminder %>%
    group_by(continent) %>%
    summarise(n = n(), meanLife = mean(lifeExp))
```

```
## # A tibble: 5 x 3
##   continent     n meanLife
##   <fct>     <int>    <dbl>
## 1 Africa      624     48.9
## 2 Americas    300     64.7
## 3 Asia        396     60.1
## 4 Europe      360     71.9
## 5 Oceania      24     74.3
```

```
gapminder %>%
    group_by(continent) %>%
    summarise(PopConti = sum(pop))
```

```
## # A tibble: 5 x 2
##   continent    PopConti
##   <fct>           <dbl>
## 1 Africa     6187585961
## 2 Americas   7351438499
## 3 Asia      30507333901
## 4 Europe     6181115304
## 5 Oceania     212992136
```

```
pets <- data.frame(names = c(rep("saneesh", 3), rep("appu", 2),
    "sanusha"), pet = c(rep("dog", 3), rep("cat", 2), "tiger"),
    number = c(2, 2, 5, 7, 8, 1), size = c(rep("medium", 2),
        rep("small", 3), "big"))

pets
```

```
##     names   pet number   size
## 1 saneesh   dog      2 medium
## 2 saneesh   dog      2 medium
## 3 saneesh   dog      5  small
## 4    appu   cat      7  small
## 5    appu   cat      8  small
## 6 sanusha tiger      1    big
```

```
pets %>%
    group_by(pet, size) %>%
    summarise(totalpet = sum(number))
```

```
## ‘summarise()‘ has grouped output by ’pet’. You can override using the ‘.groups‘
## argument.
```

```
## # A tibble: 4 x 3
## # Groups:   pet [3]
##   pet   size   totalpet
##   <chr> <chr>     <dbl>
## 1 cat   small        15
## 2 dog   medium        4
## 3 dog   small         5
## 4 tiger big           1
```

6.29 grouping with conditions

If we want make a ‘new column’ with values from ‘number’ only if ‘sp.name’ ‘a’ or any other values has the following responses ‘young’ and ‘adult’, if not enter 0 in the ‘new column’.

You need to have groups with any of stage == “young” & “adult” (group level conditions) and stage == “adult” (row-level condition):

6.30 summarise

```
library(tidyverse)
plot <- c(rep(1, 2), rep(2, 4), rep(3, 3))
bird <- c("a", "b", "a", "b", "c", "d", "a", "b", "c")
area <- c(rep(10, 2), rep(5, 4), rep(15, 3))

birdlist <- data.frame(plot, bird, area)
birdlist
```

```
##   plot bird area
## 1    1    a   10
## 2    1    b   10
## 3    2    a    5
## 4    2    b    5
## 5    2    c    5
## 6    2    d    5
## 7    3    a   15
## 8    3    b   15
## 9    3    c   15
```

```
# summarize the following data frame to a summary table.
# option 1
birdlist %>%
    group_by(plot) %>%
    summarise(bird = n(), area = unique(area))
```

```
## # A tibble: 3 x 3
##    plot  bird  area
##   <dbl> <int> <dbl>
## 1     1     2    10
## 2     2     4     5
## 3     3     3    15
```

```
# option 2
birdlist %>%
    count(plot, area, name = "bird")
```

```
##   plot area bird
## 1    1   10    2
## 2    2    5    4
## 3    3   15    3
```

```
gapminder %>%
    summarise(mean(lifeExp))
```

```
## # A tibble: 1 x 1
##   ‘mean(lifeExp)‘
##             <dbl>
## 1            59.5
```

```
gapminder %>%
    summarise(range(lifeExp))
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 2 x 1
##   `range(lifeExp)`
##              <dbl>
## 1             23.6
## 2             82.6
```

```
gapminder %>%
    filter(country == "India") %>%
    group_by(country) %>%
    summarise(GDPmax = max(gdpPercap), GDPmin = min(gdpPercap),
        GDPmean = mean(gdpPercap))
```

```
## # A tibble: 1 x 4
##   country GDPmax GDPmin GDPmean
##   <fct>    <dbl>  <dbl>   <dbl>
## 1 India    2452.   547.   1057.
```

6.31 remove duplicates from a column and summarise

```
df <- data.frame(name = c("a", "a", "b", "c"), seedling = c(1,
    0, 1, 0), adult = c(0, 5, 0, 1))

df_new <- df %>%
    group_by(name) %>%
    summarise(seedling = max(seedling, 0), adult = max(adult,
        0)) %>%
    ungroup()
```

6.32 find and remove duplicates from a dataframe

```
library(dplyr)
library(hablar)
```

```
##
## Attaching package: 'hablar'
## The following object is masked from 'package:forcats':
##
##     fct
```

```
## The following object is masked from 'package:dplyr':
##
##     na_if
```

```
## The following object is masked from 'package:tibble':
##
##     num
```

```
df <- tibble(a = c(1, 1, "a", 2, 2, 2, 4), b = c("a", "a", 1,
    "b", "b", "b", "c"))
df %>%
    print()
```

```
## # A tibble: 7 x 2
##   a     b
##   <chr> <chr>
## 1 1     a
## 2 1     a
## 3 a     1
## 4 2     b
## 5 2     b
## 6 2     b
## 7 4     c
```

```
df %>%
    find_duplicates()
```

```
## # A tibble: 5 x 2
##   a     b
##   <chr> <chr>
## 1 1     a
## 2 1     a
## 3 2     b
## 4 2     b
## 5 2     b
```

```
df %>%
    distinct() %>%
    print()
```

```
## # A tibble: 4 x 2
##   a     b
##   <chr> <chr>
## 1 1     a
## 2 a     1
## 3 2     b
## 4 4     c
```

6.33 count/summarize

6.34 count name column

```
iris %>%
    count(Species, name = "how many")
```

```
##      Species how many
## 1     setosa       50
## 2 versicolor       50
## 3  virginica       50
```

```
mtcars %>%
    count(am, name = "number") %>%
    as_tibble()
```

```
## # A tibble: 2 x 2
##      am number
##   <dbl>  <int>
## 1     0     19
## 2     1     13
```

```
mtcars %>%
    count(gear, name = "no. gear")
```

```
## # A tibble: 3 x 2
##    gear ‘no. gear‘
##   <dbl>      <int>
## 1     3         15
## 2     4         12
## 3     5          5
```

6.35 New column with paste

```
library(dplyr)

# Create a data frame with two columns named 'a' and 'b'
df <- data.frame(a = c("red", "blue", "green"), b = c(1, 2, 3))

# Create a new column named 'c' by combining values from
# 'a' and 'b'
df <- df %>%
    mutate(c = paste(a, b, sep = "_"))
```

6.36 Count birds

```
plot <- c(rep(1, 2), rep(2, 4), rep(3, 3))
bird <- as.factor(c("a", "b", "a", "b", "c", "d", "a", "b", "c"))
area <- c(rep(10, 2), rep(5, 4), rep(15, 3))

birdlist <- data.frame(plot, bird, area)
birdlist
```

```
##   plot bird area
## 1    1    a   10
## 2    1    b   10
## 3    2    a    5
## 4    2    b    5
## 5    2    c    5
## 6    2    d    5
## 7    3    a   15
## 8    3    b   15
## 9    3    c   15
```

```
# birdlist %>%  group_by(plot, area) %>%  mutate(count(bird))


birdlist %>%
  group_by(plot, area) %>%
  summarise(bird = n(), .groups = "drop")
```

```
## # A tibble: 3 x 3
##    plot  area  bird
##   <dbl> <dbl> <int>
## 1     1    10     2
## 2     2     5     4
## 3     3    15     3
```

```
# (dplyr::summarise)like this
# to summarize of a column with reference to two other variables.
```

6.37 count sites

```
treatment <- c(rep("ab", 2), rep("bgrnf", 8), rep("bgpnf", 4))
site <- c(
  "ab1",
  "ab2",
  rep("bgrnf1", 3),
  rep("bgrnf2", 2),
  "bgrnf3",
  "bgrnf4",
  "bgrnf5",
  rep("bgpnf1", 2),
  rep("bgpnf2", 2)
)
```

```r
data <- data.frame(treatment, site)

# to find the site per each treatment
data %>% group_by(treatment) %>% count(treatment, name = "#sites")
```

```
## # A tibble: 3 x 2
## # Groups:   treatment [3]
##   treatment `#sites`
##   <chr>        <int>
## 1 ab               2
## 2 bgpnf            4
## 3 bgrnf            8
```

6.38 count within years

```r
year <-  c(rep(2000, 4),
  rep(2001, 4),
  rep(2002, 4)
)
site <- c(rep("a", 3),
  rep("b", 3),
  rep("c", 3),
  rep("d", 3)
)

fire <- c("yes", "no", "yes",
  "yes", "no", "no",
  "yes", "yes", "yes",
  "yes", "yes", "yes")

df <- data.frame(year, site, fire)

df %>%
  group_by(site) %>%
  summarize(
    Burnt_once = sum(fire == "yes" &
      year %in% c(2000, 2001, 2002)) == 1,
    Burnt_twice = sum(fire == "yes" &
      year %in% c(2000, 2001, 2002)) == 2,
    Burnt_thrice = sum(fire == "yes" &
      year %in% c(2000, 2001, 2002)) == 3
  ) %>%  mutate(
    Burnt_once = ifelse(Burnt_once, 1, 0),
    Burnt_twice = ifelse(Burnt_twice, 1, 0),
    Burnt_thrice = ifelse(Burnt_thrice, 1, 0)
  ) %>%  summarise(across(where(is.numeric),    ~ sum(.x,   na.rm = TRUE)))
```

```
## # A tibble: 1 x 3
##   Burnt_once Burnt_twice Burnt_thrice
##        <dbl>       <dbl>        <dbl>
## 1          1           1            2
```

```
# df %>%
#   group_by(site) %>%
#   summarize(
#     Burnt_once = sum(fire == "yes" &
#                        year %in% c(2000, 2001, 2002)) == 1, # in these years look for 1 'yes'
#     Burnt_twice = sum(fire == "yes" &
#                         year %in% c(2000, 2001, 2002)) == 2, # in these years look for 2 'yes'
#     Burnt_thrice = sum(fire == "yes" &
#                          year %in% c(2000, 2001, 2002)) == 3 # in these years look for 3 'yes'
#   ) %>% # returns a logical vector
#   mutate(
#     Burnt_once = ifelse(Burnt_once, 1, 0),
#     Burnt_twice = ifelse(Burnt_twice, 1, 0),
#     Burnt_thrice = ifelse(Burnt_thrice, 1, 0)
#   ) %>% # convert logical response to numeric
#   summarise( # summarise data
#     across( # specifycolumns
#       where(is.numeric), # select columns with numeric ones
#       ~ sum( # selected column using the ~ formula notation
#         .x, # for each selected columns
#         na.rm = TRUE))) # remove any missing values before calculating the sum
```

6.39 case when new column

```
library(dplyr)
library(stringr)
feedback <-
  c("good_book", "good_read", "for knowledge", "adventure")
book <- c("Ramayana", "Bible", "Encyclopedia", "Mbharatha")

df <- data.frame(book, feedback)

df %>%
  mutate(response = case_when(str_starts(feedback, "good") ~ "good")) %>%
  select(book, response) %>% as_tibble()
```

```
## # A tibble: 4 x 2
##   book         response
##   <chr>        <chr>
## 1 Ramayana     good
## 2 Bible        good
## 3 Encyclopedia <NA>
## 4 Mbharatha    <NA>
```

6.40 Case when

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
iris %>%
    mutate(species.code = case_when(Species == "setosa" ~ 1,
        Species == "versicolor" ~ 2, Species == "virginica" ~
            3)) %>%
    head()
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species species.code
## 1          5.1         3.5          1.4         0.2  setosa            1
## 2          4.9         3.0          1.4         0.2  setosa            1
## 3          4.7         3.2          1.3         0.2  setosa            1
## 4          4.6         3.1          1.5         0.2  setosa            1
## 5          5.0         3.6          1.4         0.2  setosa            1
## 6          5.4         3.9          1.7         0.4  setosa            1
```

6.41 Use of if else

```
library(dplyr)

iris %>%
    select(Species) %>%
    slice_sample(n = 10) %>%
    mutate(code = if_else(Species == "setosa", 1, 0)  # you might see different result!
)
```

```
##       Species code
## 1   virginica    0
## 2  versicolor    0
## 3      setosa    1
## 4   virginica    0
## 5   virginica    0
## 6      setosa    1
## 7      setosa    1
## 8      setosa    1
## 9   virginica    0
## 10     setosa    1
```

6.42 Separate text to columns

```
df <- data.frame(films = c("Spider_man", "James_bond", "Iron_man",
    "Bat_man"))
df
```

```
##        films
## 1 Spider_man
## 2 James_bond
## 3   Iron_man
## 4    Bat_man
```

```
df1 <- df %>%
    separate(films, c("a", "b"), sep = "([_])")
df1
```

```
##        a    b
## 1 Spider  man
## 2  James bond
## 3   Iron  man
## 4    Bat  man
```

6.43 Unite text

```
df1 %>%
    unite("names", a:b, remove = FALSE)
```

```
##        names      a    b
## 1 Spider_man Spider  man
## 2 James_bond  James bond
## 3   Iron_man   Iron  man
## 4    Bat_man    Bat  man
```

6.44 Join

```
df1 <- data.frame(id = c(1:4), films = c("Spider_man", "James_bond",
    "Iron_man", "Bat_man"))

df2 <- data.frame(id = c(1:4), country = rep("us", 4))
df3 <- left_join(df1, df2, by = "id")
```

6.45 Spread & gather

We are making a wide format from long format in the first example. The second example is to make a long format from wide.

```
# the following is already in long format
classdata <- data.frame(
  studentname = c("captian", "ant", "james", "spider", "tony", "bat", "wonder"),
  subject = c("math", "his", "math", "geo", "his", "geo", "math"),
  grade = c("A+", "B", "B", "A+", "C", "B+", "C")
)

classdata %>% head()
```

```
##   studentname subject grade
## 1     captian    math    A+
## 2         ant     his     B
## 3       james    math     B
## 4      spider     geo    A+
## 5        tony     his     C
## 6         bat     geo    B+
```

```
wide.class <- spread(classdata,  subject,  grade)
# classdata= name of the data frame
# subject= new columns to be made
# grade= values to go into new columns


head(wide.class)
```

```
##   studentname  geo  his math
## 1         ant <NA>    B <NA>
## 2         bat   B+ <NA> <NA>
## 3     captian <NA> <NA>   A+
## 4       james <NA> <NA>    B
## 5      spider   A+ <NA> <NA>
## 6        tony <NA>    C <NA>
```

```
gather(wide.class, subject,  grade, geo, his, math) %>%
  drop_na()
```

```
##   studentname subject grade
## 1         bat     geo    B+
## 2      spider     geo    A+
## 3         ant     his     B
## 4        tony     his     C
## 5     captian    math    A+
## 6       james    math     B
## 7      wonder    math     C
```

```
# wide.class= name of the data frame
# subject= name of the column to put data into
# grade= name of the column to put value into
# geo, his, math= from where values has to be gathered
```

6.46 Join rows

bind rows

```
df1 <-
  data.frame(
    id = c(1:4),
    films = c("Spider_man", "James_bond", "Iron_man", "Bat_man")
  )
df2 <-
  data.frame(
    id = c(5:8),
    films = c("King Cong", "Silence of the lambs", "Intersteller", "Gravity")
  )
dplyr::bind_rows(df1, df2)
```

```
##   id                 films
## 1  1            Spider_man
## 2  2            James_bond
## 3  3              Iron_man
## 4  4               Bat_man
## 5  5             King Cong
## 6  6 Silence of the lambs
## 7  7          Intersteller
## 8  8               Gravity
```

6.47 Across

For multiple variables

```
library(tidyverse)
srno <- c(1:2)
film <- c("arabica", "robust")
rate <- c("good", "better")
lang_Eng <- c("yes", "yes")

films <- data.frame(srno, film, rate, lang_Eng)

str(films)
```

```
## 'data.frame':    2 obs. of  4 variables:
##  $ srno    : int  1 2
##  $ film    : chr  "arabica" "robust"
##  $ rate    : chr  "good" "better"
##  $ lang_Eng: chr  "yes" "yes"
```

```
films <- films %>%
  mutate(across(c(rate, lang_Eng), as.factor))

str(films)
```

```
## 'data.frame':    2 obs. of  4 variables:
##  $ srno    : int  1 2
##  $ film    : chr  "arabica" "robust"
##  $ rate    : Factor w/ 2 levels "better","good": 2 1
##  $ lang_Eng: Factor w/ 1 level "yes": 1 1
```

6.48 Everthing

Select a key variable and everything or every other columns.

```
library(gapminder)
gapminder %>%
    select(pop, everything()) %>%
    head(3)
```

```
## # A tibble: 3 x 6
##        pop country     continent  year lifeExp gdpPercap
##      <int> <fct>       <fct>     <int>   <dbl>     <dbl>
## 1  8425333 Afghanistan Asia       1952    28.8      779.
## 2  9240934 Afghanistan Asia       1957    30.3      821.
## 3 10267083 Afghanistan Asia       1962    32.0      853.
```

6.49 toupper and lower

```
library(stringr)

data <- data.frame(Dose.Cm = c("d1", "D2", "D3"), Len.km = c("High",
    "low", "Low"))
glimpse(data)
```

```
## Rows: 3
## Columns: 2
## $ Dose.Cm <chr> "d1", "D2", "D3"
## $ Len.km  <chr> "High", "low", "Low"
```

```
data %>%
    mutate(Dose.Cm = tolower(Dose.Cm), Len.km = toupper(Len.km))
```

```
##   Dose.Cm Len.km
## 1      d1   HIGH
## 2      d2    LOW
## 3      d3    LOW
```

6.50 factor

```
data <- data.frame(Dose.Cm = c("d1", "D2", "D3"), Len.km = c("high",
    "low", "medium"))
data <- data %>%
    mutate(len = as.factor(Len.km))

glimpse(data)
```

```
## Rows: 3
## Columns: 3
## $ Dose.Cm <chr> "d1", "D2", "D3"
## $ Len.km  <chr> "high", "low", "medium"
## $ len     <fct> high, low, medium
```

6.51 change order of factor

```
data %>%
    mutate(len = fct_relevel(len, c("low", "medium", "high")))
```

```
##   Dose.Cm Len.km    len
## 1      d1   high   high
## 2      D2    low    low
## 3      D3 medium medium
```

6.52 parse_number

This drops any non-numeric characters before or after the first number. The grouping mark specified by the locale is ignored inside the number.

```
library(tidyverse)
class <- c("8th", "9th", "10th")
students <- c("25-30", "35-41", "21-28")
school <- data.frame(class, students)
school
```

```
##   class students
## 1   8th    25-30
## 2   9th    35-41
## 3  10th    21-28
```

```
glimpse(school)  # notice students is a binned variable it is a not a numeric.
```

```
## Rows: 3
## Columns: 2
## $ class    <chr> "8th", "9th", "10th"
## $ students <chr> "25-30", "35-41", "21-28"
```

```
school %>%
    mutate(students = parse_number(students)) %>%
    glimpse()
```

```
## Rows: 3
## Columns: 2
## $ class    <chr> "8th", "9th", "10th"
## $ students <dbl> 25, 35, 21
```

```
school %>%
    mutate(students = parse_number(students))
```

```
##   class students
## 1   8th       25
## 2   9th       35
## 3  10th       21
```

```
# now students because number with first value of the
# column
```

6.53 pivot longer

```
library(tidyverse)

rawdata <- data.frame(species_1 = rnorm(n = 40, mean = 300, sd = 18.5),
    species_2 = rnorm(40, 305, 16.7))
data <- pivot_longer(data = rawdata, cols = species_1:species_2,
    names_to = "species", values_to = "weight")
```

6.54 Pivot wider

```
library(tidyverse)

df <- data.frame(name = c("saneesh", "sanusha", "appu", "jaru"),
    fav.no = c(11, 7, 20, 21), animal = c("human", "human", "human",
        "dog"))

df %>%
    pivot_wider(names_from = "animal", values_from = "fav.no")
```

```
## # A tibble: 4 x 3
##   name    human   dog
##   <chr>   <dbl> <dbl>
## 1 saneesh    11    NA
## 2 sanusha     7    NA
## 3 appu       20    NA
## 4 jaru       NA    21
```

```
# but when we have similar names in the grouping column
df1 <- data.frame(name = c("saneesh", "sanusha", "appu", "jaru",
    "saneesh"), fav.no = c(11, 7, 20, 21, 12), animal = c("human",
    "human", "human", "dog", "human"))

df1 %>%
    pivot_wider(names_from = "animal", values_from = "fav.no")
```

```
## Warning: Values from `fav.no` are not uniquely identified; output will contain
## list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = {summary_fun}` to summarise duplicates.
## * Use the following dplyr code to identify duplicates.
##   {data} |>
##   dplyr::summarise(n = dplyr::n(), .by = c(name, animal)) |>
##   dplyr::filter(n > 1L)
## # A tibble: 4 x 3
##   name    human     dog
##   <chr>   <list>    <list>
## 1 saneesh <dbl [2]> <NULL>
## 2 sanusha <dbl [1]> <NULL>
## 3 appu    <dbl [1]> <NULL>
## 4 jaru    <NULL>    <dbl [1]>
```

```
# because saneesh is repeated twice but with two fav.nos
# the solution is to add a row id, make pivot wide and get
# rid of the row id
df1 %>%
    mutate(id = row_number()) %>%
    group_by(name) %>%
    pivot_wider(names_from = "animal", values_from = "fav.no",
        values_fill = 0) %>%
    select(-id)
```

```
## # A tibble: 5 x 3
## # Groups:   name [4]
##   name    human   dog
##   <chr>   <dbl> <dbl>
## 1 saneesh    11     0
## 2 sanusha     7     0
## 3 appu       20     0
## 4 jaru        0    21
## 5 saneesh    12     0
```

6.55 Scoring numbers to likert

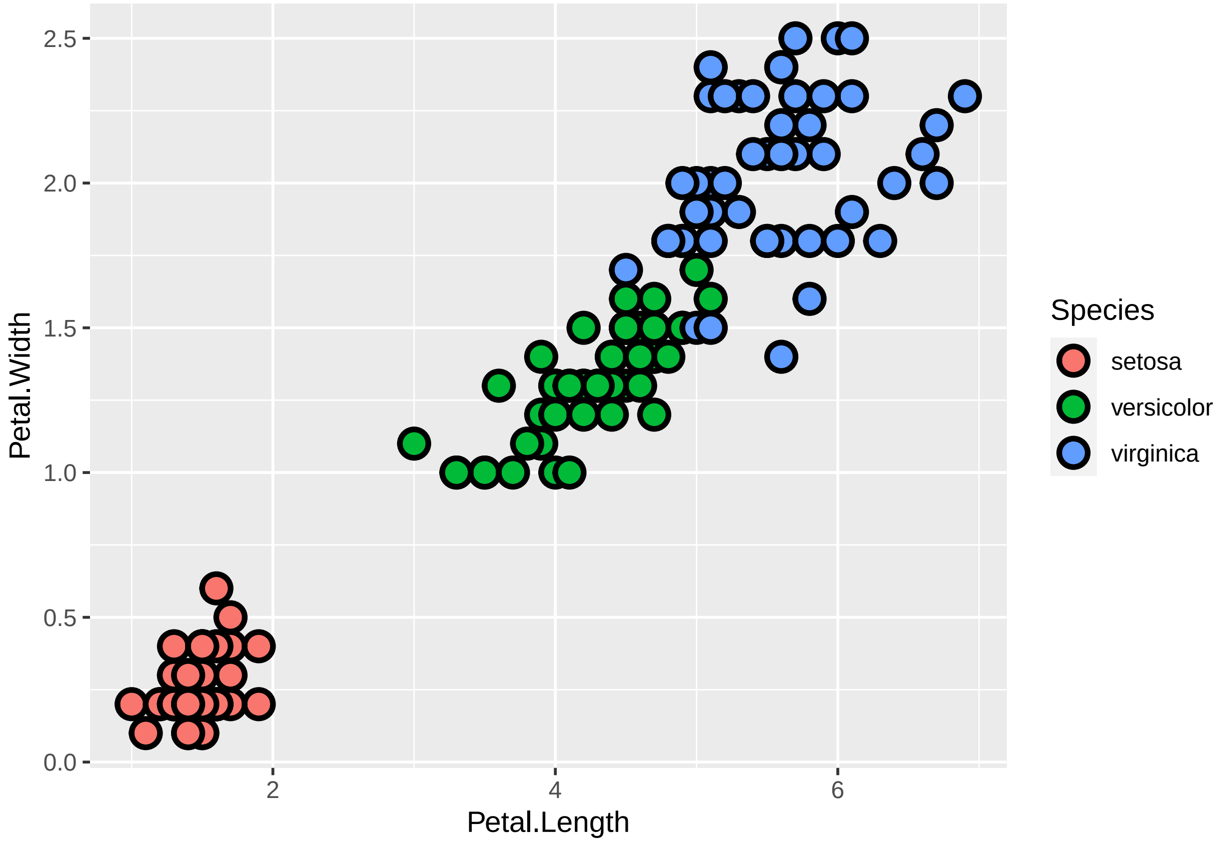
```
library(tidyverse)
numbers <- data.frame(test = seq(1:10))

numbers <-
  numbers %>% mutate(test1 = as.numeric(cut_number(test, 3)))
numbers <- numbers %>% mutate(test1 = as.factor(test1)) %>%
  mutate(test2 = recode(
    test1,
    "1" = "low",
    "2" = "medium",
    "3" = "high"
  ))
```

7 Ggplot

sthda

7.1 add border to points

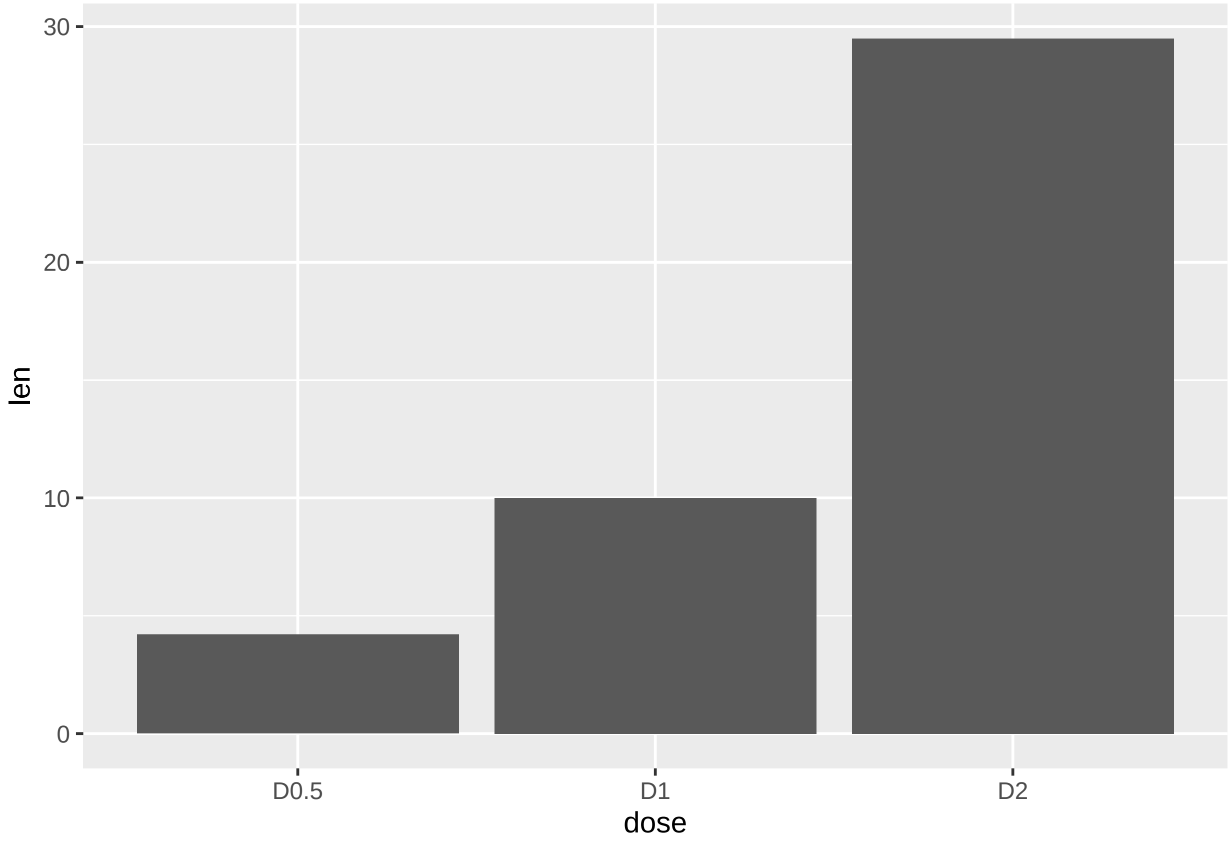
```
library(ggplot2)
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, fill = Species),
    alpha = 0.07) + geom_point(size = 4, shape = 21, color = "black",
    stroke = 1.5)
```



```
df <- data.frame(dose = c("D0.5", "D1", "D2"), len = c(4.2, 10,
    29.5))
```

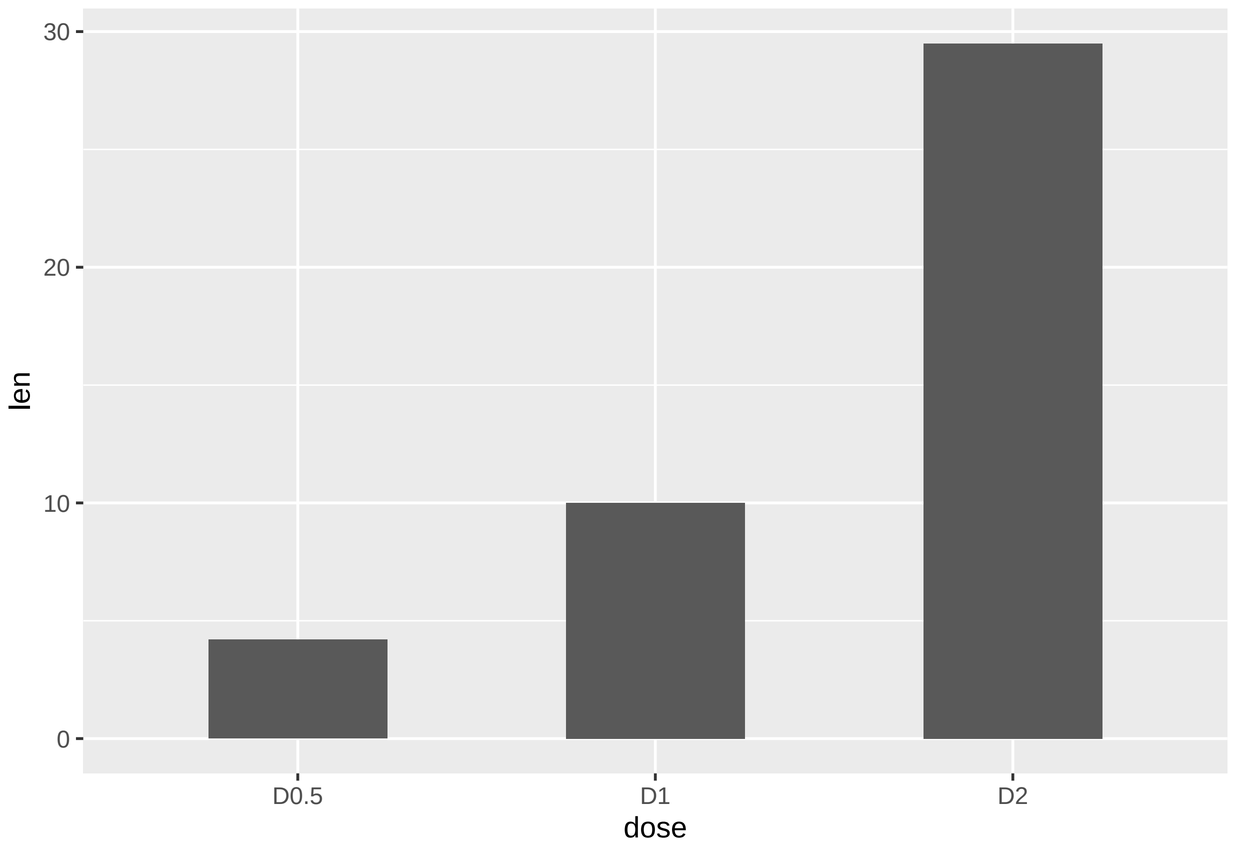
7.2 bar plot

```
library(ggplot2)
# Basic barplot
p <- ggplot(data = df, aes(x = dose, y = len)) + geom_bar(stat = "identity")
p
```

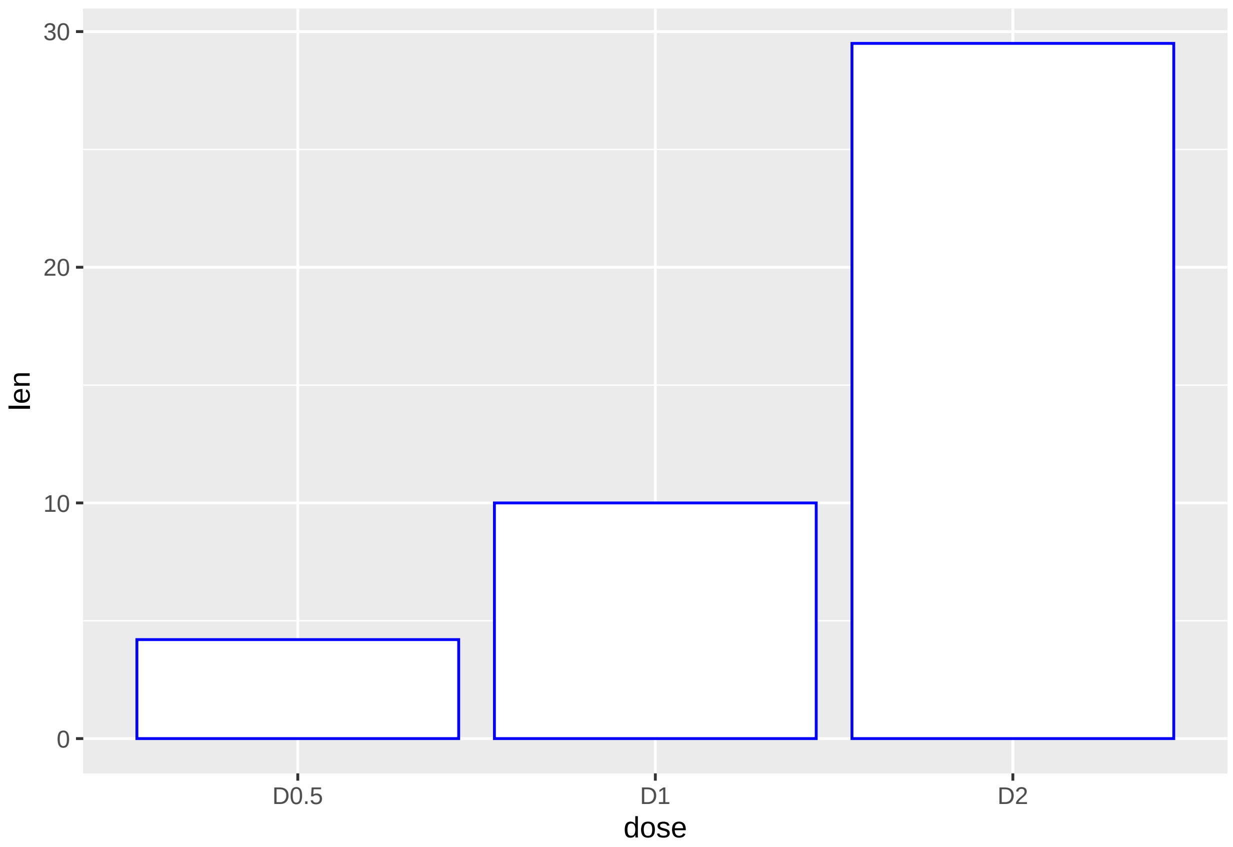


```
# Horizontal bar plot p + coord_flip()
```

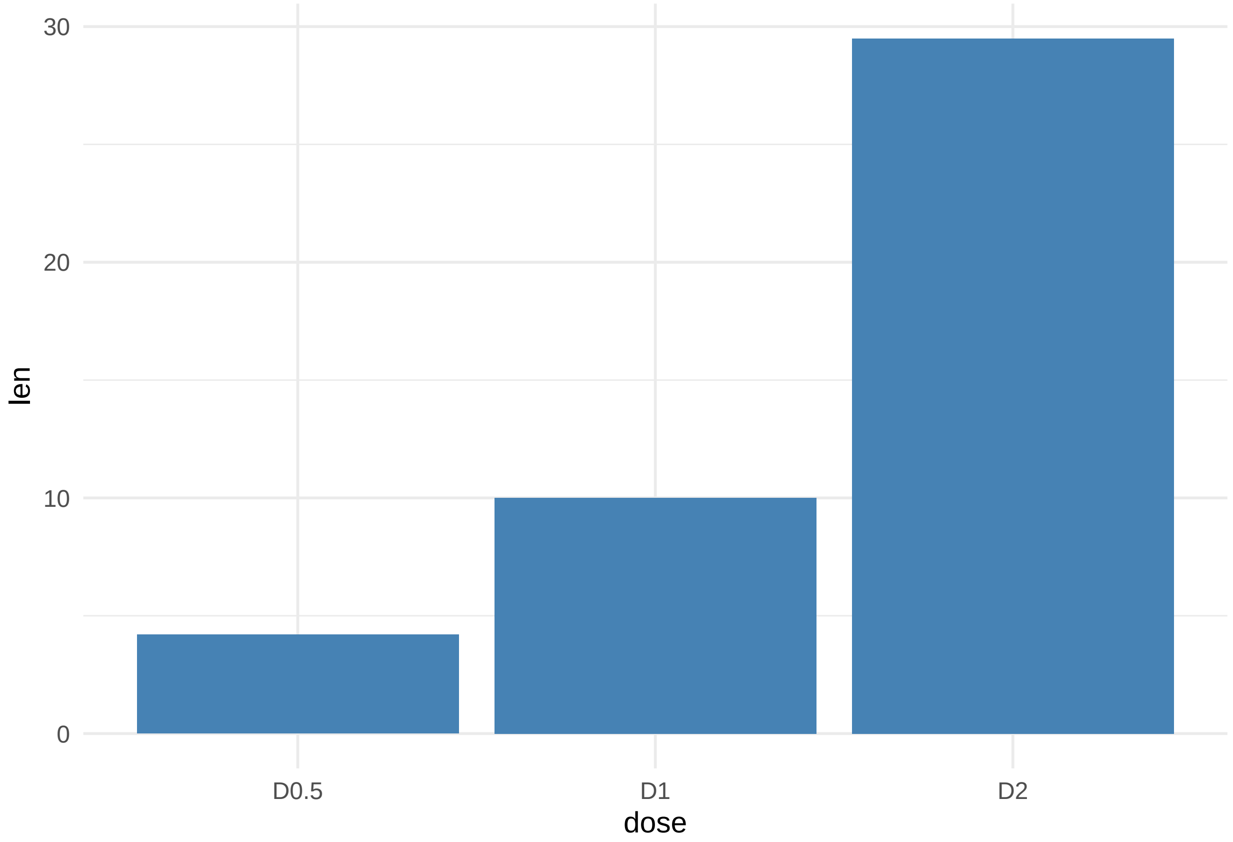
```
# Change the width of bars
ggplot(data = df, aes(x = dose, y = len)) + geom_bar(stat = "identity",
    width = 0.5)
```



```
# Change colors
ggplot(data = df, aes(x = dose, y = len)) + geom_bar(stat = "identity",
    color = "blue", fill = "white")
```

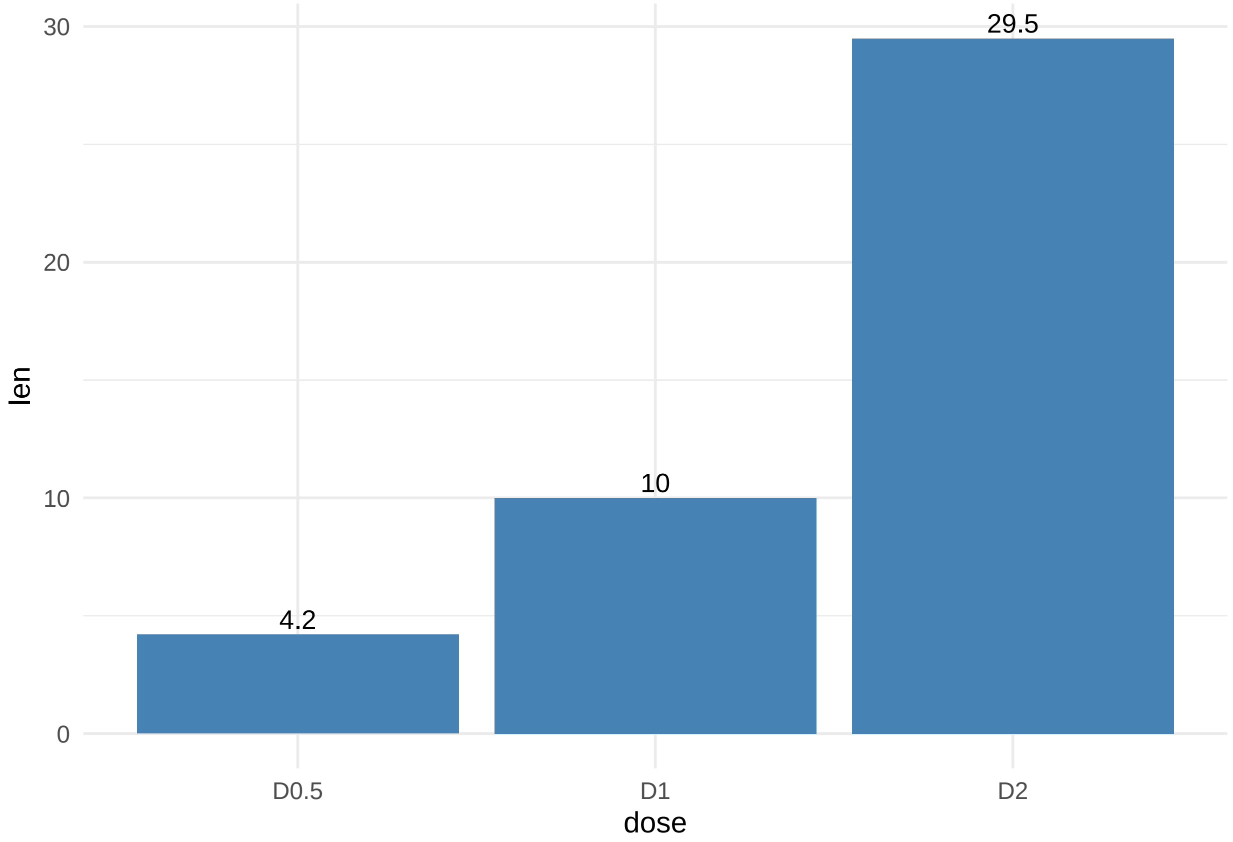


```
# Minimal theme + blue fill color
p <- ggplot(data = df, aes(x = dose, y = len)) + geom_bar(stat = "identity",
    fill = "steelblue") + theme_minimal()
p
```

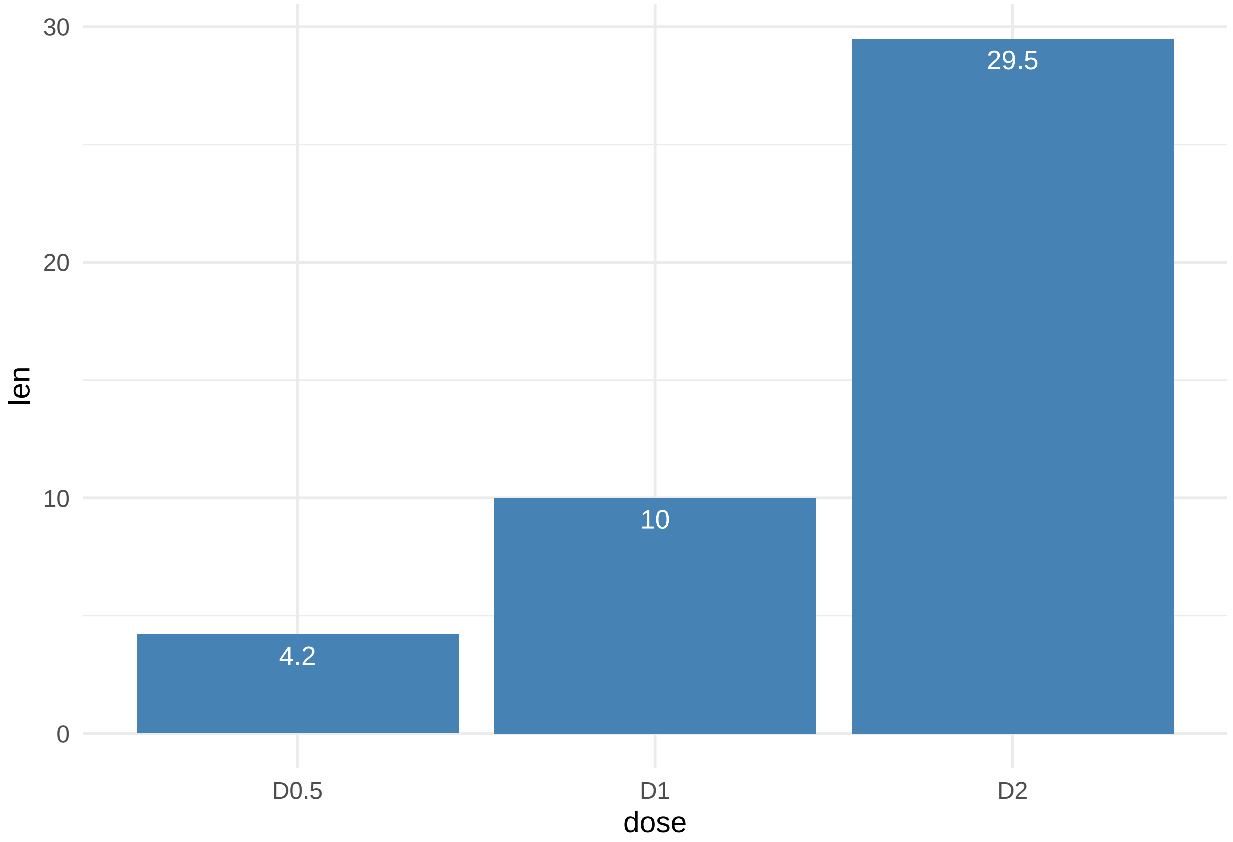


7.3 labels

```
# out side the bars
p + geom_text(aes(label = len), vjust = -0.3, size = 3.5) + theme_minimal()
```

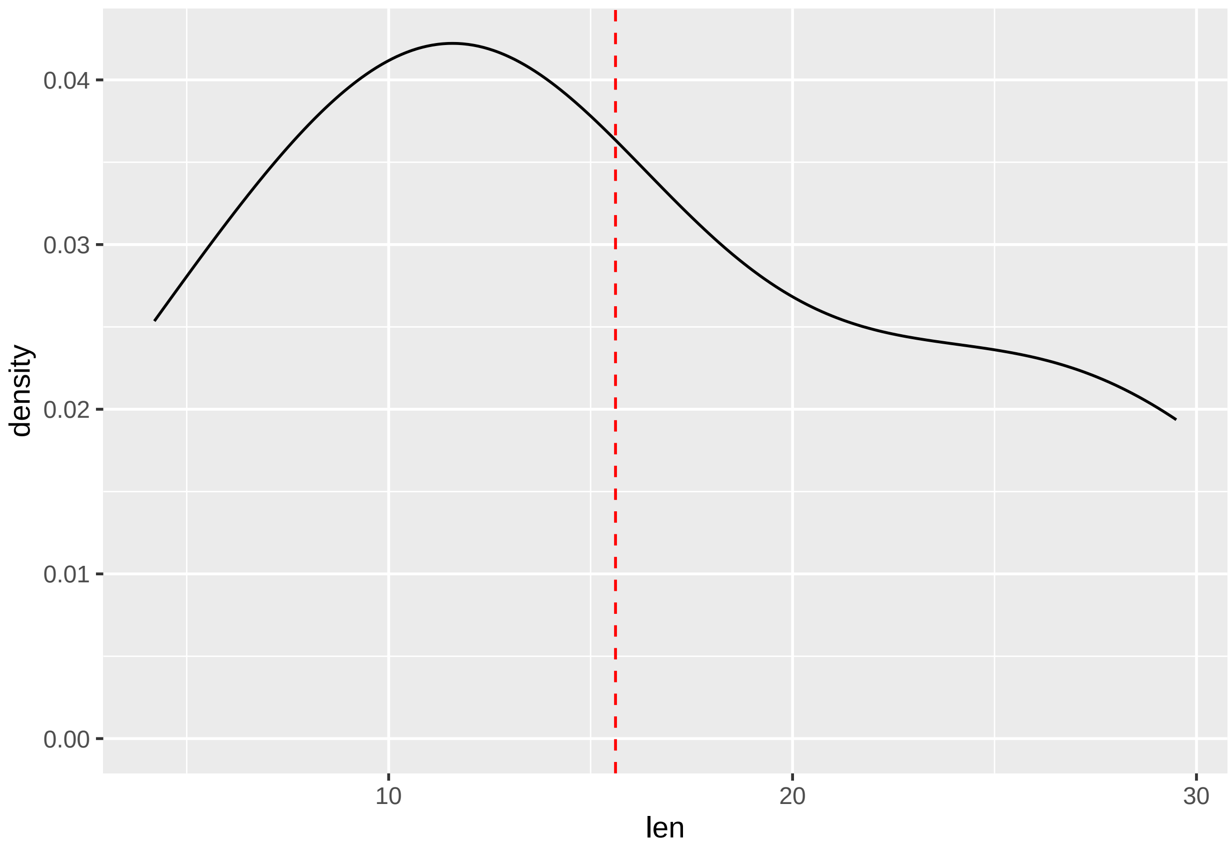


```r
p + geom_text(aes(label = len), vjust = 1.6, color = "white",
    size = 3.5) + theme_minimal()
```



7.4 geom_vline

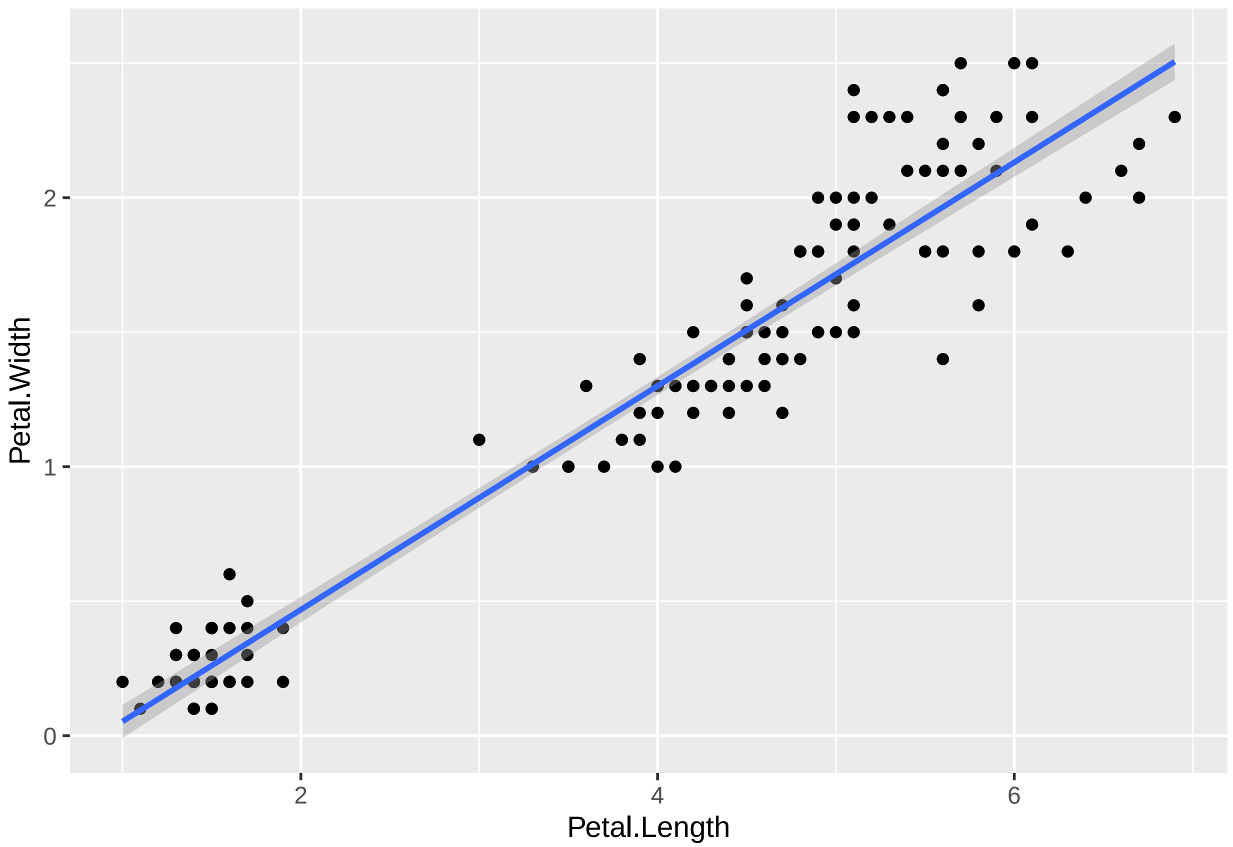
```
df <- data.frame(dose = c("D0.5", "D1", "D2", "pp", "kk", "rr"),
    len = c(4.2, 10, 29.5, 12, 15, 23))
library(ggplot2)

ggplot(df, aes(len)) + geom_density() + geom_vline(aes(xintercept = mean(len)),
    col = "red", linetype = "dashed")
```



7.5 scatter plot with lm

```
library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width)) + geom_point() +
    geom_smooth(method = "lm")
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



7.6 raincloud plot

```
library(ggdist)
library(tidyverse)
library(tidyquant)
```

```
## Loading required package: PerformanceAnalytics

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R’s lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or        #
## # source() into this session won’t work correctly.                             #
## #                                                                             #
## # Use stats::lag() to make sure you’re not using dplyr::lag(), or you can add #
## # conflictRules(’dplyr’, exclude = ’lag’) to your .Rprofile to stop           #
## # dplyr from breaking base R’s lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It’s protected by R’s namespace mechanism #
## # Set ‘options(xts.warn_dplyr_breaks_lag = FALSE)‘ to suppress this warning.  #
## #                                                                             #
## ###############################################################################
```

```
##
## Attaching package: ’xts’
```

```
## The following objects are masked from ’package:dplyr’:
##
##     first, last
```

```
##
## Attaching package: ’PerformanceAnalytics’
```

```
## The following object is masked from ’package:graphics’:
##
##     legend
```

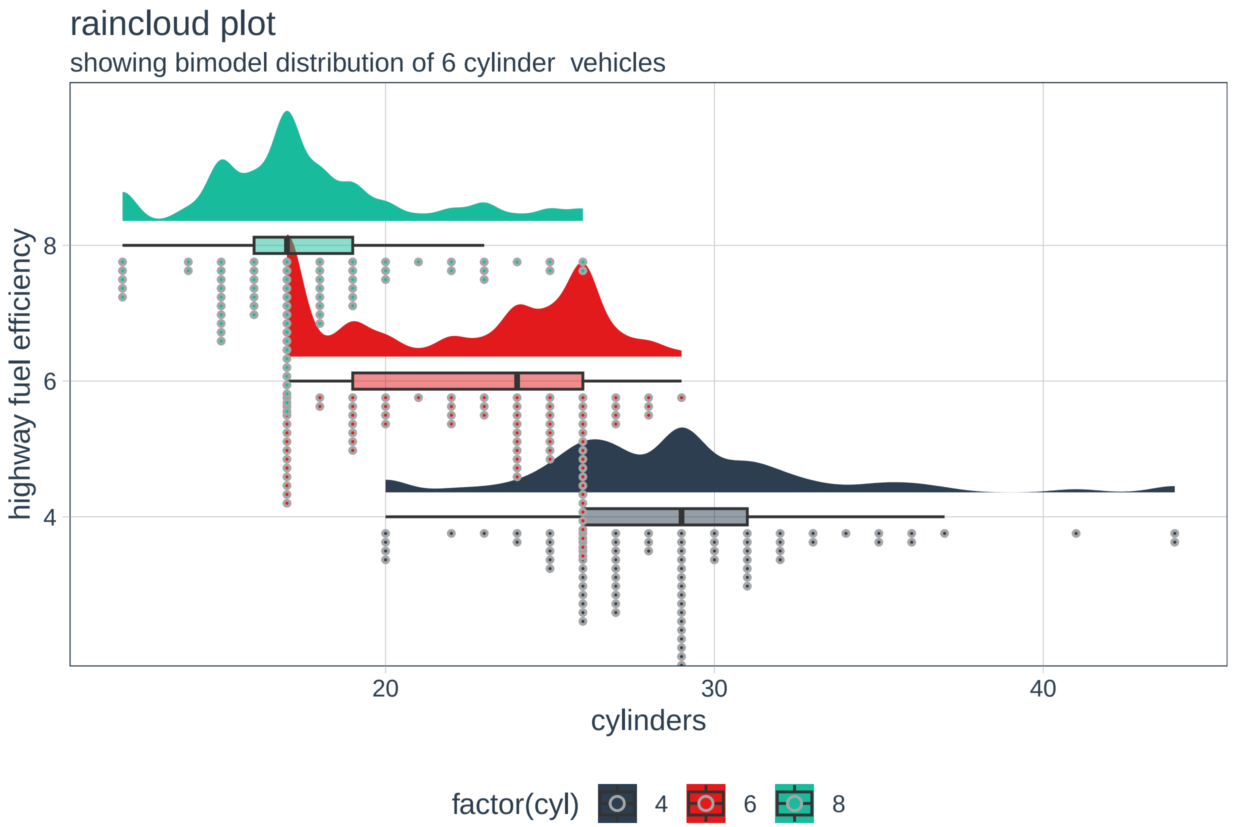
```
## Loading required package: quantmod
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by ’quantmod’:
##   method            from
##   as.zoo.data.frame zoo
```

```r
mpg %>% filter(cyl %in% c(4, 6, 8)) %>%
  ggplot(aes(
    x = factor(cyl),
    y = hwy,
    fill = factor(cyl)
  )) +
  # add half violin from `ggdist` package
  ggdist::stat_halfeye(
    # custom bandwidth
    adjust = 0.5,
    # move geom to right
    justification = -0.2,
    # remove slab interval
    .width = 0,
    point_color = NA
  ) +
  # add boxplot
```

```
geom_boxplot(width = 0.12,
  # remove outliers
  outlier.colour = NA,
  alpha = 0.5) +
# add dot plots from `ggdist` package
ggdist::stat_dots( # orientation of the plot
  side = "left",
  # move geom to the left
  justification = 1.1,
  # adjust grouping of observation
  binwidth = 0.25) +
# adjust theme
scale_fill_tq() +
theme_tq() +
labs(
  title = "raincloud plot",
  subtitle = "showing bimodel distribution of 6 cylinder  vehicles",
  x = "highway fuel efficiency",
  y = "cylinders"
) +
coord_flip()
```



7.7 hex plot

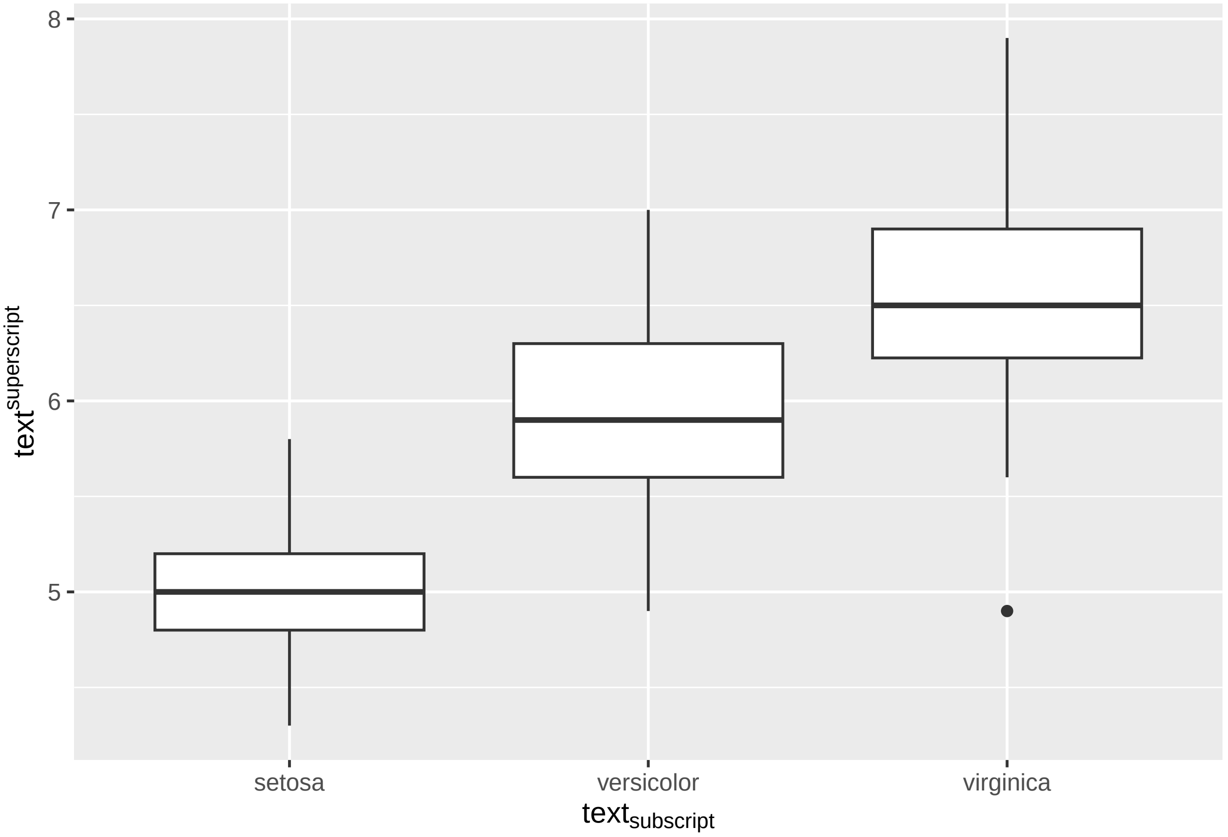
```r
library(tidyverse)
# install.packages("hexbin")
class <- c(rep("10th", 8))
students <- c("10 to 15",
  "15-20",
  "17 to 24",
  "20  to 25",
  "25 to 30",
  "30 to 40",
  "45 to 47",
  "50 to 55")
latitude <- c(
  11.50897246,
  11.48323136,
  11.48719031,
  11.46366611,
  11.41097322,
  11.52111154,
  11.44491386,
  11.46569568
)
longitude <- c(
  76.06032062,
  76.06192685,
  76.04266851,
  76.04156575,
  76.05075092,
  76.02846331,
  76.03084141,
  76.01766216
)
school <- data.frame(class, students, latitude, longitude)

school %>% mutate(students = parse_number(students)) %>%
  ggplot(aes(latitude, longitude, z = students)) +
  stat_summary_hex() +
  scale_fill_viridis_c(alpha = 0.8) +
  labs(fill = "students", title = "school students")
```

```
## Warning: Computation failed in ‘stat_summary_hex()‘
## Caused by error in ‘compute_group()‘:
## ! The package "hexbin" is required for ‘stat_summary_hex()‘
```

school students
longitude
latitude

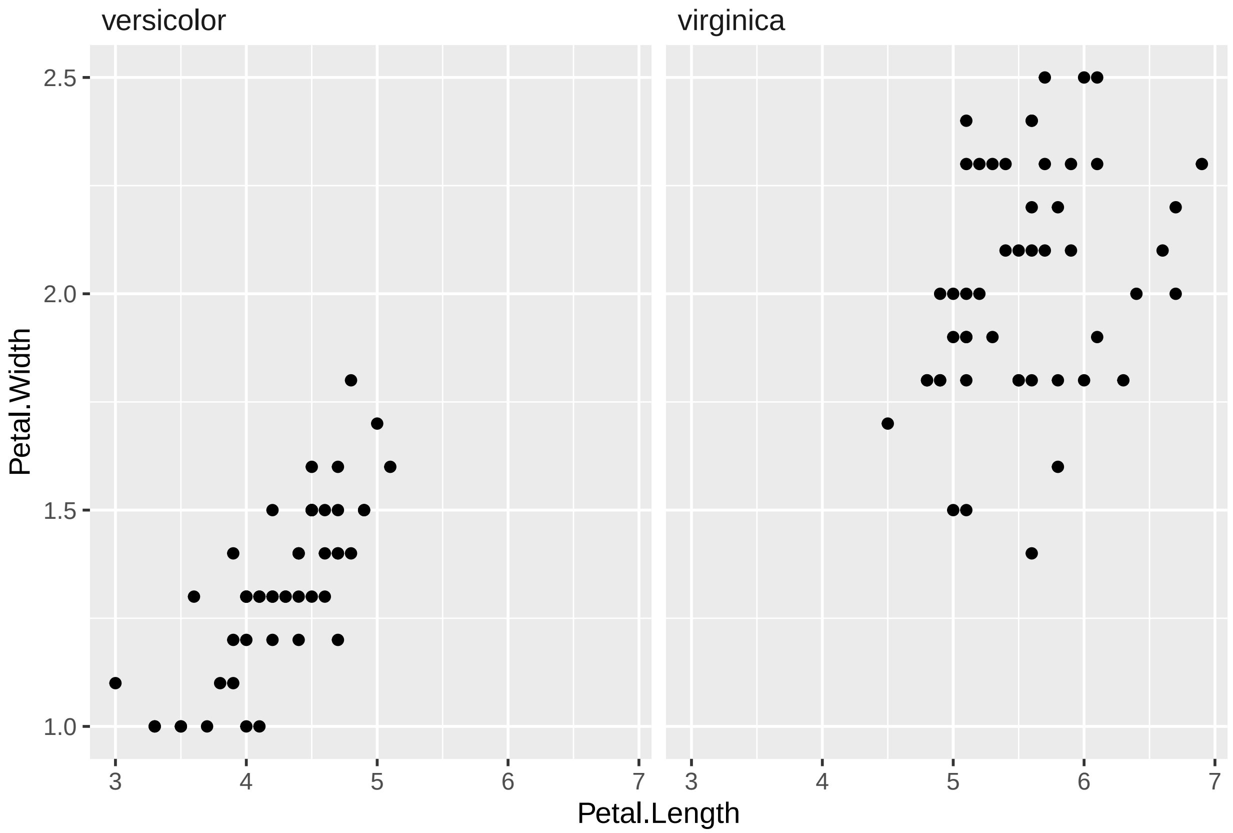
7.8 Subscript and superscript

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) + geom_boxplot() +
    labs(x = expression(text[subscript]), y = expression(text^superscript))
```



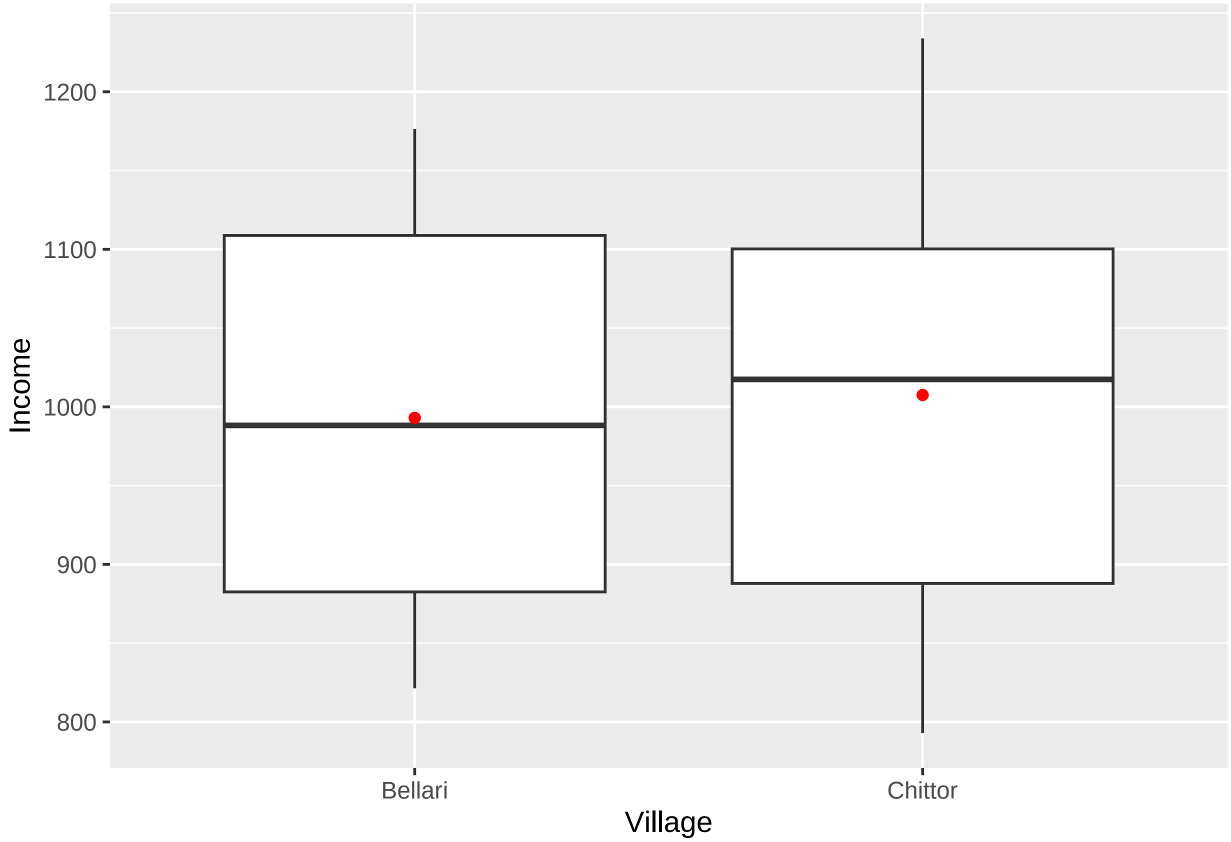
7.9 Two subtitles in two different positions in ggplot2

```
library(ggplot2)
library(dplyr, warn = FALSE)
iris %>%
    filter(Species != "setosa") %>%
    ggplot(aes(x = Petal.Length, y = Petal.Width)) + geom_point() +
    facet_wrap(~Species) + theme(strip.background.x = element_blank(),
    strip.text.x = element_text(hjust = 0, size = 11))
```



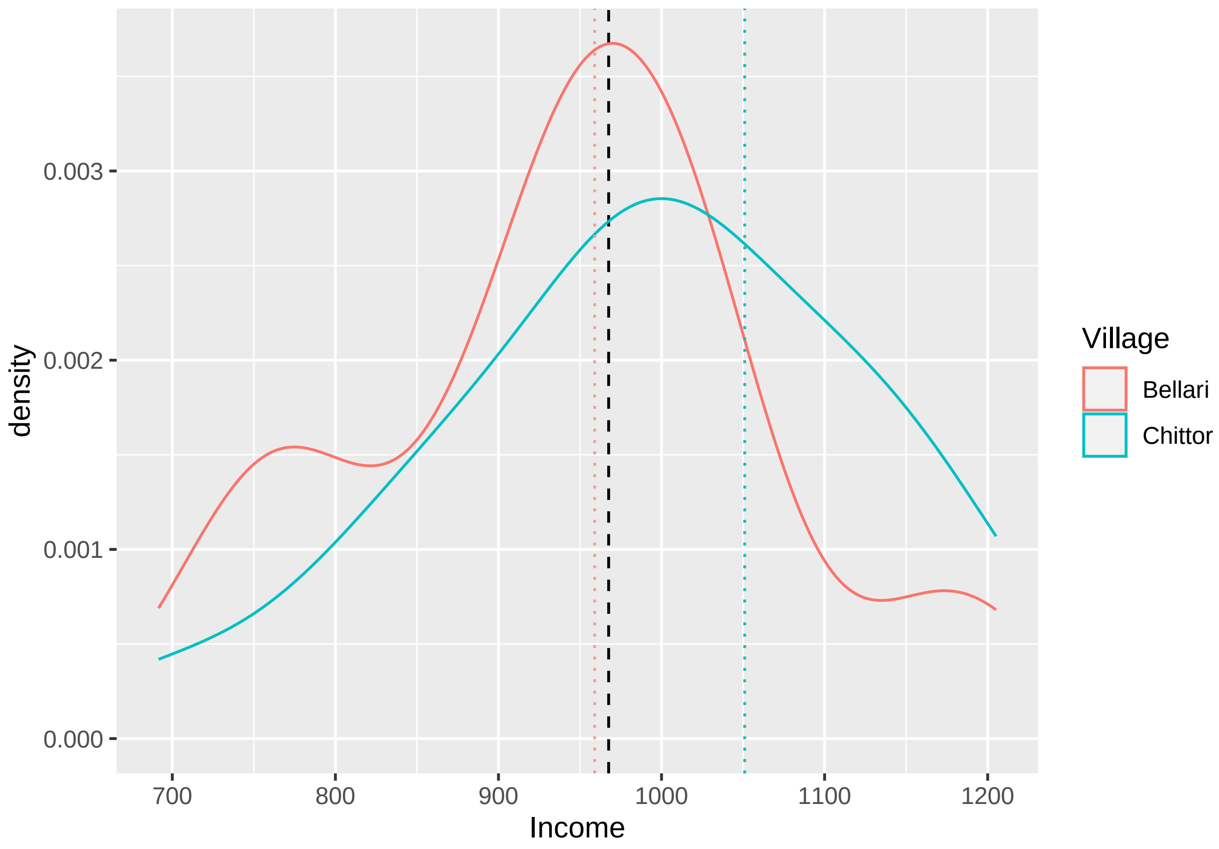
7.10 stat summary

```
income.data <- data.frame(Village = c(rep("Chittor", 20), rep("Bellari",
    20)), Income = c(rnorm(n = 20, mean = 1000, sd = 150), rnorm(n = 20,
    mean = 1000, sd = 150)))
library(ggplot2)
ggplot(income.data, aes(Village, Income)) + geom_boxplot() +
    stat_summary(geom = "point", fun = mean, col = "red")
```



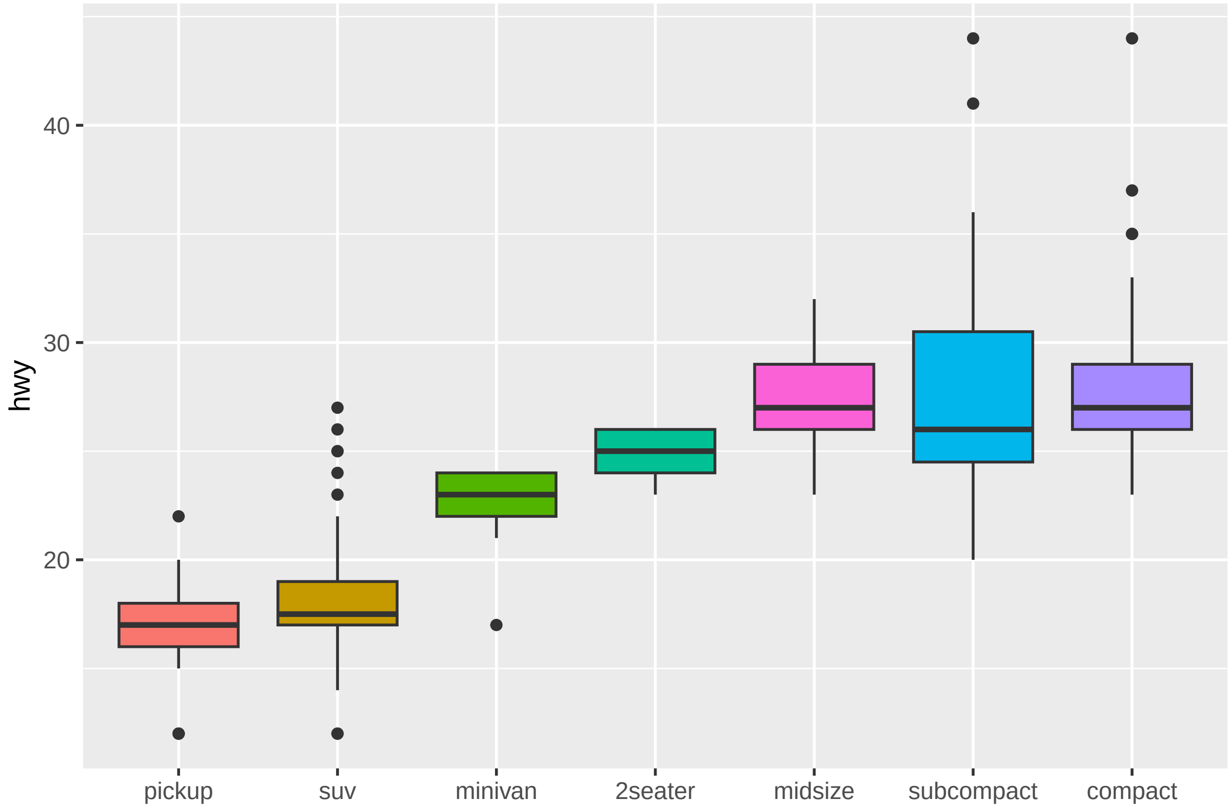
7.11 geom_density

```
income.data <- data.frame(Village = c(rep("Chittor", 20), rep("Bellari",
    20)), Income = c(rnorm(n = 20, mean = 1000, sd = 150), rnorm(n = 20,
    mean = 1000, sd = 150)))
library(ggplot2)
ggplot(income.data) + geom_vline(aes(xintercept = mean(Income)),
    linetype = "dashed") + geom_density(aes(x = Income, color = Village)) +
    geom_vline(xintercept = 959, linetype = "dotted", col = "#f39c96") +
    geom_vline(xintercept = 1051, linetype = "dotted", col = "#00bfc4")
```



7.12 reorder axis

```
library(tidyverse)
# Using median
mpg %>%
    mutate(class = fct_reorder(class, hwy, .fun = "median")) %>%
    ggplot(aes(x = reorder(class, hwy), y = hwy, fill = class)) +
    geom_boxplot() + xlab("class") + theme(legend.position = "none") +
    xlab("")
```



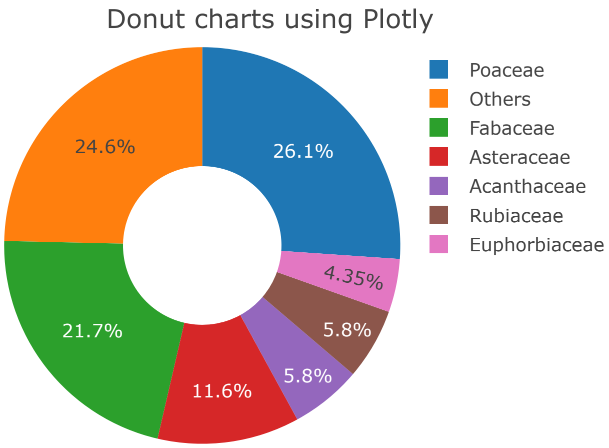
7.13 pie chart

```
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
data <- data.frame(category = c("Poaceae", "Fabaceae", "Asteraceae",
    "Acanthaceae", "Rubiaceae", "Euphorbiaceae", "Others"), count = c(18,
    15, 8, 4, 4, 3, 17))
```

```r
fig <- data %>%
    plot_ly(labels = ~category, values = ~count)
fig <- fig %>%
    add_pie(hole = 0.4) %>%
    layout(title = "Donut charts using Plotly", showlegend = T)

fig
```

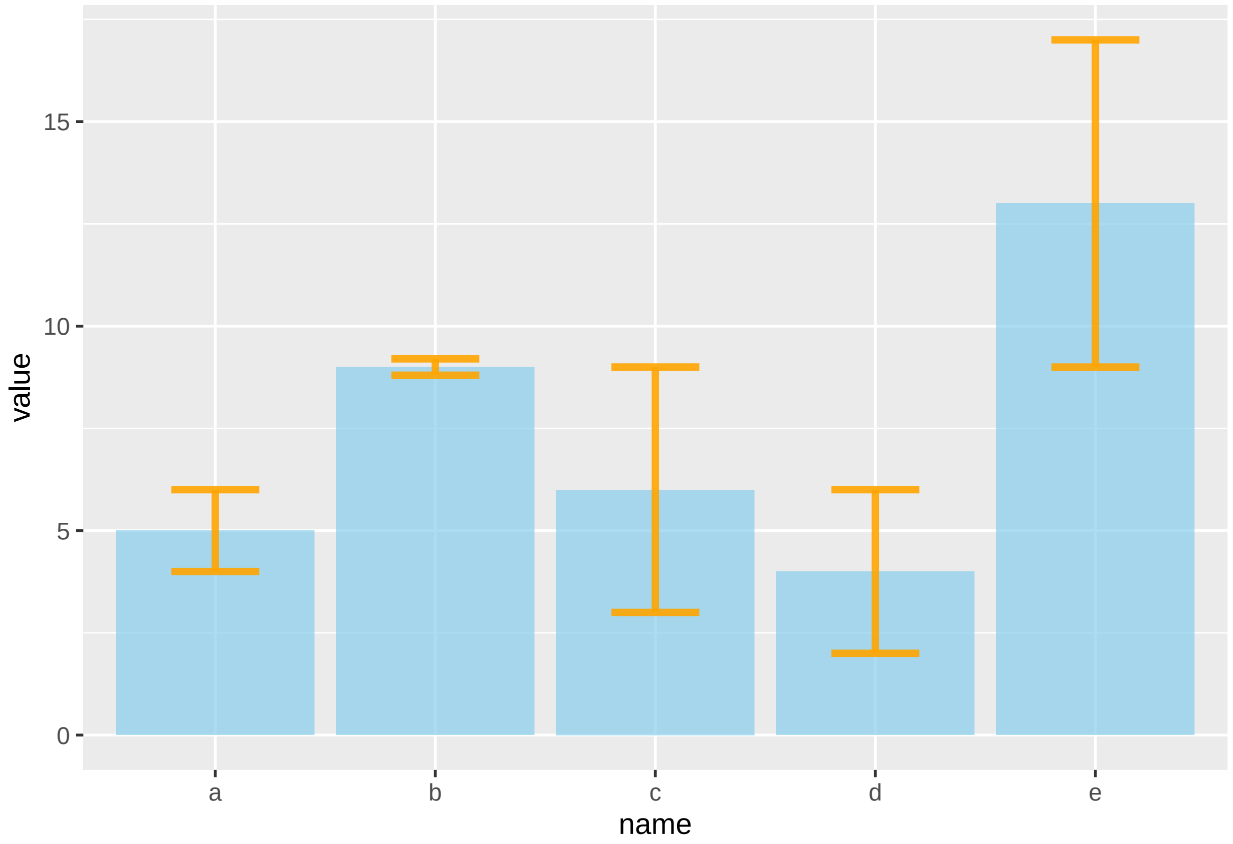

Donut charts using Plotly
Poaceae
Others
Fabaceae
Asteraceae
Acanthaceae
Rubiaceae
Euphorbiaceae
26.1%
24.6%
21.7%
11.6%
5.8%
5.8%
4.35%

7.14 barplot with error bar

```
# create dummy data
data <- data.frame(name = letters[1:5], value = sample(seq(4,
    15), 5), sd = c(1, 0.2, 3, 2, 4))

# Most basic error bar
library(viridis)
```

```
## Loading required package: viridisLite
```

```
ggplot(data) + geom_bar(aes(x = name, y = value), stat = "identity",
    fill = "skyblue", alpha = 0.7) + scale_fill_viridis_d() +
    geom_errorbar(aes(x = name, ymin = value - sd, ymax = value +
        sd), width = 0.4, colour = "orange", alpha = 0.9, linewidth = 1.3)
```

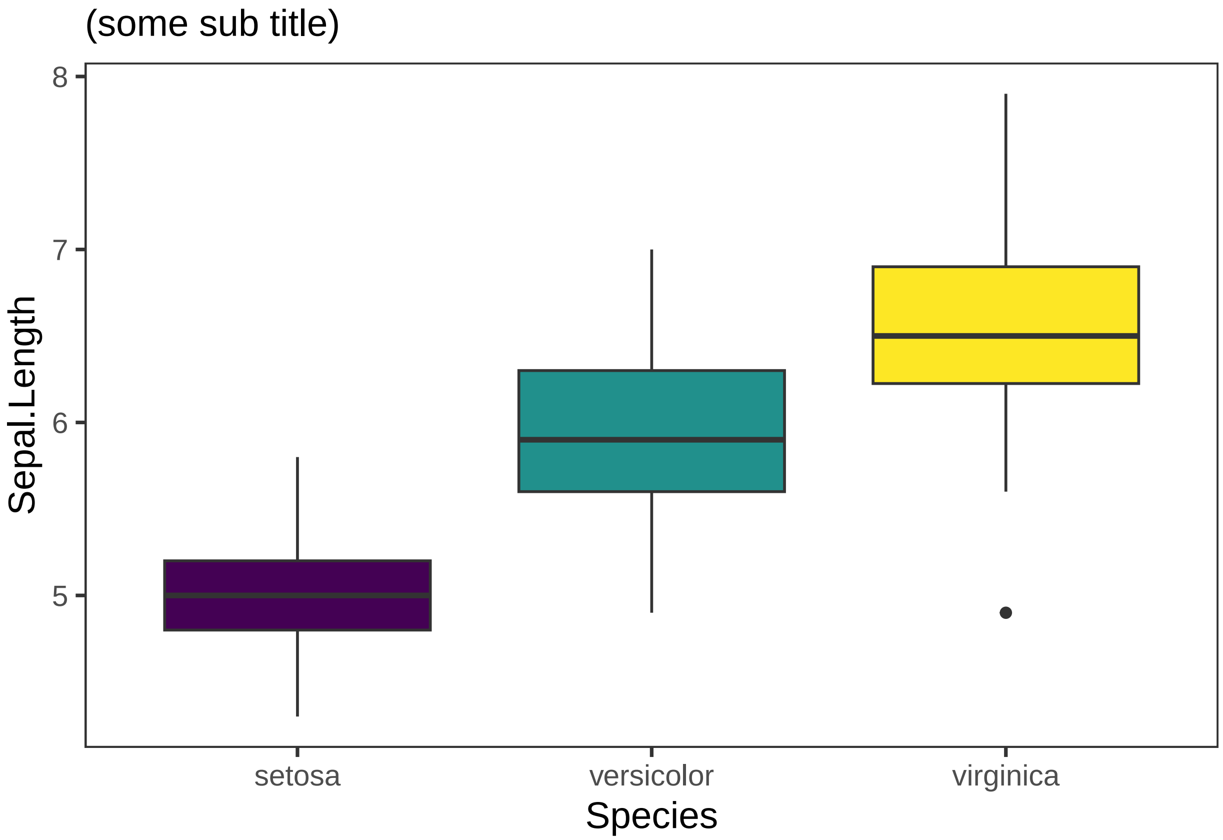


7.15 Expressions on labs

```
test <- iris %>%
    select(Species, Sepal.Length) %>%
    ggplot() + geom_boxplot(aes(Species, Sepal.Length, fill = Species)) +
    labs(y = expression(paste("Sepal ", length[(`in cm`)])))
```

7.16 Themes

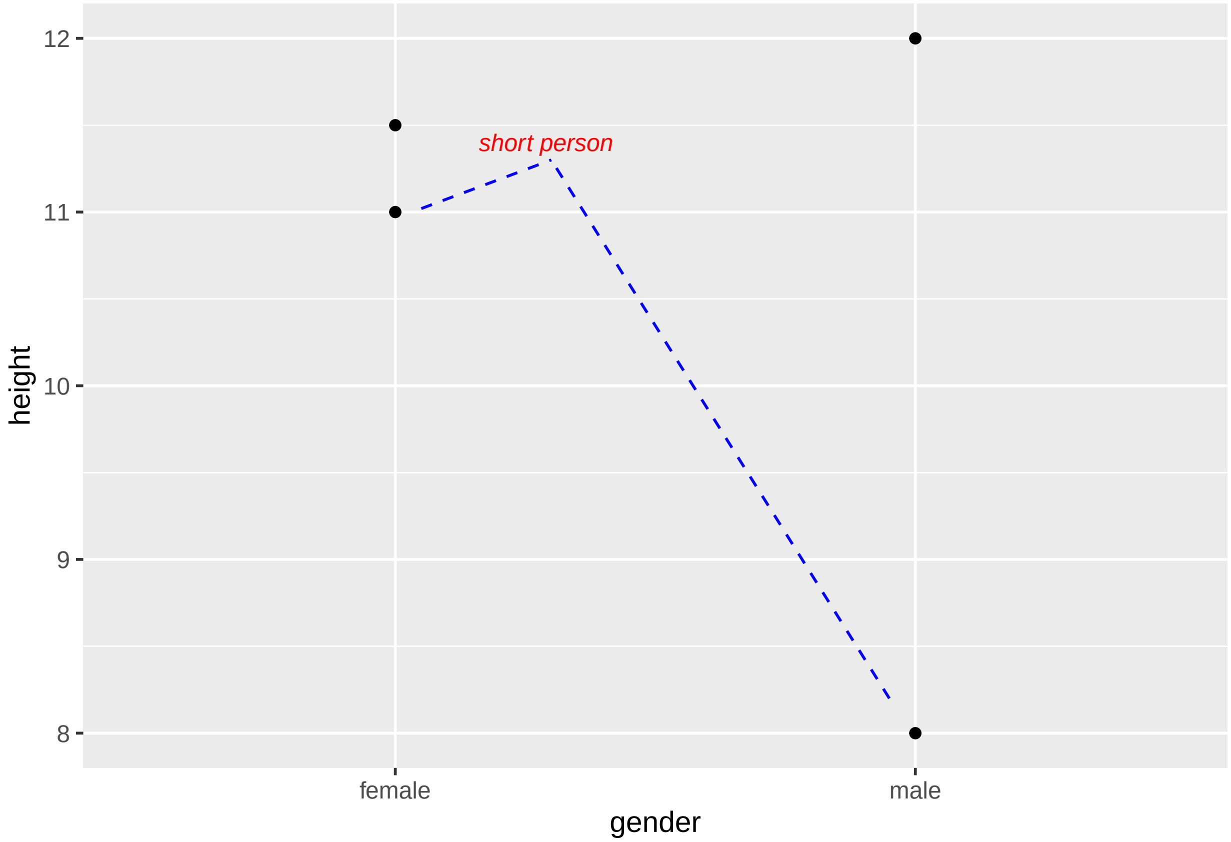
```
library(viridis)
iris %>%
    select(Species, Sepal.Length) %>%
    ggplot() + geom_boxplot(aes(Species, Sepal.Length, fill = Species)) +
    scale_color_viridis(discrete = T, option = "D") + scale_fill_viridis(discrete = T,
    option = "D") + theme_bw(base_size = 14) + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), strip.background = element_rect(colour = "black",
        fill = "white"), legend.position = " ") + labs(subtitle = "(some sub title)") +
    guides(fill = "none")
```



```
library(ggThemeAssist)
test <- iris %>%
    select(Species, Sepal.Length) %>%
    ggplot() + geom_boxplot(aes(Species, Sepal.Length, fill = Species)) +
    labs(y = expression(paste("Sepal ", length[(`in cm`)])))

# run the ggThemeAssistGadget(test)
```

7.17 annotate

```r
library(tidyverse)
df <- tribble(~gender,
  ~height,
  "male",
  12,
  "male",
  8,
  "female",
  11.5,
  "female",
  11)

ggplot(df, aes(gender, height)) +
  geom_point() +
  annotate(
    geom = "text",
    x = 1.29,
    y = 11.4,
    label = "short person",
    color = "red",
    size = 3,
    fontface = "italic"
  ) +
  annotate(
    geom = "segment",
    x = 1.05,
    # starting point on x, this decides length
    xend = 1.3,
    # end point on x, this decides length
    y = 11.02,
    # starting point on y
    yend = 11.3,
    # ending point on y
    color = "blue",
    linetype = "dashed"
  ) +
  annotate(
    geom = "segment",
    x = 1.95,
    # starting point on x, this decides length
    xend = 1.3,
    # end point on x, this decides length
    y = 8.2,
    # starting point on y
    yend = 11.3,
    # ending point on y
    color = "blue",
    linetype = "dashed"
  )
```

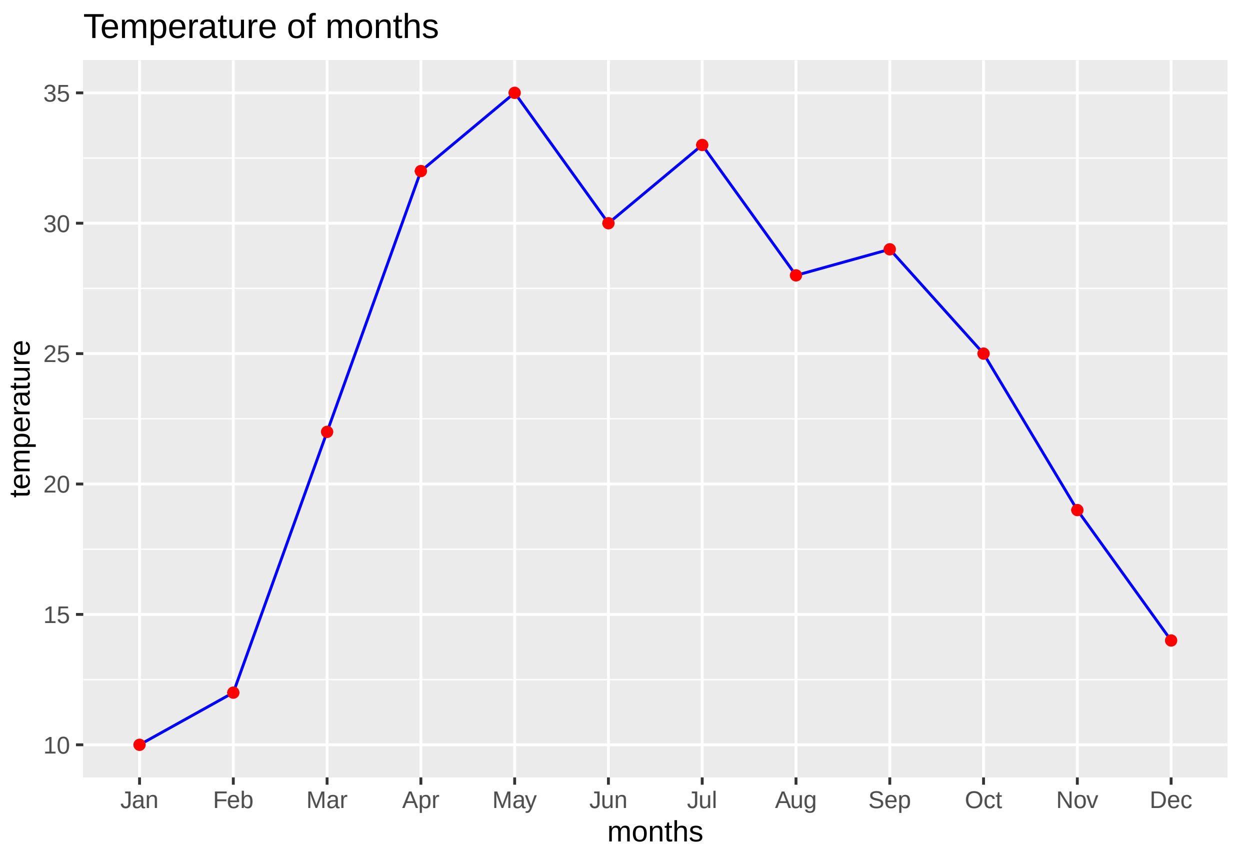


7.18 months

```
library(lubridate)
months <- seq(month(1:12))  # make moths
months <- month.abb[months]  # make abbriviations
temperature <- c(10, 12, 22, 32, 35, 30, 33, 28, 29, 25, 19,
    14)
myframe <- data.frame(months, temperature)  # creating a new data frame

library(tidyverse)
glimpse(myframe)
```

```
## Rows: 12
## Columns: 2
## $ months      <chr> "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "S~
## $ temperature <dbl> 10, 12, 22, 32, 35, 30, 33, 28, 29, 25, 19, 14
```

```
library(ggplot2)
ggplot(myframe, aes(x = months, y = temperature, group = 1)) +
    geom_line(col = "blue") + geom_point(col = "red") + ggtitle("Temperature of months") +
    scale_x_discrete(limits = month.abb)  # this will order months on the x axis
```

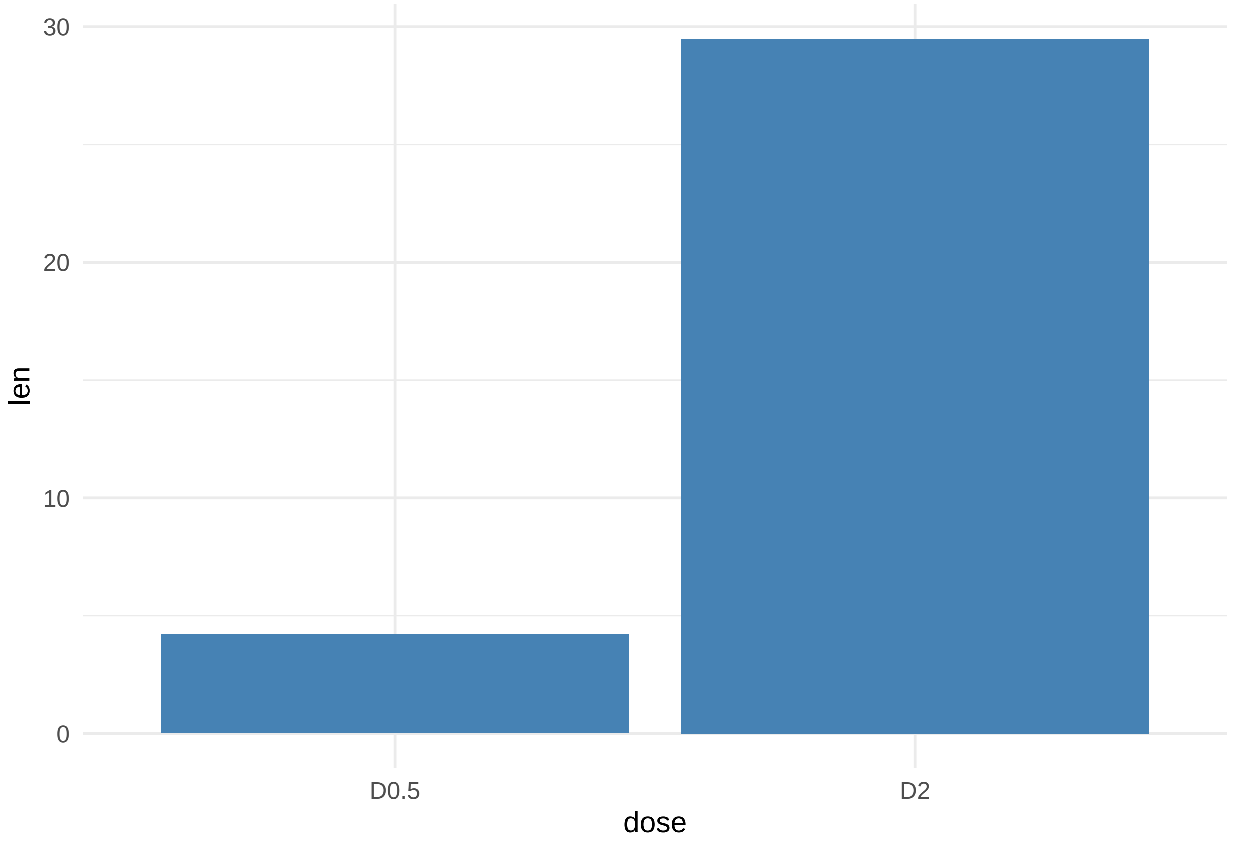


```r
# create and view data frame
df <- data.frame(date = c("05/30/2021", "08/18/2021", "09/13/2021",
    "02/19/2021"), sales = c(3, 15, 14, 9))

df <- df %>%
    mutate(date = as.Date(date, format = "%m/%d/%Y")) %>%
    arrange(date)
df
```

```
##         date sales
## 1 2021-02-19     9
## 2 2021-05-30     3
## 3 2021-08-18    15
## 4 2021-09-13    14
```

```r
p + scale_x_discrete(limits = c("D0.5", "D2"))
```

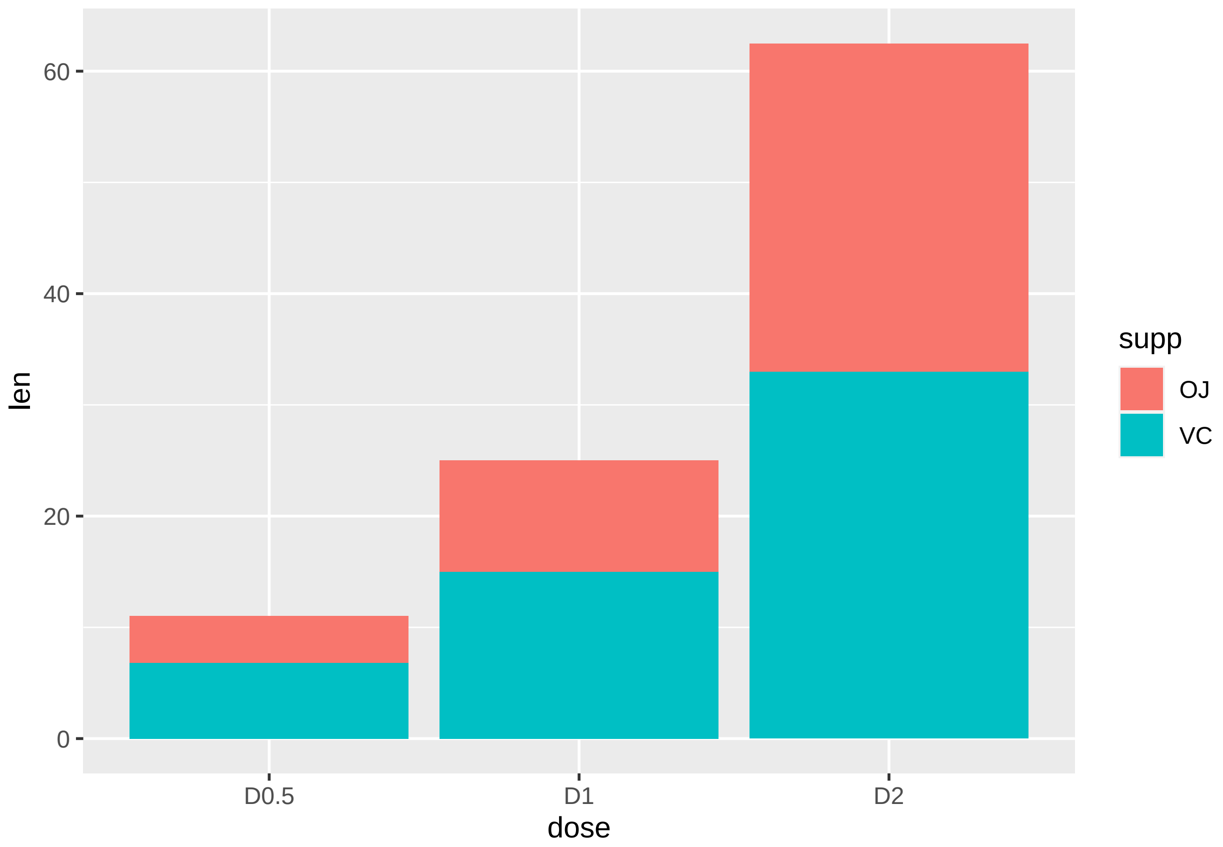
```
## Warning: Removed 1 rows containing missing values (‘position_stack()‘).
```



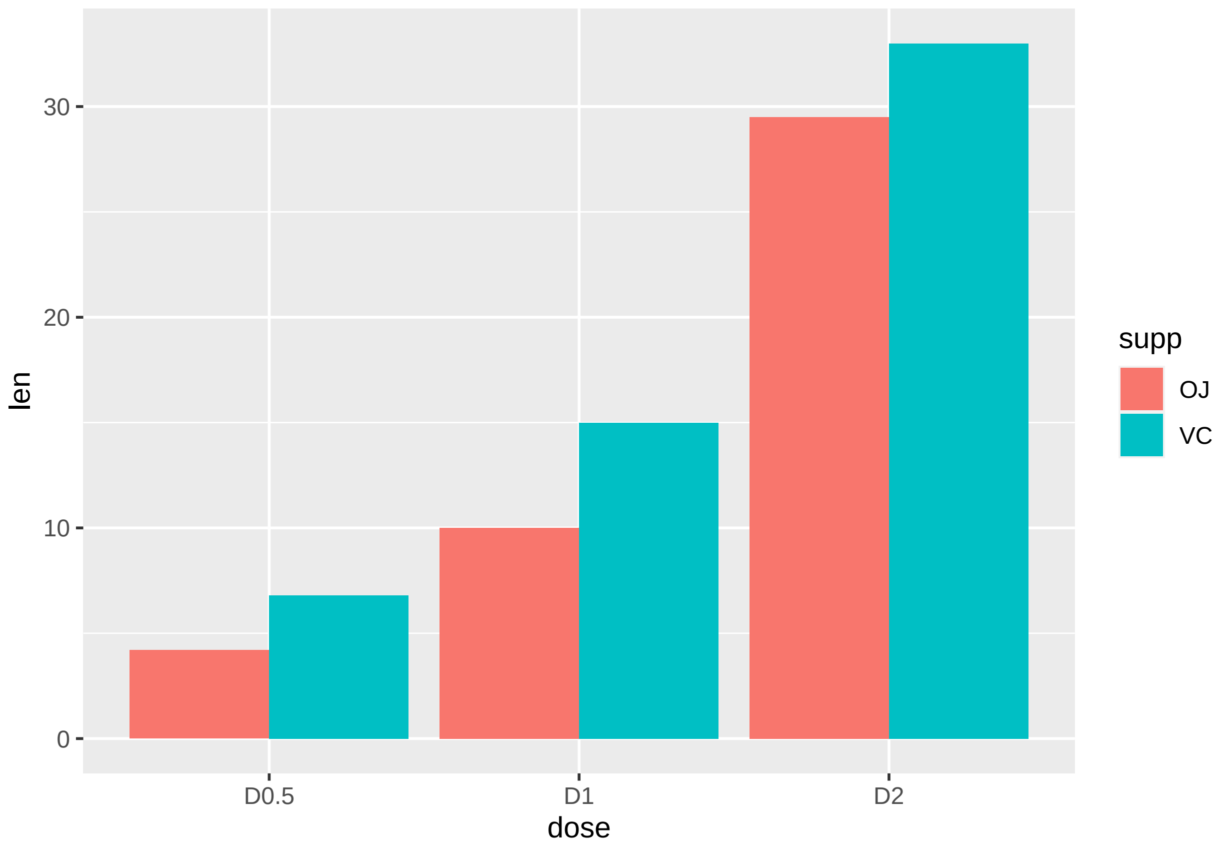
```
df2 <- data.frame(supp = rep(c("VC", "OJ"), each = 3), dose = rep(c("D0.5",
    "D1", "D2"), 2), len = c(6.8, 15, 33, 4.2, 10, 29.5))
```

```
p <- ggplot(data = df2, aes(x = dose, y = len, fill = supp)) +
    geom_bar(stat = "identity", position = position_dodge()) +
    geom_text(aes(label = len), vjust = 1.6, color = "white",
        position = position_dodge(0.9), size = 3.5) + scale_fill_brewer(palette = "Paired") +
    theme_minimal()
```

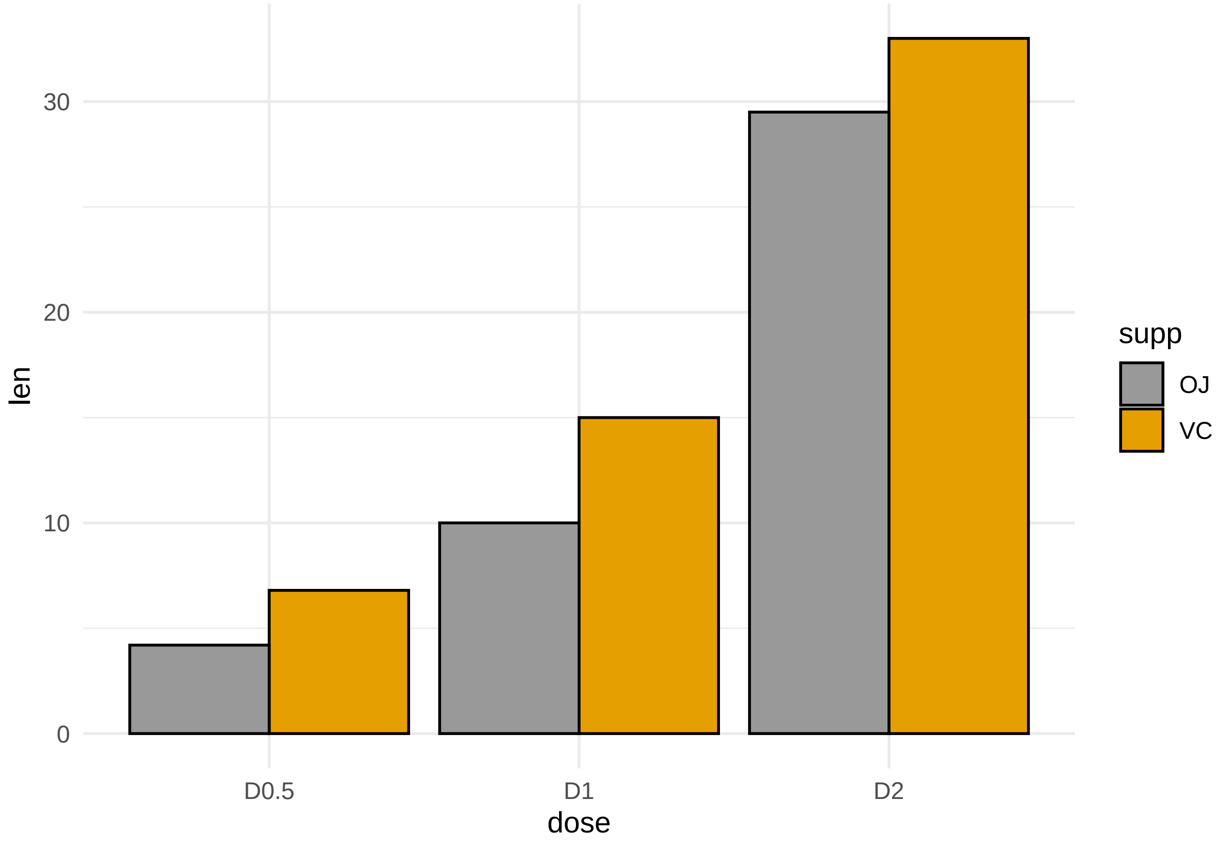
```
# Stacked barplot with multiple groups
ggplot(data = df2, aes(x = dose, y = len, fill = supp)) + geom_bar(stat = "identity")
```



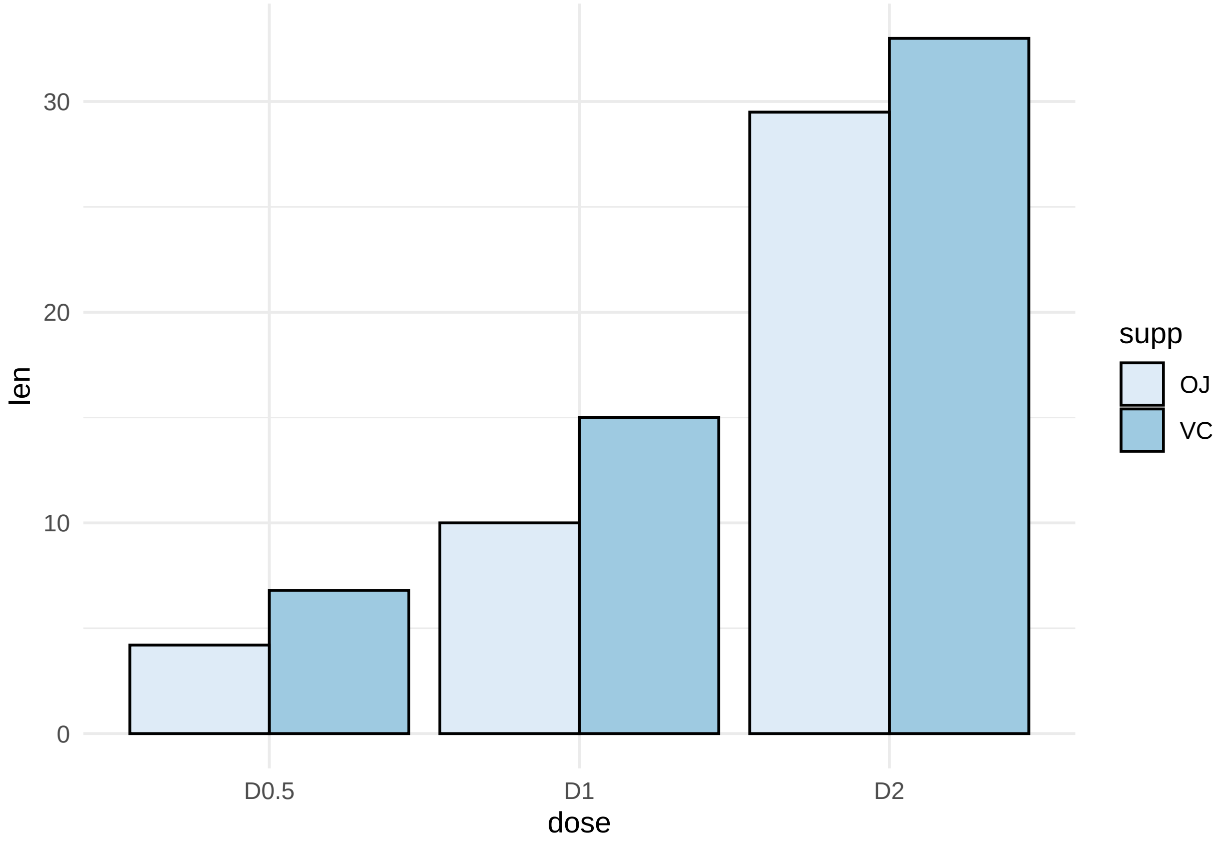
```
# Use position=position_dodge()
ggplot(data = df2, aes(x = dose, y = len, fill = supp)) + geom_bar(stat = "identity",
    position = position_dodge())
```



```r
# Change the colors manually
p <- ggplot(data = df2, aes(x = dose, y = len, fill = supp)) +
    geom_bar(stat = "identity", color = "black", position = position_dodge()) +
    theme_minimal()
# Use custom colors
p + scale_fill_manual(values = c("#999999", "#E69F00"))
```



```
# Use brewer color palettes
p + scale_fill_brewer(palette = "Blues")
```



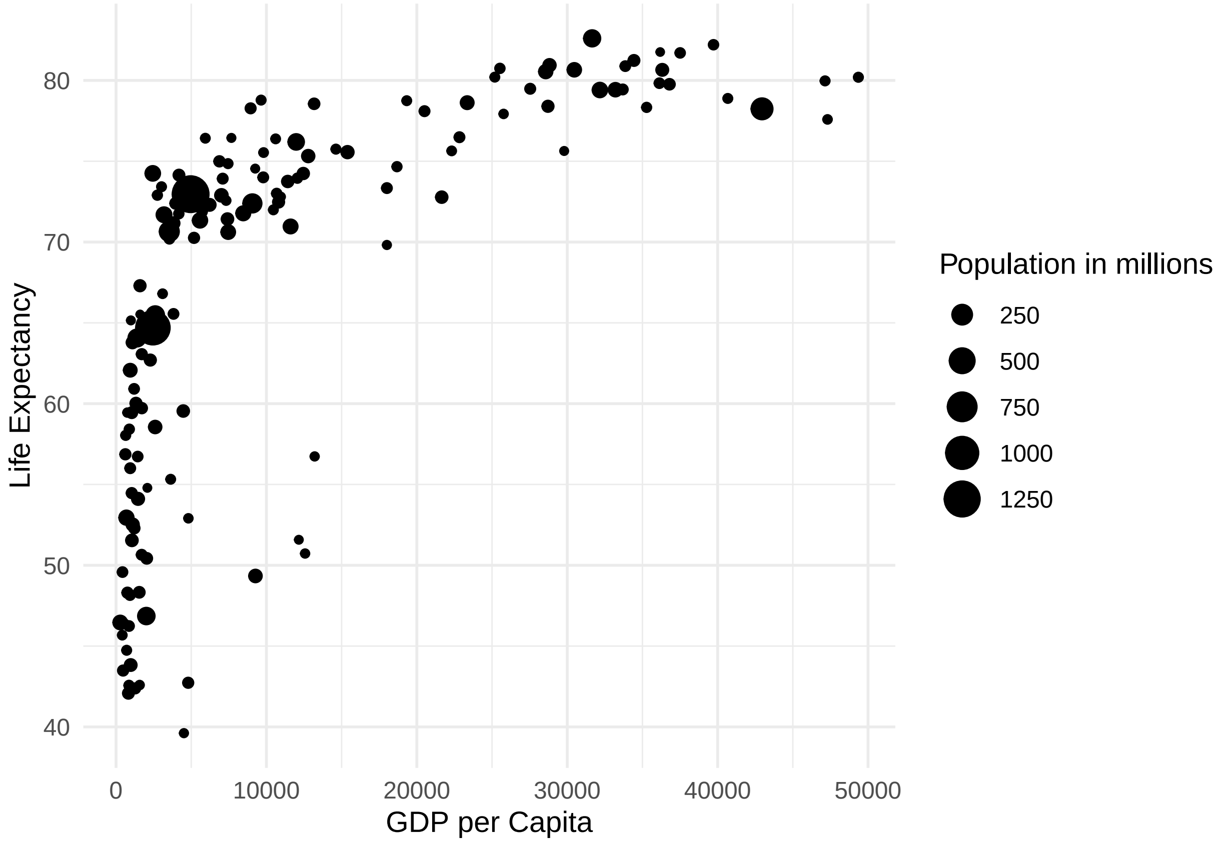
7.19 Color Palettes

libraries

```r
# install.packages('MetBrewer')
library(MetBrewer)
```

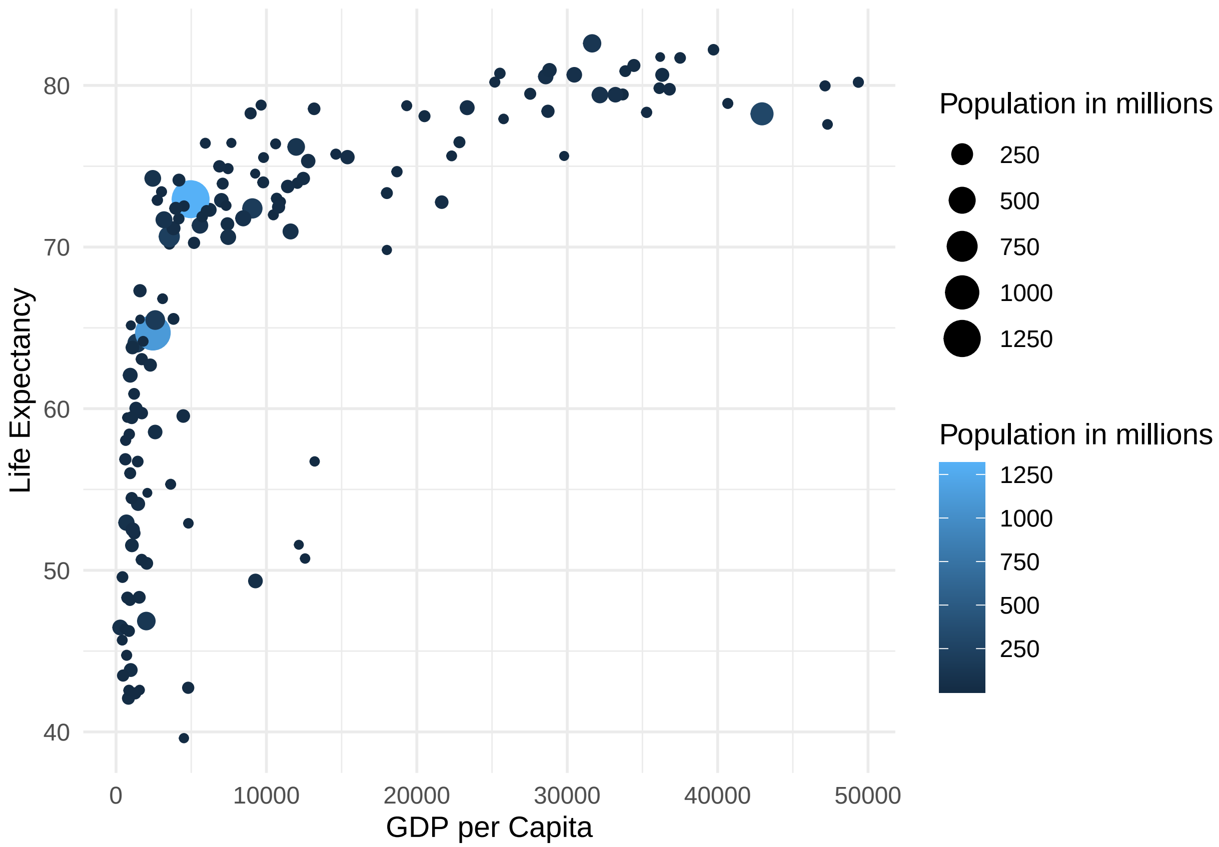
Plot the point plot using GDP per Capita as the x- axis and LE as the y axis. Numerical variable Population to control the size of each point.

```r
plot <- gapminder %>%
    filter(year == 2007) %>%
    ggplot() + labs(x = "GDP per Capita", y = "Life Expectancy",
    color = "Population in millions", size = "Population in millions") +
    theme_minimal()

plot + geom_point(aes(gdpPercap, lifeExp, size = pop/1e+06))
```



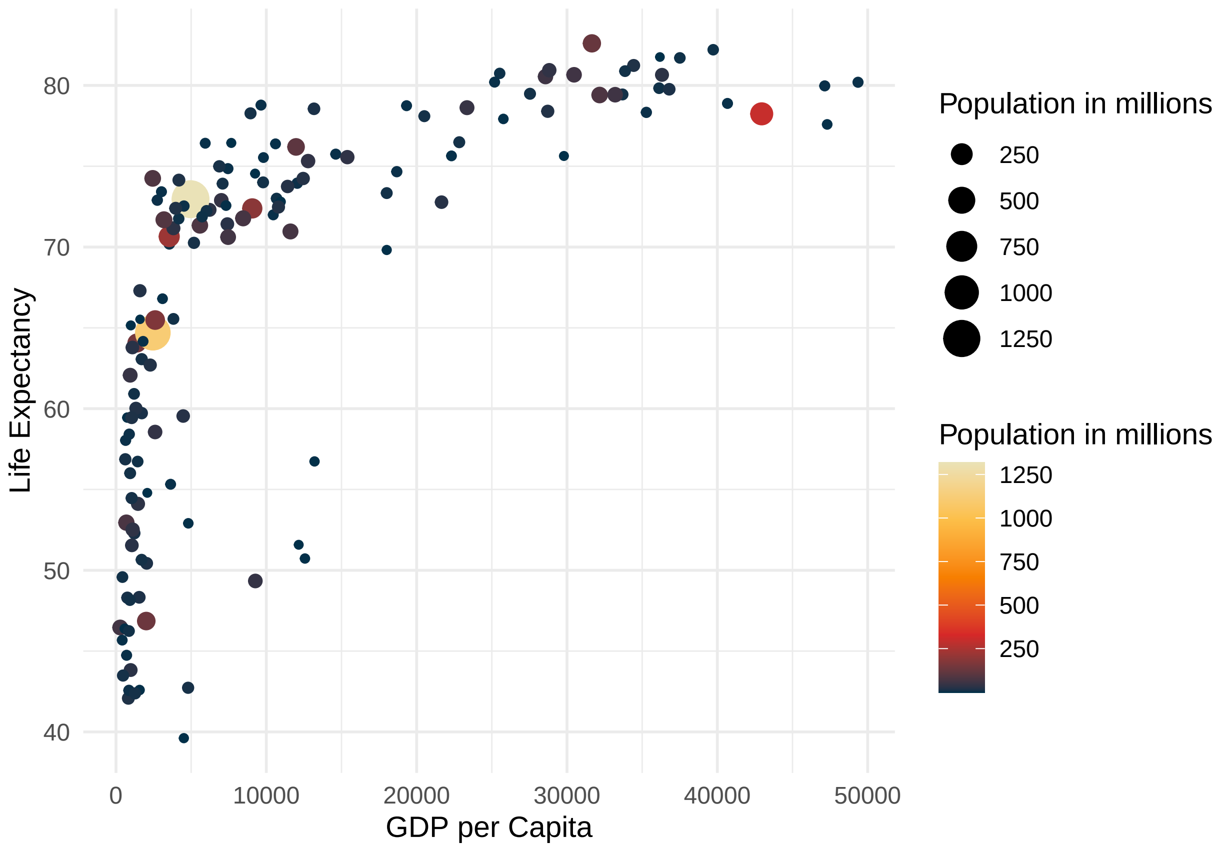
To use color in the plot, assign the Population variable to the color aesthetic. Since nothing is specied, ggplot2 chooses a color spectrum for this numerical variable (shades of blue).

```
plot + geom_point(aes(gdpPercap, lifeExp, size = pop/1e+06, color = pop/1e+06))
```



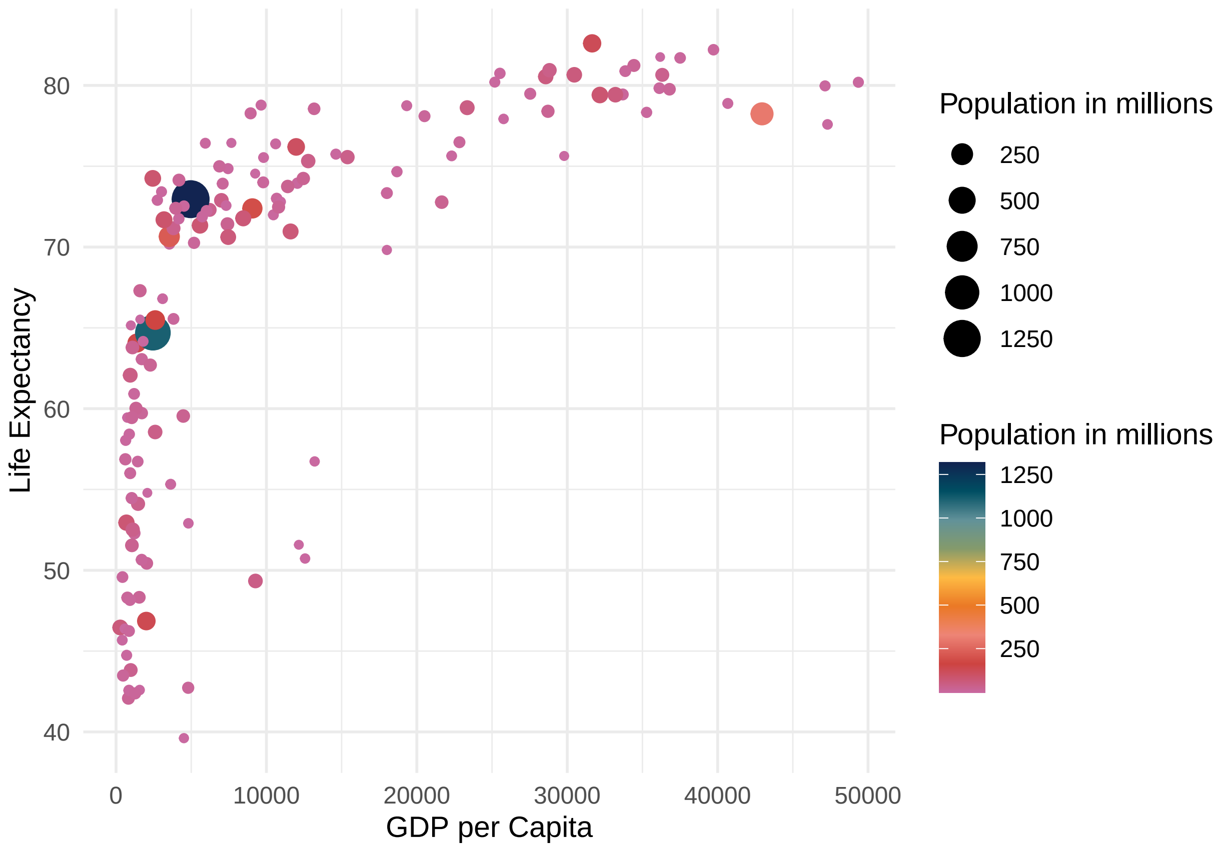
To control the color spectrum, we need to introduce a color scale. In the following plot, we have to provide a vector of hex color values. You would choose this if you got your colors from one of the mentioned above websites.

```
plot + geom_point(aes(gdpPercap, lifeExp, size = pop/1e+06, color = pop/1e+06)) +
    scale_color_gradientn(colors = c("#003049", "#D62828", "#F77F00",
        "#FCBF49", "#EAE2B7"))
```



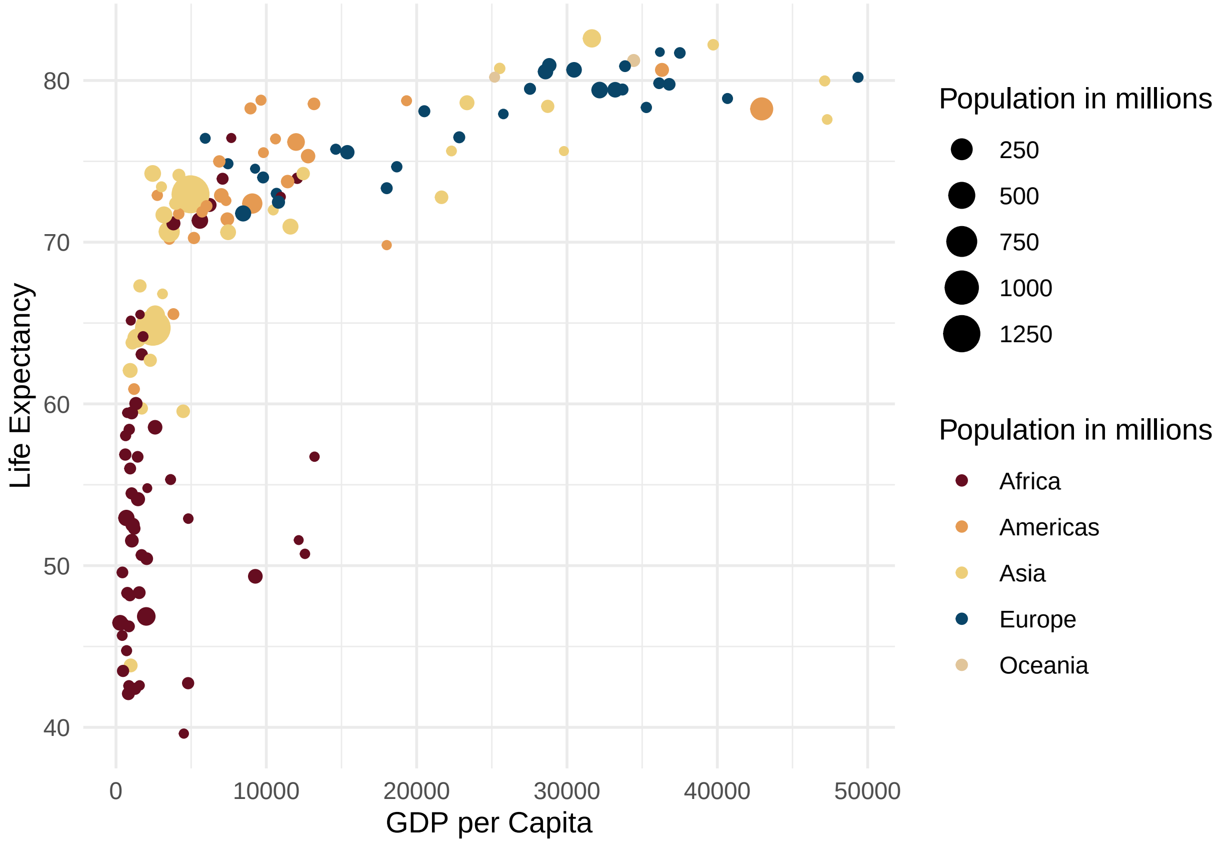
To apply one of the MetBrewer palettes, replace the hex-vector with a MetBrewer function. Within the function call, you provide the palette's name, then several colors, and tell it that we need a continuous palette since it is a numerical variable.

```
plot + geom_point(aes(gdpPercap, lifeExp, size = pop/1e+06, color = pop/1e+06)) +
    scale_color_gradientn(colors = met.brewer("Cross", n = 500,
        type = "continuous"))
```



You might also want to use color palettes with non-numerical variables. Let us assume we want to apply color to the Continent variable. This implies using a manual color scale and providing a MetBrewer palette.

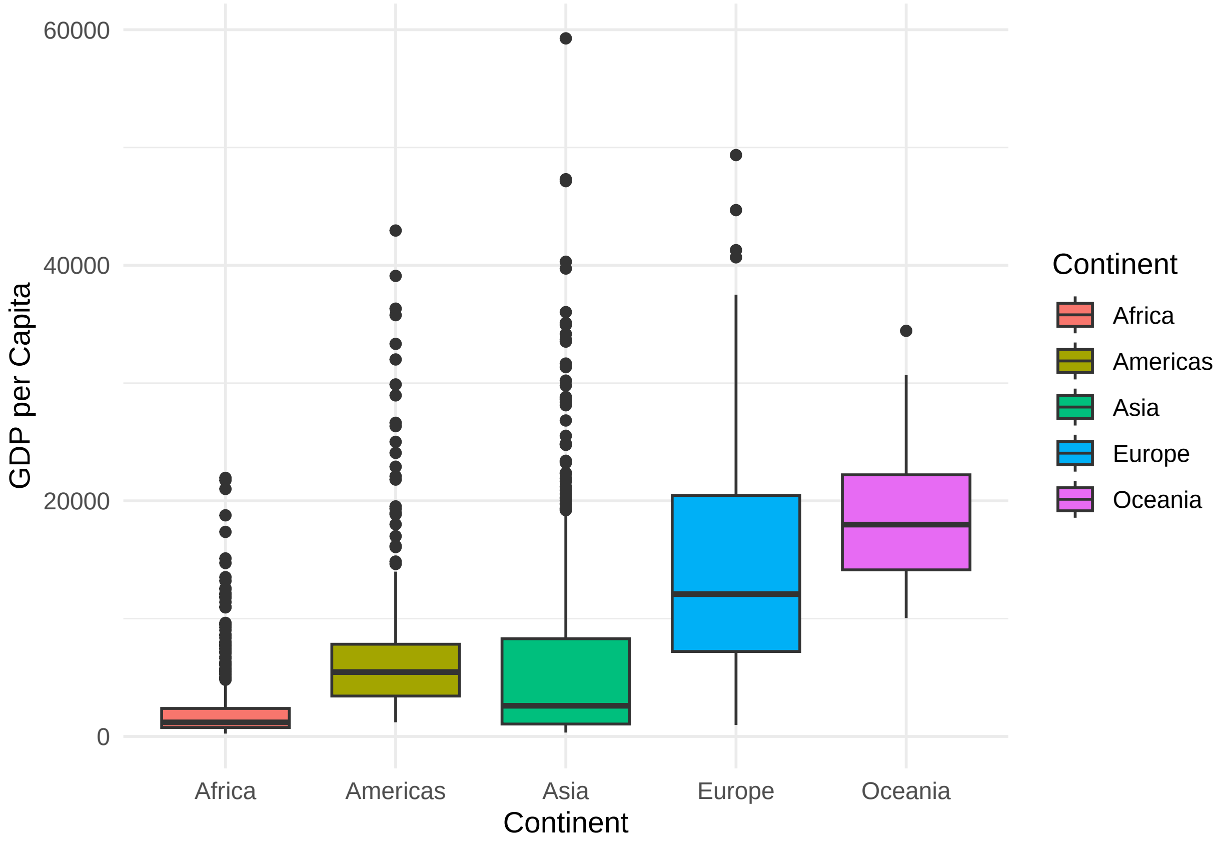
```
plot + geom_point(aes(gdpPercap, lifeExp, size = pop/1e+06, color = continent)) +
    scale_color_manual(values = met.brewer("Navajo", 5))
```



Please note if you want to apply color to the fill aesthetic rather than the color aesthetic, consider using the scale_fill_manuel function instead of the scale_color_manuel. This is useful for boxplots or bar charts.

```
gapminder %>%
    filter(gdpPercap < 60000) %>%
    ggplot(aes(continent, gdpPercap, color = year, fill = continent)) +
    geom_boxplot() + theme_minimal() + labs(x = "Continent",
    y = "GDP per Capita", fill = "Continent")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```

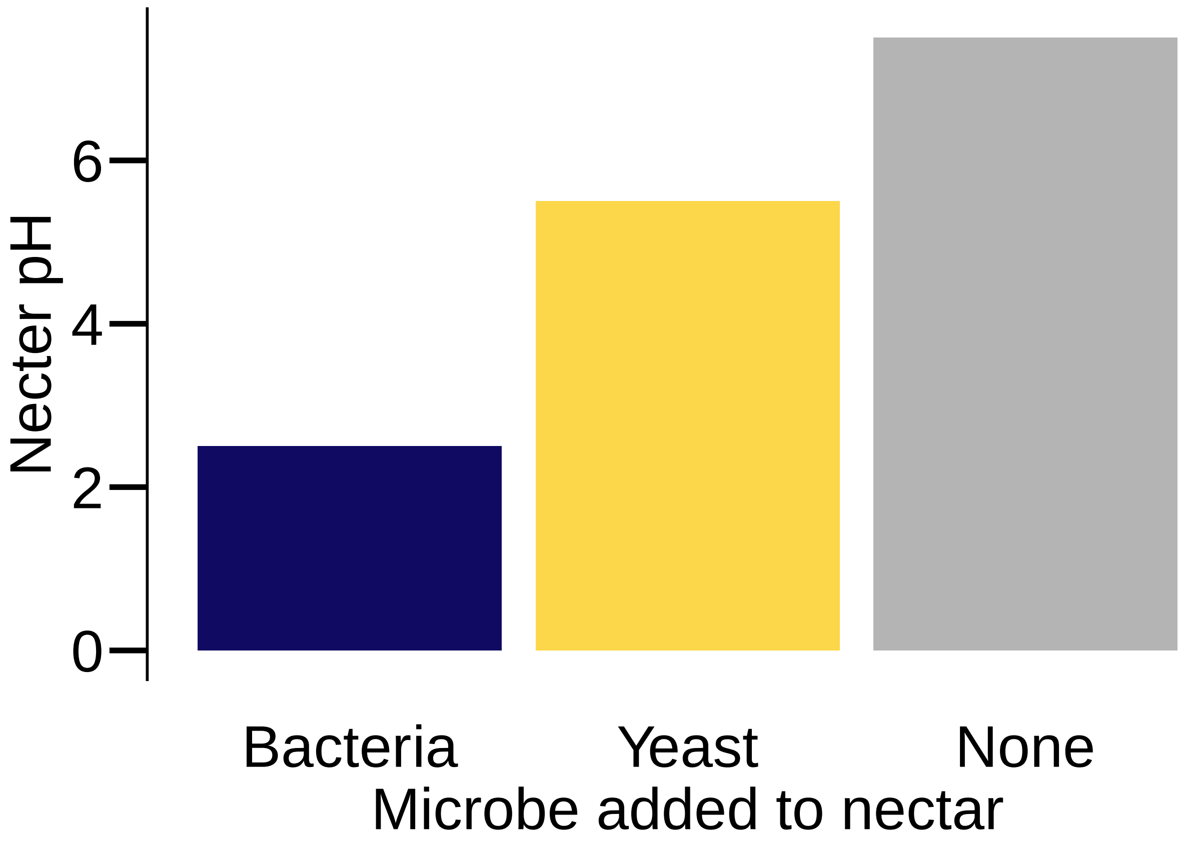


7.20 scale fill manual

7.21 themes

```
df <- data.frame(Names = as.factor(c("Bacteria", "Yeast", "None")),
    Quantity = c(2.5, 5.5, 7.5))

library(ggplot2)
library(tidyverse)
df <- df %>%
    mutate(Names = fct_relevel(Names, c("Bacteria", "Yeast",
        "None")))

ggplot(df, aes(Names, Quantity, fill = Names)) + geom_bar(stat = "identity") +
    scale_fill_manual(values = c("#110a62", "#fcd749", "#b5b4b5")) +
    labs(y = "Necter pH", x = "Microbe added to nectar") + theme_classic() +
    theme(legend.position = "none", axis.ticks.x = element_blank()) +
    theme(axis.text = element_text(size = 22, color = "black")) +
    theme(axis.line.x = element_blank()) + theme(axis.ticks = element_line(size = 1,
    color = "black"), axis.ticks.length = unit(0.5, "cm")) +
    theme(text = element_text(size = 22))
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



```
# ggThemeAssist::ggThemeAssistGadget(name of the plot)
```

7.21.1 graphics

```
x11()  # opne a new window for graphics
graphics.off()  # close the new window
```

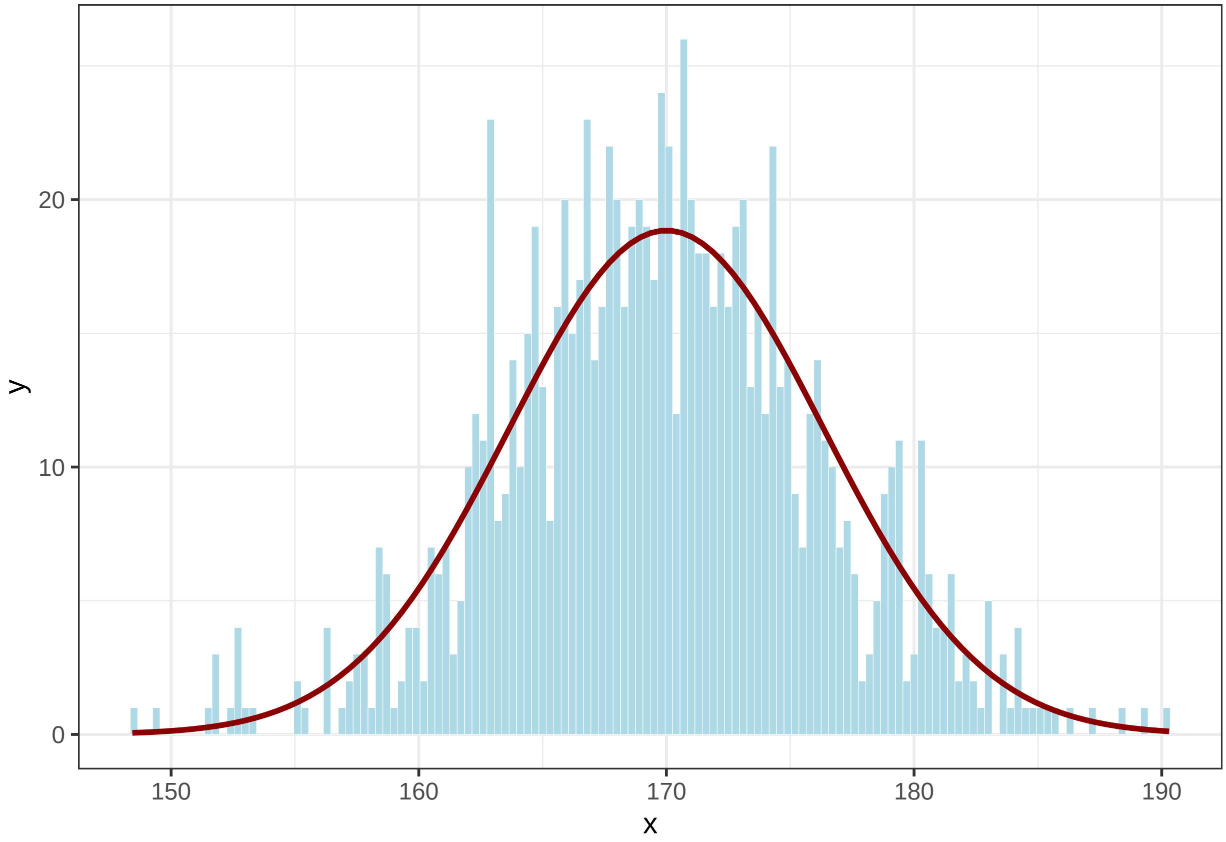
7.22 Normal distribution

Normal distribution, also known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean.

```
library(tidyverse)
n = 1000
mean = 170  # cm
sd = 6.35  # cm
```

```
binwidth = 0.3
set.seed(1234)
df <- data.frame(x = rnorm(n, mean, sd))
ggplot(df, aes(x = x, mean = mean, sd = sd, binwidth = binwidth,
    n = n)) + theme_bw() + geom_histogram(binwidth = binwidth,
    colour = "white", fill = "lightblue", size = 0.1) + stat_function(fun = function(x) dnorm(x,
    mean = mean, sd = sd) * n * binwidth, color = "darkred",
    linewidth = 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.
```



7.23 wordcloud

```
library(googlesheets4)
library(dplyr)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
##
## Attaching package: 'wordcloud'

## The following object is masked from 'package:PerformanceAnalytics':
##
##     textplot
```

```
library(RColorBrewer)

# gs4_auth()
path <- ("https://docs.google.com/spreadsheets/d/1ac8CuAQdRNXp9MjKsG7YWiHcT64tRgnCqlY9UhX-jEo/edit?usp=
test <- read_sheet(path)
```

```
## ! Using an auto-discovered, cached token.

##   To suppress this message, modify your code or options to clearly consent to
##   the use of a cached token.

##   See gargle's "Non-interactive auth" vignette for more details:

##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## i The googlesheets4 package is using a cached token for 'cssaneesh@gmail.com'.

## v Reading from "wordcloud".

## v Range 'Sheet1'.
```

```
head(test, 3)
```

```
## # A tibble: 3 x 2
##   courses                                topic
##   <chr>                                  <chr>
## 1 Critical Reasoning and Logic (Science) Philosphy
## 2 Data Science with R: Advanced          R
## 3 Data Science with R: Intermediate      R
```

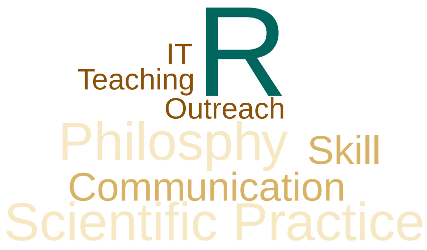
```
test1 <- data.frame(test %>%
    select(topic) %>%
    count(topic) %>%
    mutate(count = n * 10))
head(test1, 3)
```

```
##           topic n count
## 1 Communication 2    20
## 2            IT 1    10
## 3      Outreach 1    10
```

```
max(test1$count)
```

```
## [1] 90
```

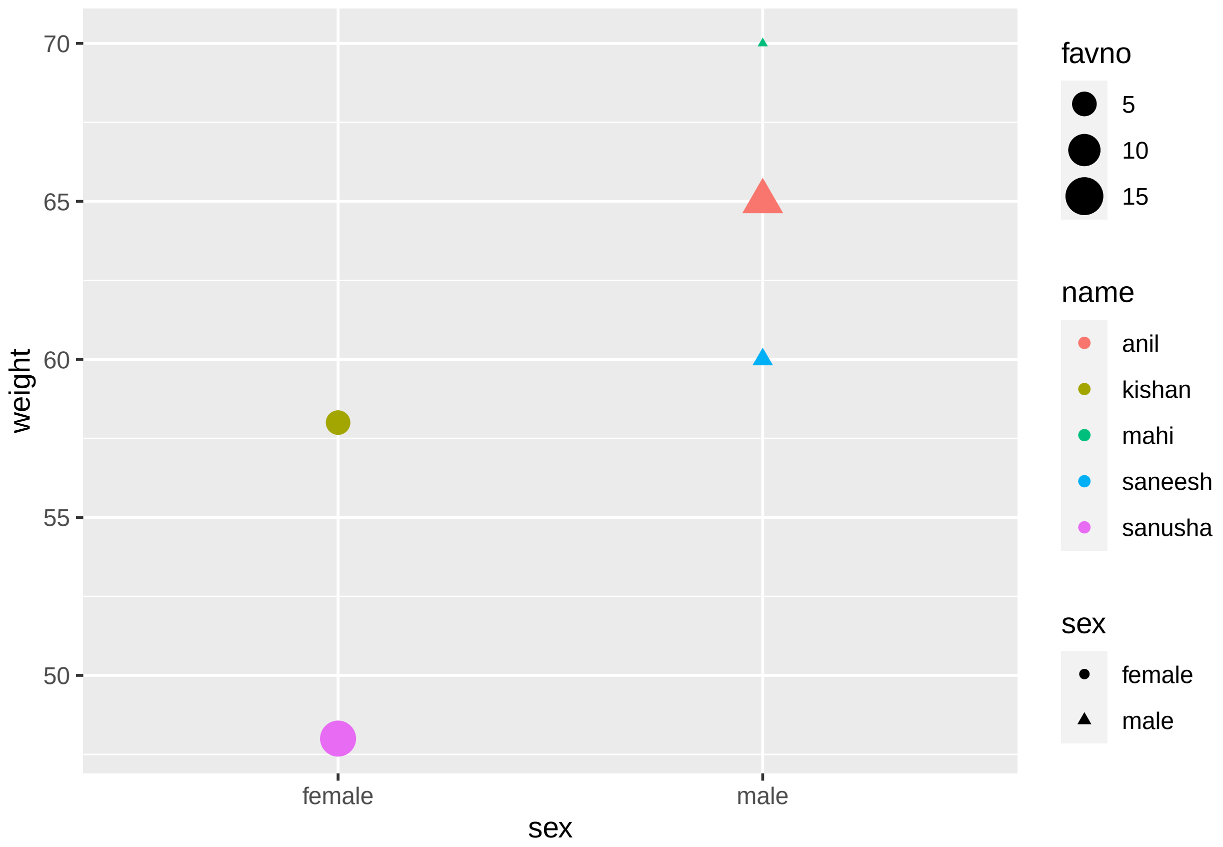
```
set.seed(123)

wordcloud(words = test1$topic, freq = test1$count, min.freq = 10,
    max.words = 50, colors = brewer.pal(7, "BrBG"))
```



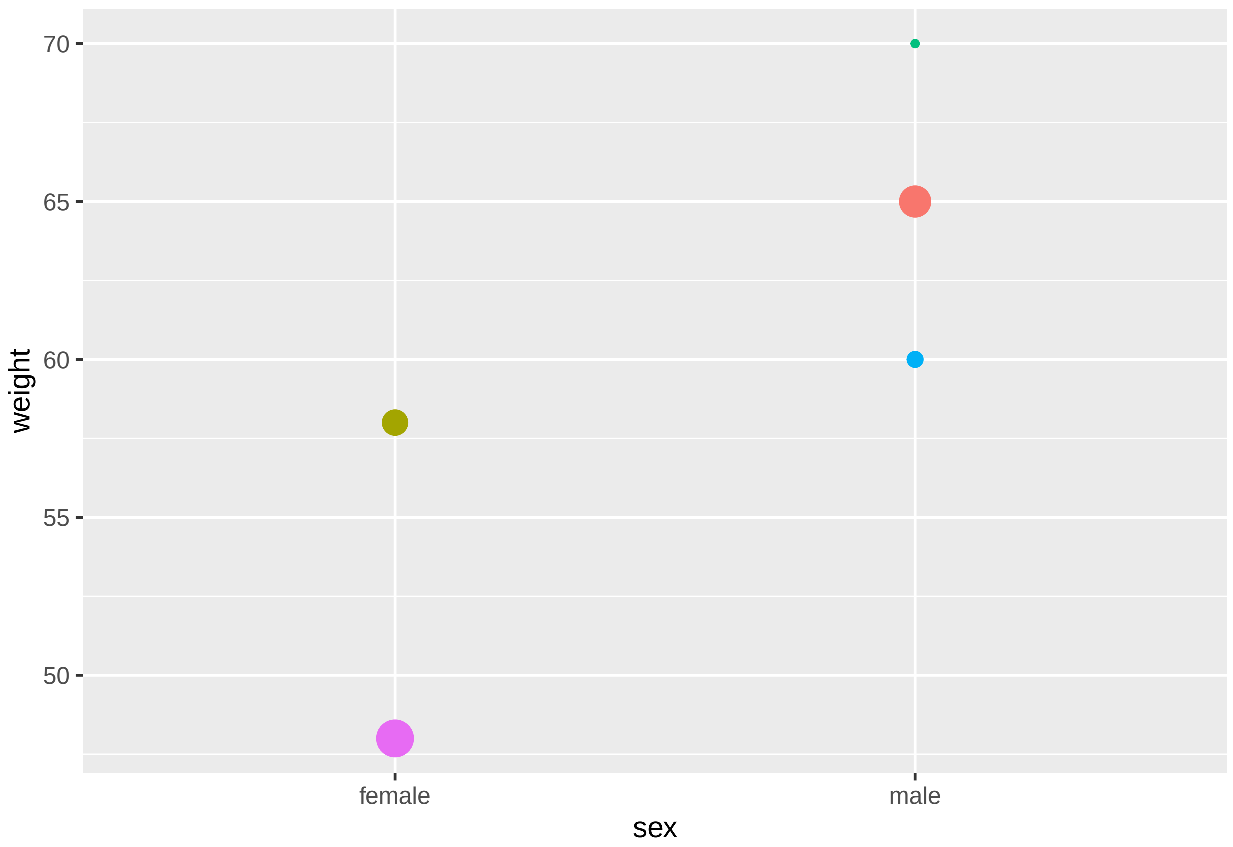
```
# export the file as .pdf
```

7.24 Legend

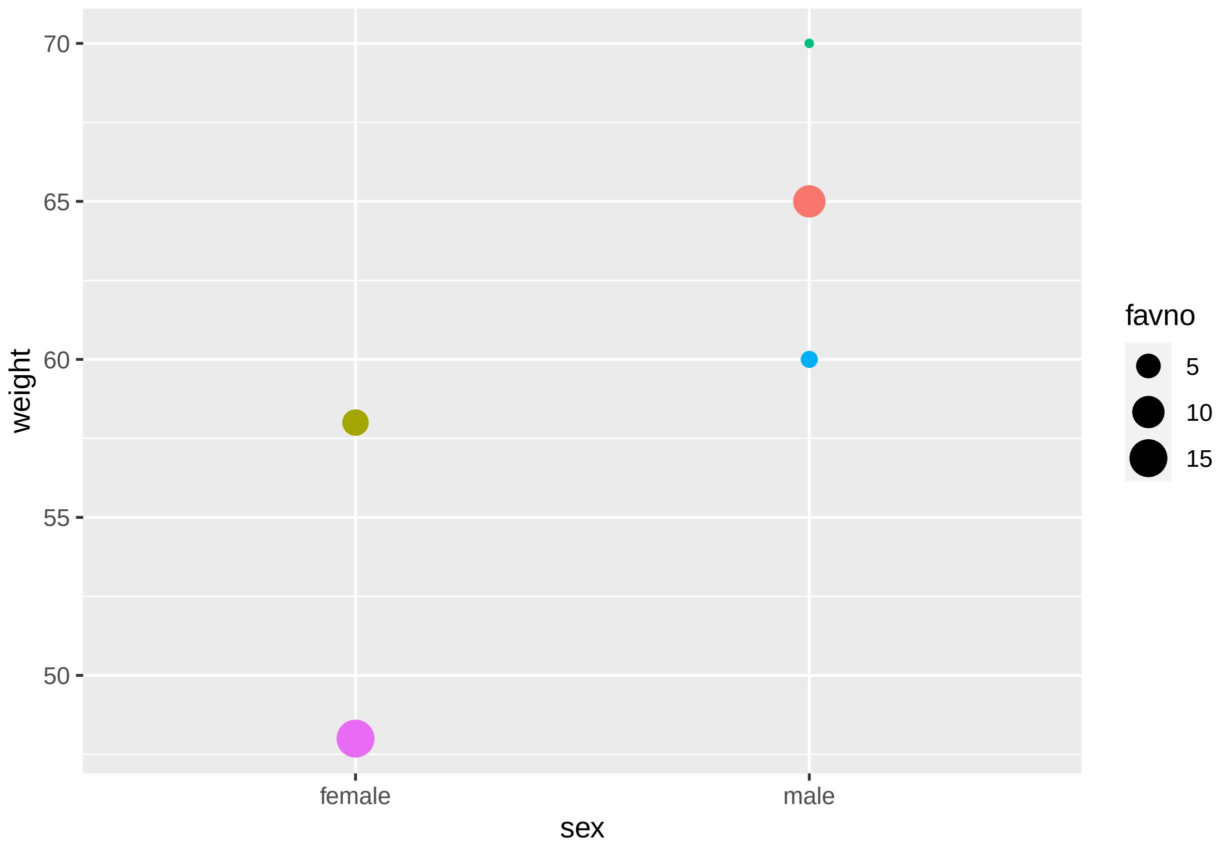
```
df <- data.frame(name = c("saneesh", "kishan", "anil", "mahi",
    "sanusha"), sex = c("male", "female", "male", "male", "female"),
    weight = c(60, 58, 65, 70, 48), favno = c(2, 6, 10, 1, 15))

ggplot(df, aes(x = sex, y = weight, col = name, size = favno,
    shape = sex)) + geom_point()
```



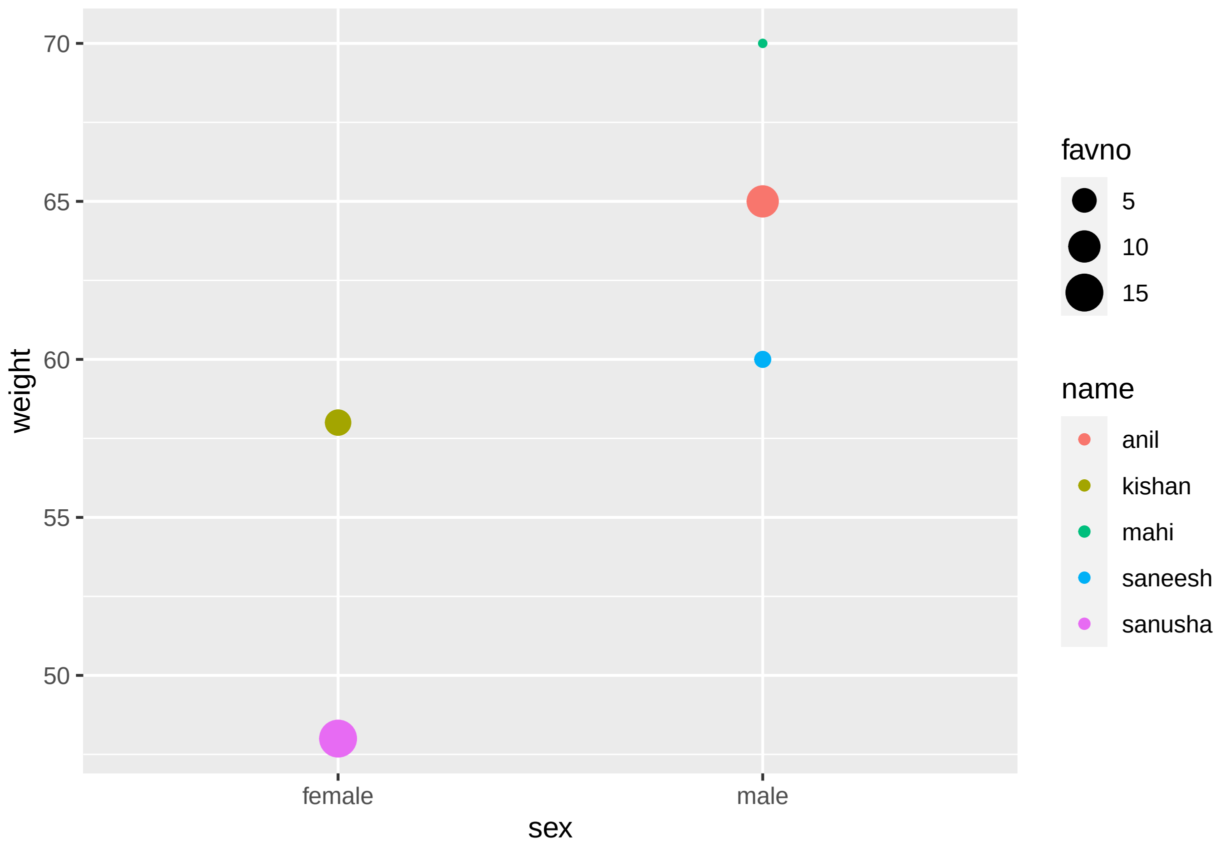
```
# remove all legends
ggplot(df, aes(x = sex, y = weight, col = name, size = favno)) +
    geom_point() + theme(legend.position = "none")
```



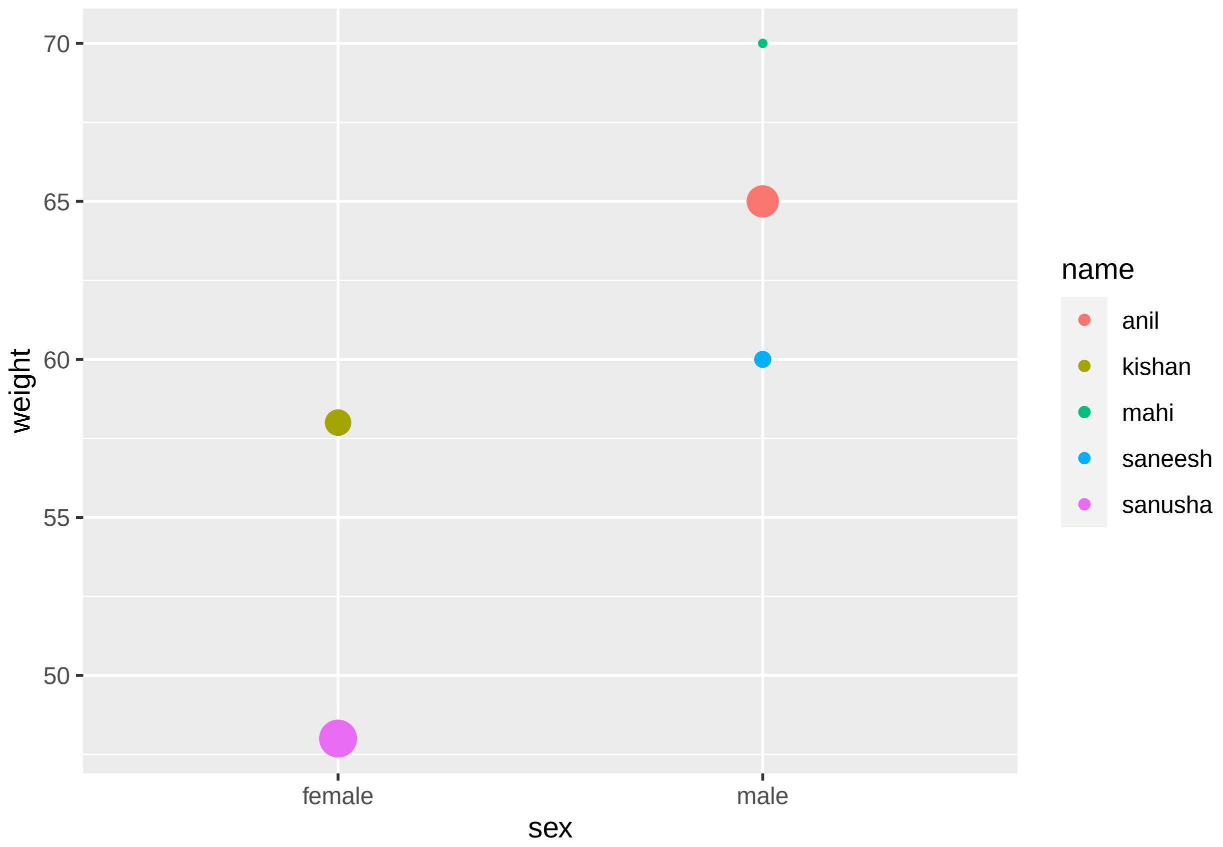
```r
# remove legend created by color
ggplot(df, aes(x = sex, y = weight, col = name, size = favno)) +
    geom_point() + guides(color = "none")
```



```r
# remove legend created by shape
ggplot(df, aes(x = sex, y = weight, col = name, size = favno)) +
    geom_point() + guides(shape = "none")
```

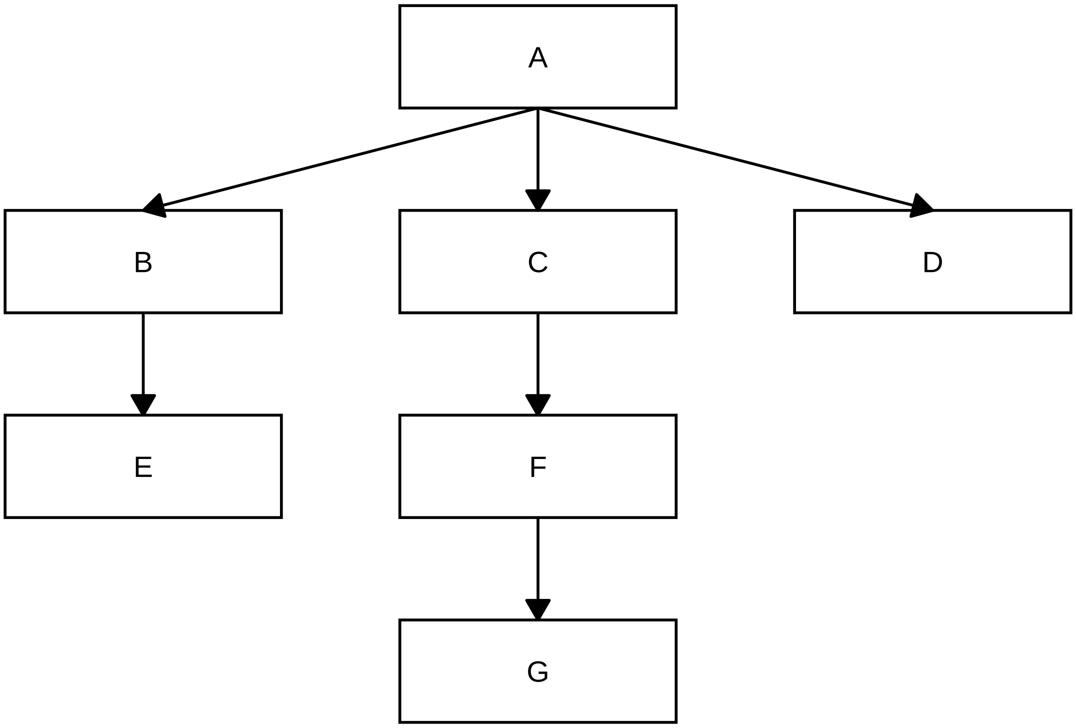


```
# remove legend created by size
ggplot(df, aes(x = sex, y = weight, col = name, size = favno)) +
    geom_point() + guides(size = "none")
```



8 ggflowchart

```
# install.packages('ggflowchart')
library(ggflowchart)

data <- tibble::tibble(from = c("A", "A", "A", "B", "C", "F"),
    to = c("B", "C", "D", "E", "F", "G"))

ggflowchart(data)
```



talk blog

9 Functions

9.0.1 dice

```
dice <- c(1:6)

myluck <- function(x) {
    myluck <- sample(dice, size = 1, replace = T)
    return(myluck)
}

myluck()
```

```
## [1] 6
```

9.0.2 pick a name

```
names <- c("saneesh", "appu", "sanusha")
who <- function(x) {
    who <- sample(names, 1, T)
    return(who)
}

who()
```

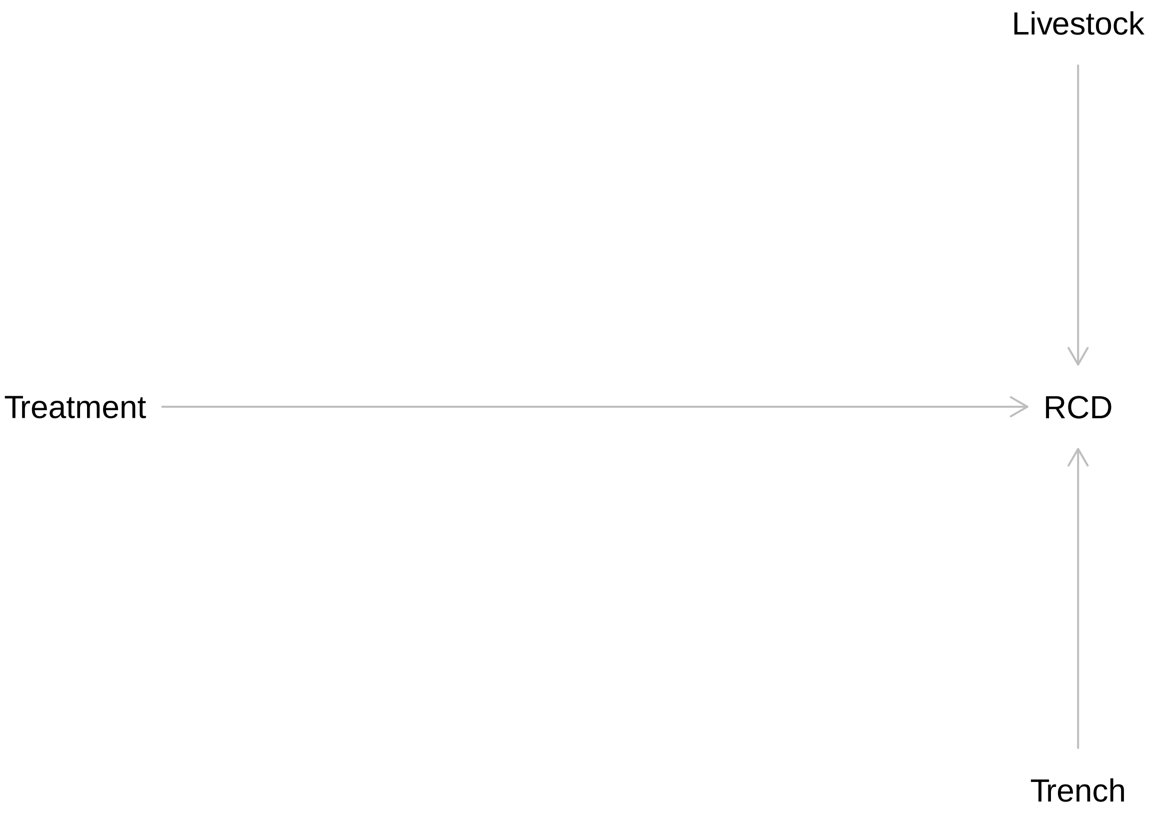
```
## [1] "saneesh"
```

10 DAG

```
library(dagitty)
```

```
##
## Attaching package: 'dagitty'

## The following object is masked from 'package:hablar':
##
##     convert
```

```
sapling <- dagitty("dag{
    Treatment-> RCD <- Livestock;
    Trench -> RCD
}")
coordinates(sapling) <- list(x = c(Treatment = 1, Livestock = 2,
    Trench = 2, RCD = 2  # column 2
), y = c(Treatment = 0,
    RCD = 0, Livestock = -1, Trench = 1))

# Treatment=1 column 1 Livestock= 2, column 2 Trench= 2,
# column 2 RCD=2 column 2

# Treatment=0, middle row/0 RCD=0, middle row/0 Livestock=
# -1, above middle row -1 Trench= 1 below the middle row/1

plot(sapling)
```



11 function to split

```
df <- data.frame(name = as.factor(c("James Bond", "Spider Man",
    "Iron Man")))
# df <- df %>% separate(name, c('Genus', 'Species'), sep =
# '([ ])')

shorten <- function(df) {
    name_split <- df %>%
        separate(name, c("Genus", "Species"), sep = "([ ])")
    print(name_split)
}

shorten(df)
```

```
##    Genus Species
## 1  James    Bond
## 2 Spider     Man
## 3   Iron     Man
```

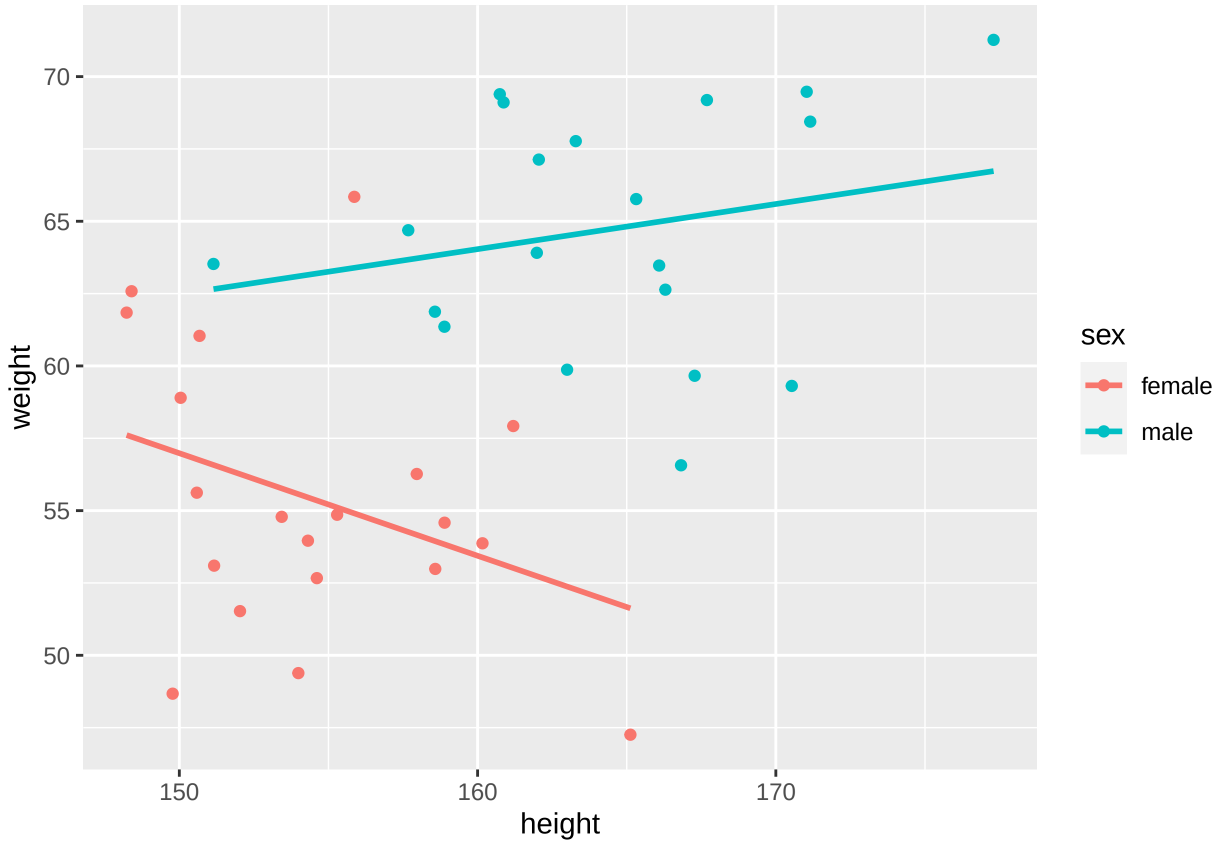
12 Model

12.0.1 Model with interaction

Model interaction

```
library(ggplot2)
library(dplyr)

data <- data.frame(sex = rep(c("male", "female"), each = 20),
    weight = c(rnorm(20, mean = 65, 5), rnorm(20, mean = 55,
        5)), height = c(rnorm(20, mean = 165, 6), rnorm(20, mean = 152,
        sd = 6)))


# Plot the interaction using ggplot2
data %>%
    ggplot(aes(x = height, y = weight, color = sex)) + geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



13 web scraping

```
library(rvest)
```

```
##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
# page <-
# read_html('https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population')
# tables <- html_table(page) typeof(tables) unlist(tables)
# table2 <- as.data.frame(tables[[2]]) head(table2,2)
```

14 Rmarkdown

14.1 knitr golbal options

to apply to every chunk in the file

inside the `chunk` write `knitr::opts_chunk$set(include= ,echo = , message= , warning= )`

```
# knitr::opts_chunk$set(message = TRUE, echo = TRUE,
# warning = TRUE)
```

`include:` to show or hide code and results from appearing
`echo:` to show or hide code in the output but shows result
`message` to hide or show the messages generated by the code
`warning:` to show or hide warning generated by the code

these options can be written for individual chunks as well

```
## [1] 5
```

14.1.1 headings

1 # heading 1
2 ## heading 2 3 ### heading 3

italics
italic

bold
bold

`plot()` to show r code/function
@Saneesh

14.2 blockquotes are writtedn after >

> this is a blockquote
> — Saneesh

14.3 plain code

```
hello
```

14.4 unordered items

- item 1
- item 2
  - sub item 1a
  - sub item 2b

14.5 ordered items

1. Item 1
2. Item 2
   - Item 2a # give two spaces before the +
   - Item 2b

14.6 writing mathematical functions

14.7 adding a link

```
# [mathematical
# notations](https://rpruim.github.io/s341/S19/from-class/MathinRmd.html)
```

$by$ `$by$`
$\mu$ `$\mu$`
$\sum$ `$\sum$`
$a \pm b$ `$a\pm b$`
$x = y$ `$x=y$`
$x > y$ `$x>y$`
x^2 `$x^2$`
$x \le y$ `$x\le y$`

$\sum_{n=1}^{10} n^2$ `$\sum_{n=1}^{10} n^2$`
$LUI_i = \frac{1}{2}(gi/gm) + \frac{1}{2}(ti/tm)$ `$LUI_i=\frac12(gi/gm)+\frac12(ti/tm)$` $x_1 + x_2 + \cdots + x_n$ `$x_{1}+ x_{2}+\cdots+x_{n}$`
$|A|$ `$|A|$`
$A \subset B$ `$A\subset B$`
$A \subseteq B$ `$A \subseteq B$`

$A \cup B$ `$A \cup B$`
$A \cap B$ `$A \cap B$`
$P(A|B)$ `$P(A|B)$`
$\alpha$ `$\alpha$`
$\beta$ `$\beta$`
$\gamma$ `$\gamma$`
$\theta$ `$\theta$`
$H_2O$ `$H_2O$`

14.8 adding image and caption

Figure 1: write

Inside a chunk after three … `r`, `echo=FALSE,out.width="70%",fig.align="center",fig.cap='write'` close the curly bracket, then write knitr::include_graphics(“Idly.jpg”) # keep the image in the project folder, then close the chunk. with “ “’

write an exclamation mark `!`, then square brackets `[caption]` write caption in it, the normal brackets `(Idly.jpg)` write the name of the file and it’s extension i.e., `idly.jpg`

15 Resources

bbcplot
colorhunt
colors
colorpaletts
colorpaletts
coloradobe
colormind
datavizpyr
datatoviz

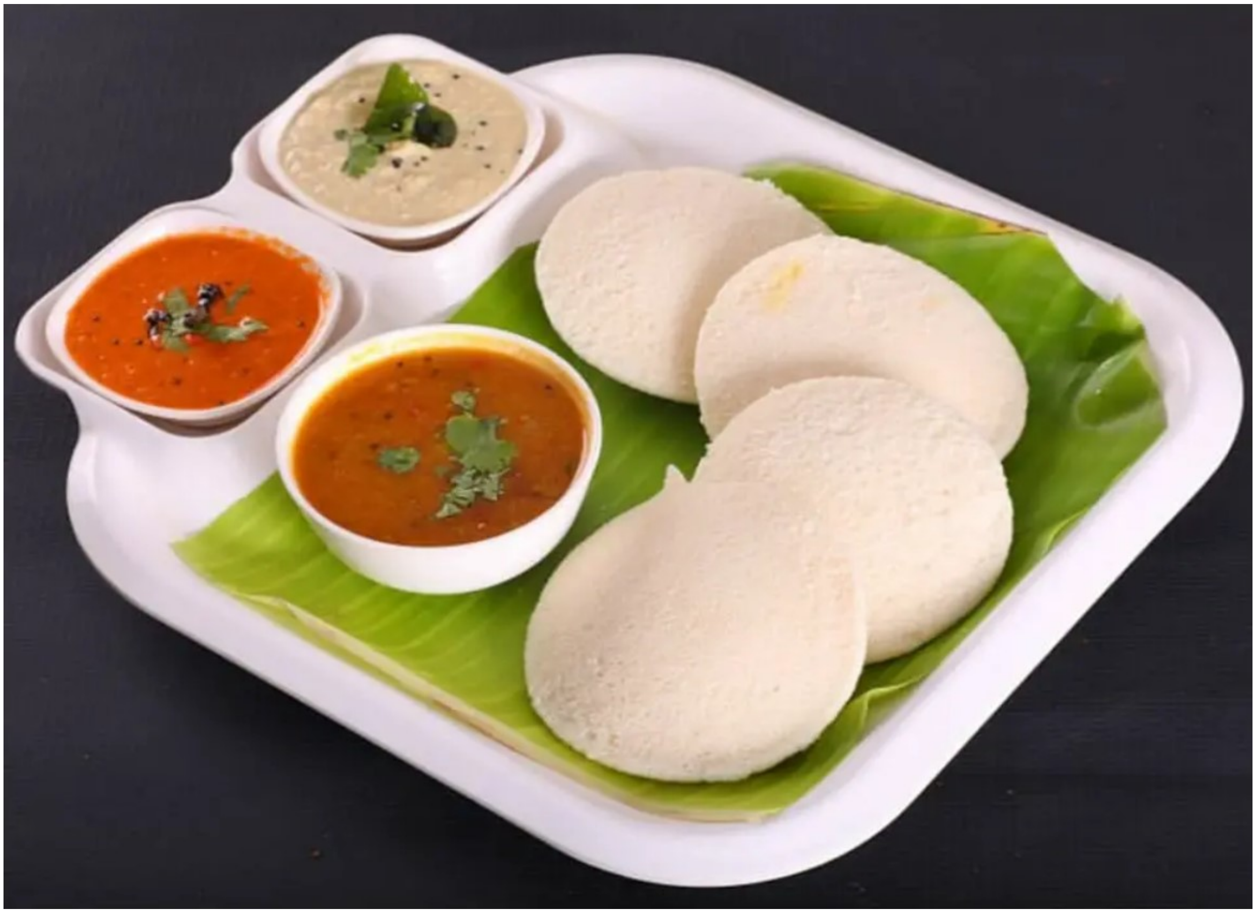


Figure 2: Idly

Cédric Scherer
ggplottheme
mycolor
viz-palette
Intro to r