Introduction to Data Science in R

Mehman Ismayilli

2023-08-17

R basics

```
library(dslabs)
data("murders")
```

Sorting

From page 61

`sort` function

```
sort(murders$total)
```

```
##  [1]    2    4    5    5    7    8   11   12   12   16   19   21   22   27   32
## [16]   36   38   53   63   65   67   84   93   93   97   97   99  111  116  118
## [31]  120  135  142  207  219  232  246  250  286  293  310  321  351  364  376
## [46]  413  457  517  669  805 1257
```

`order` function

```
index <- order(murders$total) ## generates index for murders
states <- murders$abb[index] ## finds abbreviations
```

`max` and `which.max`

```
max(murders$total)
```

```
## [1] 1257
```

```
#find the index number
ind_max <- which.max(murders$total)
#find the particular state
murders$state[ind_max]
```

```
## [1] "California"
```

`rank`

```
rank(murders$total)
```

```
##  [1] 32.0 11.0 36.0 23.5 51.0 20.0 25.5 17.0 27.0 49.0 45.0  5.0  8.5 44.0 33.0
## [16] 12.0 19.0 29.0 43.0  7.0 40.0 30.0 46.0 18.0 31.0 42.0  8.5 15.0 22.0  3.5
## [31] 37.0 21.0 48.0 39.0  2.0 41.0 28.0 16.0 47.0 10.0 34.0  6.0 35.0 50.0 13.0
## [46]  1.0 38.0 23.5 14.0 25.5  3.5
```

Vector arithmetics

California has the highest murder number. Is it so dangerous in California? To do so, let us find murder rate per 100 K in each state

```
murder_rate <- murders$total/murders$population * 100000
```

Now find the state with the highest murder rate for 100 K

```
murders$abb[order(murder_rate)]
```

```
##  [1] "VT" "NH" "HI" "ND" "IA" "ID" "UT" "ME" "WY" "OR" "SD" "MN" "MT" "CO" "WA"
## [16] "WV" "RI" "WI" "NE" "MA" "IN" "KS" "NY" "KY" "AK" "OH" "CT" "NJ" "AL" "IL"
## [31] "OK" "NC" "NV" "VA" "AR" "TX" "NM" "CA" "FL" "TN" "PA" "AZ" "GA" "MS" "MI"
## [46] "DE" "SC" "MD" "MO" "LA" "DC"
```

You can see that the DC has the highest murder rate for 100 K.

Indexing

Now indexing based on a criterion

```
ind <- murder_rate < 0.71
murders$state[ind]
```

```
## [1] "Hawaii"        "Iowa"          "New Hampshire" "North Dakota"
## [5] "Vermont"
```

```
#change the condition
ind <- murder_rate <= 0.71
murders$state[ind]
```

```
## [1] "Hawaii"        "Iowa"          "New Hampshire" "North Dakota"
## [5] "Vermont"
```

```
#count how many in there
sum(ind)
```

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

`which` function indexing

```
ind <- which(murders$state=="California")
murder_rate[ind]
```

```
## [1] 3.374138
```

`match`

```
ind <- match(c("New York", "Florida", "Texas"), murders$state)
murder_rate[ind]
```

```
## [1] 2.667960 3.398069 3.201360
```

`%in$` To check whether a set contains certain elements

```
c("Boston", "Dakota", "Washington") %in% murders$state
```

```
## [1] FALSE FALSE  TRUE
```

Exercise 2.14

```
low <- murder_rate < 1
ind <- which(low == TRUE)
murders$state [ind]
```

```
##  [1] "Hawaii"        "Idaho"         "Iowa"          "Maine"
##  [5] "Minnesota"     "New Hampshire" "North Dakota"  "Oregon"
##  [9] "South Dakota"  "Utah"          "Vermont"       "Wyoming"
```

```
ind <- which(low == TRUE & murders$region=="Northeast")
murders$state[ind]
```

```
## [1] "Maine"         "New Hampshire" "Vermont"
```

```
ind <- which(murder_rate < mean(murder_rate))
length(ind)
```

```
## [1] 27
```

```
ind <- match(c("AK", "MI", "IA"), murders$abb)
murders$state[ind]
```

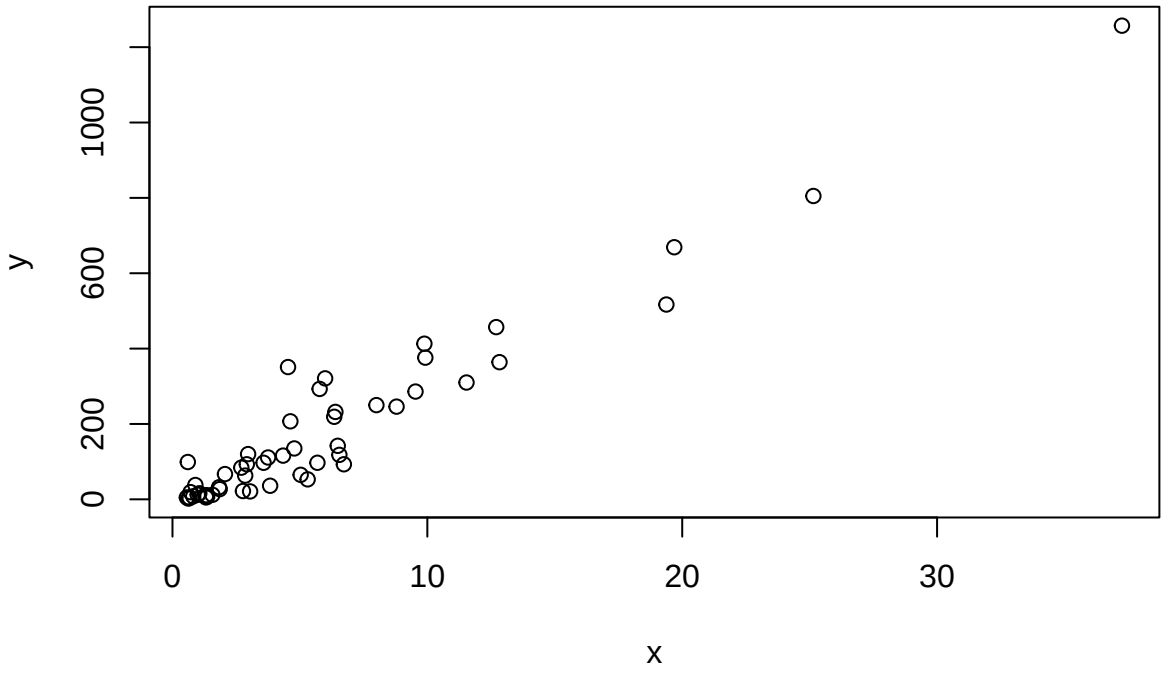
```
## [1] "Alaska"   "Michigan" "Iowa"
```

```
c("MA", "ME", "MI", "MO", "MU") %in% murders$abb
```

```
## [1]  TRUE  TRUE  TRUE  TRUE FALSE
```

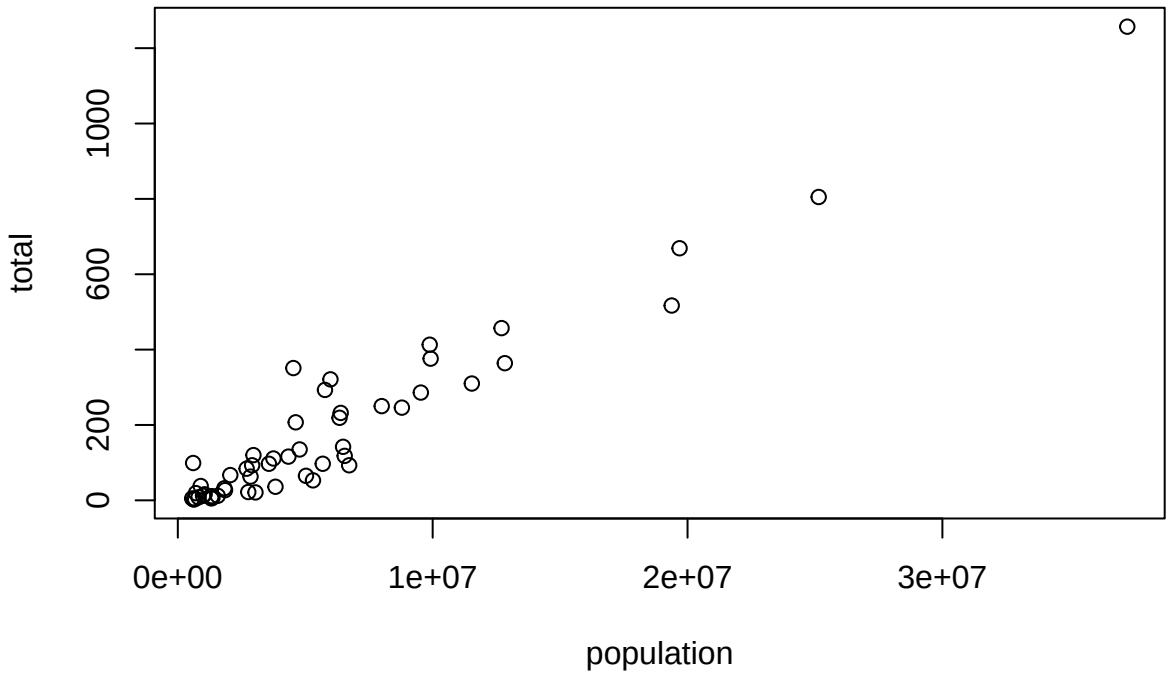
Basic plots

```
x <- murders$population / 10^6
y <- murders$total
plot(x, y)
```



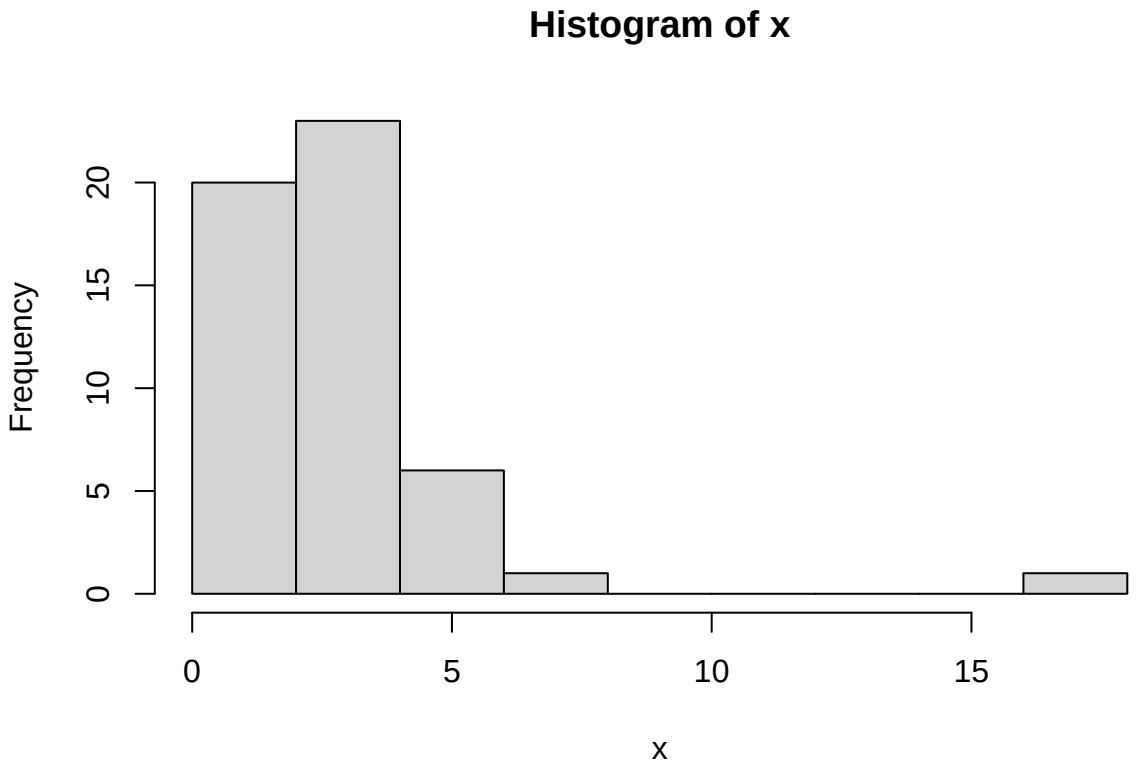
For a quick plot that avoids accessing variables twice, we can use `with` function

```
with(murders, plot(population, total))
```



histogram

```
x <- with(murders, total/population * 100000)
hist(x)
```

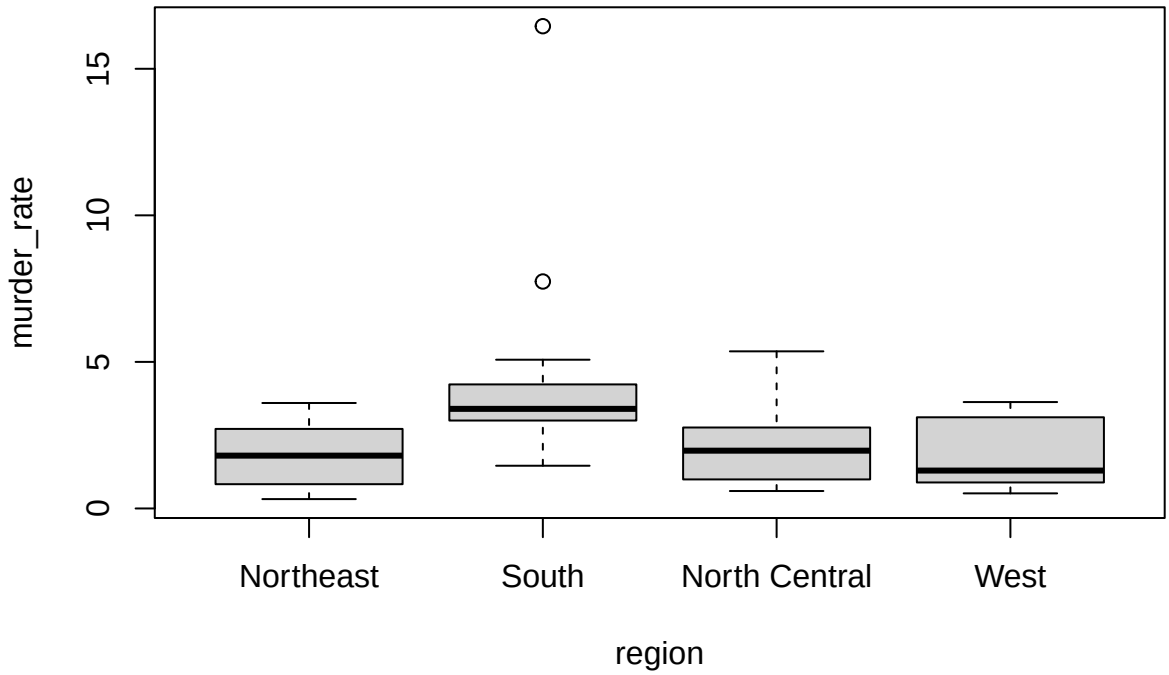



```
murders$state[which.max(x)]
```

```
## [1] "District of Columbia"
```

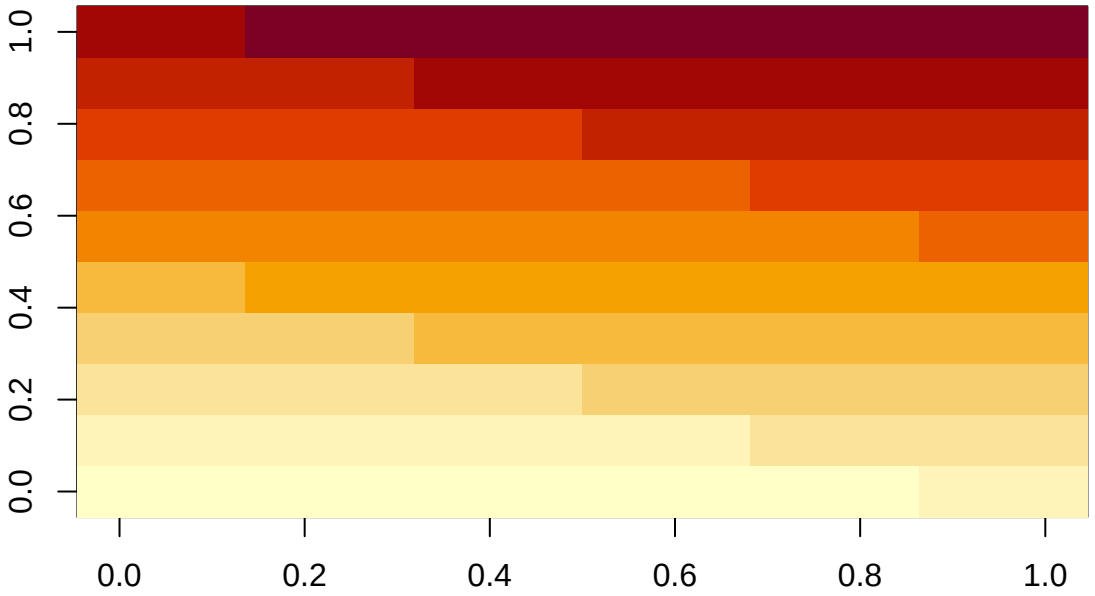
boxplot

```r
boxplot(murder_rate~region, data = murders)
```



image

```r
x <- matrix(1:120, 12, 10)
image(x)
```



Chapter 3 Programming basics

```r
a <- 0

if(a!=0){
  print(1/a)
```

```
} else{
  print("No reciprocal for 0.")
}
```

```
## [1] "No reciprocal for 0."
```

```
ind <- which.min(murder_rate)

if (murder_rate[ind] <0.5){
  print(murders$state[ind])
} else {
  print("No state has murder rate that below")
}
```

```
## [1] "Vermont"
```

A shorter version of `if` function, `ifelse`

```
a <- 0
ifelse(a>0, 1/a, NA)
```

```
## [1] NA
```

```
a <- c(0, 1, 2, -4, 5)
result <- ifelse(a>0, 1/a, NA)
result
```

```
## [1]  NA 1.0 0.5  NA 0.2
```

```
data("na_example")
no_nas <- ifelse(is.na(na_example), 0, na_example)
sum(is.na(no_nas))
```

```
## [1] 0
```

`any` and `all` functions

```
z <- c(TRUE, TRUE, FALSE)
any(z)
```

```
## [1] TRUE
```

```
all(z)
```

```
## [1] FALSE
```

Defining functions

```
avg <- function(x){
  s <- sum(x)
  n <- length(x)
  s/n
}
```

```
x <- 1:100
avg(x)
```

```
## [1] 50.5
```

```
identical(mean(x), avg(x))
```

```
## [1] TRUE
```

A bit complex function

```
avg <- function(x, arithmetic=TRUE){
  n <- length(x)
  ifelse(arithmetic, sum(x)/n, prod(x)^(1/n))
}
avg(x, arithmetic = FALSE)
```

```
## [1] 37.99269
```

For-loops

```
compute_s_n <- function(n){
  x <- 1:n
  sum(x)
}
compute_s_n(100)
```

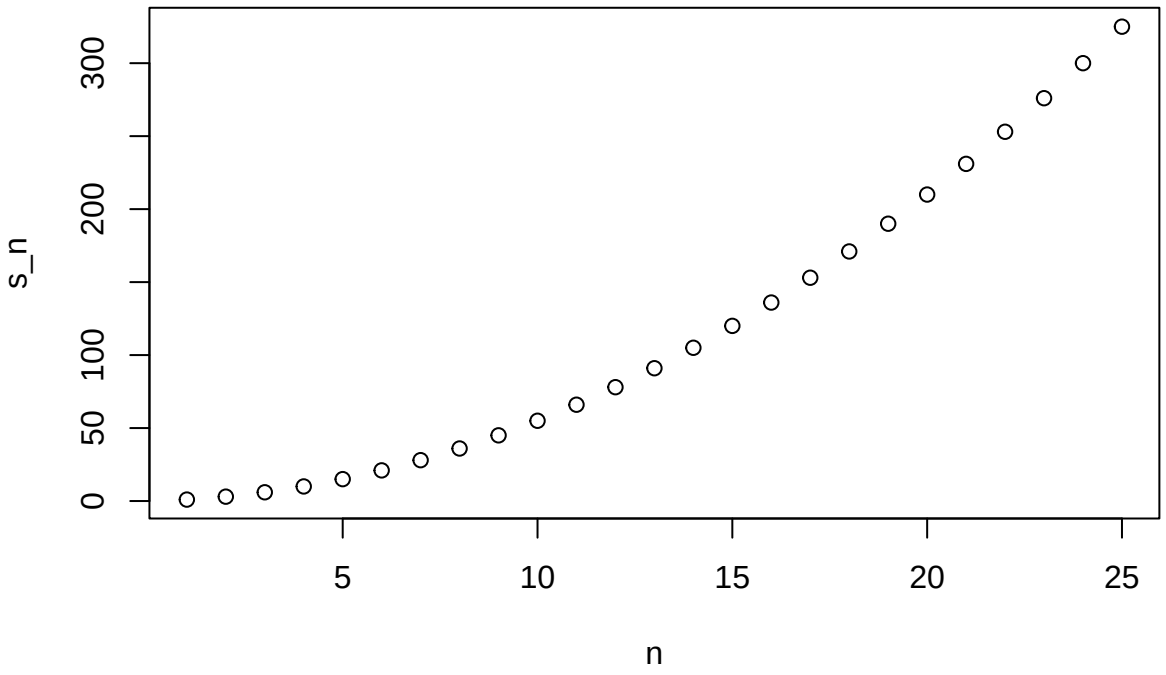
```
## [1] 5050
```

```
for(i in 3){
  print(i)
}
```

```
## [1] 3
```

```
m <- 25
s_n <- vector(length = m) #create an empty vector
for(n in 1:m){
  s_n[n] <- compute_s_n(n)
}
```

```
n <- 1:m
plot(n, s_n)
```



Vectorisation and functionals

```
x<- 1:10
sqrt(x)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

```
y <- 1:10
x*y
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

The following piece of code does not run the function on each entry of `n`.

```
n <- 1:25
compute_s_n(n)
```

```
## Warning in 1:n: numerical expression has 25 elements: only the first used
```

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

```
x <- 1:10
sapply(x, sqrt)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

fix the problem above

```
n <- 1:25
s_n <- sapply(n, compute_s_n)
s_n
```

```
##  [1]   1   3   6  10  15  21  28  36  45  55  66  78  91 105 120 136 153 171 190
## [20] 210 231 253 276 300 325
```

The tidyverse

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Create a new column by `mutate`

```
murders <- mutate(murders, rate = total/population*100000)
```

Subsetting with `filter`

```
filter(murders, rate <=0.81)
```

```
##           state abb        region population total      rate
## 1        Hawaii  HI          West    1360301     7 0.5145920
## 2         Idaho  ID          West    1567582    12 0.7655102
## 3          Iowa  IA North Central    3046355    21 0.6893484
## 4 New Hampshire  NH     Northeast    1316470     5 0.3798036
## 5  North Dakota  ND North Central     672591     4 0.5947151
## 6          Utah  UT          West    2763885    22 0.7959810
## 7       Vermont  VT     Northeast     625741     2 0.3196211
```

Selecting columns with `select` and create a new data tables

```
new_table <- select(murders, state, region, rate)
filter(new_table, rate <= .71)
```

```
##           state        region      rate
## 1        Hawaii          West 0.5145920
## 2          Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5       Vermont     Northeast 0.3196211
```

```
murders <- mutate(murders, pop_in_millions = population/10^6)
```

```
select(murders, state, population) %>%head()
```

```
##        state population
## 1    Alabama    4779736
## 2     Alaska     710231
## 3    Arizona    6392017
## 4   Arkansas    2915918
## 5 California   37253956
## 6   Colorado    5029196
```

```
filter(murders, state == "New York")
```

```
##      state abb    region population total    rate pop_in_millions
```

```
## 1 New York  NY Northeast   19378102   517 2.66796         19.3781
```

Choose the data excluding "Florida"

```
no_florida <- filter(murders, state != "Florida")
```

```
filter(murders, population < 5000000 & region == "Northeast")
```

```
##           state abb    region population total      rate pop_in_millions
## 1   Connecticut  CT Northeast    3574097    97 2.7139722        3.574097
## 2         Maine  ME Northeast    1328361    11 0.8280881        1.328361
## 3 New Hampshire  NH Northeast    1316470     5 0.3798036        1.316470
## 4  Rhode Island  RI Northeast    1052567    16 1.5200933        1.052567
## 5       Vermont  VT Northeast     625741     2 0.3196211        0.625741
```

```
filter(murders, state %in% c("New York", "Texas"))
```

```
##      state abb    region population total    rate pop_in_millions
## 1 New York  NY Northeast   19378102   517 2.66796        19.37810
## 2    Texas  TX     South   25145561   805 3.20136        25.14556
```

The pipe %>%

```
murders %>% select(state, region, rate) %>% filter(rate <= .71)
```

```
##           state        region      rate
## 1        Hawaii          West 0.5145920
## 2          Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5       Vermont     Northeast 0.3196211
```

```
16 %>% sqrt() %>% log2()
```

```
## [1] 2
```

Summarising data

```
data("heights")
```

Average heights and st dev of females

```
s <- heights %>%
  filter(sex == "Female") %>%
  summarise(average = mean(height), st_dev = sd(height))
s
```

```
##    average   st_dev
## 1 64.93942 3.760656
```

```
s$average
```

```
## [1] 64.93942
```

```
s$st_dev
```

```
## [1] 3.760656
```

```
heights %>% filter(sex == "Female") %>%
  filter(sex == "Female") %>%
```

```
  summarise(median = median(height),
            minimum = min(height),
            maximum = max(height))
```

```
##     median minimum maximum
## 1 64.98031      51      79
```

```
heights %>%
  filter(sex == "Female") %>%
  summarise(range = quantile(height, c(0, 0.5, 1)))
```

```
## 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.
```

```
##      range
## 1 51.00000
## 2 64.98031
## 3 79.00000
```

US murder rate

```
us_murder_rate <- murders %>%
  summarise(rate = sum(total)/sum(population)*100000)
us_murder_rate
```

```
##       rate
## 1 3.034555
```

`pull`

```
class(us_murder_rate)
```

```
## [1] "data.frame"
```

`dplyr` returns outcome in a dataframe but we want to `pull` the output

```
us_murder_rate %>% pull(rate)
```

```
## [1] 3.034555
```

which is equivalent of `us_murder_rate$rate`. To get the value immediated

```
us_murder_rate <- murders %>%
  summarise(rate = sum(total)/sum(population)*100000) %>%
  pull(rate)
us_murder_rate
```

```
## [1] 3.034555
```

Group then summarise with `group_by`

Split data into *groups* and then compute summaries for each group.

```
heights %>% group_by(sex) %>%
  summarise(average = mean(height), standard_deviation = sd(height))
```

```
## # A tibble: 2 x 3
##   sex    average standard_deviation
##   <fct>    <dbl>              <dbl>
## 1 Female    64.9               3.76
## 2 Male      69.3               3.61
```

```
murders %>% group_by(region) %>%
  summarise(median_rate = median(rate))
```

```
## # A tibble: 4 x 2
##   region        median_rate
##   <fct>               <dbl>
## 1 Northeast            1.80
## 2 South                3.40
## 3 North Central        1.97
## 4 West                 1.29
```

Sorting data frames

```
murders %>% arrange(population) %>% head()
```

```
##                  state abb        region population total       rate
## 1              Wyoming  WY          West     563626     5  0.8871131
## 2 District of Columbia  DC         South     601723    99 16.4527532
## 3              Vermont  VT     Northeast     625741     2  0.3196211
## 4         North Dakota  ND North Central     672591     4  0.5947151
## 5               Alaska  AK          West     710231    19  2.6751860
## 6         South Dakota  SD North Central     814180     8  0.9825837
##   pop_in_millions
## 1        0.563626
## 2        0.601723
## 3        0.625741
## 4        0.672591
## 5        0.710231
## 6        0.814180
```

```
murders %>% arrange(rate) %>% head()
```

```
##           state abb        region population total      rate pop_in_millions
## 1       Vermont  VT     Northeast     625741     2 0.3196211        0.625741
## 2 New Hampshire  NH     Northeast    1316470     5 0.3798036        1.316470
## 3        Hawaii  HI          West    1360301     7 0.5145920        1.360301
## 4  North Dakota  ND North Central     672591     4 0.5947151        0.672591
## 5          Iowa  IA North Central    3046355    21 0.6893484        3.046355
## 6         Idaho  ID          West    1567582    12 0.7655102        1.567582
```

top_n function Top n ordering, e.g., top 5 highest murder rate

```
murders %>% top_n(5, rate)
```

```
##                  state abb        region population total      rate
## 1 District of Columbia  DC         South     601723    99 16.452753
## 2            Louisiana  LA         South    4533372   351  7.742581
## 3             Maryland  MD         South    5773552   293  5.074866
## 4             Missouri  MO North Central    5988927   321  5.359892
## 5       South Carolina  SC         South    4625364   207  4.475323
##   pop_in_millions
```

```
## 1        0.601723
## 2        4.533372
## 3        5.773552
## 4        5.988927
## 5        4.625364
```

```
murders %>% top_n(5, desc(rate))
```

```
##           state abb        region population total      rate pop_in_millions
## 1        Hawaii  HI          West    1360301     7 0.5145920        1.360301
## 2          Iowa  IA North Central    3046355    21 0.6893484        3.046355
## 3 New Hampshire  NH     Northeast    1316470     5 0.3798036        1.316470
## 4  North Dakota  ND North Central     672591     4 0.5947151        0.672591
## 5       Vermont  VT     Northeast     625741     2 0.3196211        0.625741
```

```
library(NHANES)
data(NHANES)
```

```
ref <- NHANES %>% filter(AgeDecade == " 20-29") %>%
          summarise(average = mean(BPSysAve, na.rm=TRUE), st_dev = sd(BPSysAve, na.rm = TRUE))
ref
```

```
## # A tibble: 1 x 2
##   average st_dev
##     <dbl>  <dbl>
## 1    113.   11.7
```

```
ref_avg <- ref %>% pull(average)
ref_avg
```

```
## [1] 113.1583
```

```
NHANES %>% filter(AgeDecade == " 20-29") %>%
  summarise(max_pressure = max(BPSysAve, na.rm = TRUE),
            min_pressure = min(BPSysAve, na.rm = TRUE)) %>%
  pull(max_pressure, min_pressure)
```

```
##  84
## 179
```

```
NHANES %>% filter(Gender == "female") %>%
  group_by(AgeDecade) %>%
  summarise(average = mean(BPSysAve, na.rm=TRUE), st_dev = sd(BPSysAve, na.rm = TRUE))
```

```
## # A tibble: 9 x 3
##   AgeDecade average st_dev
##   <fct>       <dbl>  <dbl>
## 1 " 0-9"       100.   9.07
## 2 " 10-19"     104.   9.46
## 3 " 20-29"     108.  10.1
## 4 " 30-39"     111.  12.3
## 5 " 40-49"     115.  14.5
## 6 " 50-59"     122.  16.2
## 7 " 60-69"     127.  17.1
## 8 " 70+"       134.  19.8
## 9  <NA>        142.  22.9
```

```
NHANES %>% filter(Gender == "male") %>%
  group_by(AgeDecade) %>%
```

```
  summarise(average = mean(BPSysAve, na.rm=TRUE), st_dev = sd(BPSysAve, na.rm = TRUE))
```

```
## # A tibble: 9 x 3
##   AgeDecade average st_dev
##   <fct>       <dbl>  <dbl>
## 1 " 0-9"       97.4   8.32
## 2 " 10-19"    110.   11.2
## 3 " 20-29"    118.   11.3
## 4 " 30-39"    119.   12.3
## 5 " 40-49"    121.   14.0
## 6 " 50-59"    126.   17.8
## 7 " 60-69"    127.   17.5
## 8 " 70+"      130.   18.7
## 9  <NA>       136.   23.5
```

`do` function

```
my_summary <- function(dat){
  x <- quantile(heights$height, c(0, 0.5, 1))
  tibble(min=x[1], median = x[2], max = x[3])
}
```

```
heights %>%
  group_by(sex) %>%
  my_summary
```

```
## # A tibble: 1 x 3
##     min median   max
##   <dbl>  <dbl> <dbl>
## 1    50   68.5  82.7
```

However, we want this outcome for each sex; therefore, we can use `do` function instead

```
heights %>%
  group_by(sex) %>%
  do(my_summary(.))
```

```
## # A tibble: 2 x 4
## # Groups:   sex [2]
##   sex      min median   max
##   <fct>  <dbl>  <dbl> <dbl>
## 1 Female    50   68.5  82.7
## 2 Male      50   68.5  82.7
```

if we do not use dot then it would not return any outcome.

purr package

It provides much advanced version of `sapply` functions to receive each ourput withing specified classification.

```
library("purrr")
```

```
n <- 25
s_n <- sapply(n, compute_s_n)
class(s_n)
```

```
## [1] "integer"
```

`map()` works as `sapply` but always returns list

```
s_n <- map(n, compute_s_n)
s_n
```

```
## [[1]]
## [1] 325
```

for a numeric vector

```
s_n <- map_dbl(n, compute_s_n)
s_n
```

```
## [1] 325
```

Tidyverse conditionals

case_when

```
x <- c(-2, -1, 0, 1, 2)
case_when(x<0 ~ "Negative", x>0 ~ "Positive", TRUE ~ "Zero")
```

```
## [1] "Negative" "Negative" "Zero"     "Positive" "Positive"
```

A more advanced but practical example showing how states are located at different regions

```
murders %>%
  mutate(group = case_when(
    abb %in% c("ME", "NH", "VT", "MA", "RI", "CT") ~ "New England",
    abb %in% c("WA", "OR", "CA") ~ "West Coast",
    region == "South" ~ "South",
    TRUE ~ "Other")) %>%
  group_by(group) %>%
  summarise(rate = sum(total)/sum(population)*10^5)
```

```
## # A tibble: 4 x 2
##   group        rate
##   <chr>       <dbl>
## 1 New England  1.72
## 2 Other        2.71
## 3 South        3.63
## 4 West Coast   2.90
```

Exercise 4.15

part 1

```
is_tibble(murders)
```

```
## [1] FALSE
```

```
class(murders)
```

```
## [1] "data.frame"
```

part 2

```
murders_tibble <- as_tibble(murders)
```

part 3

```
murders %>%
  group_by(region)
```

```
## # A tibble: 51 x 7
## # Groups:   region [4]
##    state                abb   region    population total  rate pop_in_millions
##    <chr>                <chr> <fct>          <dbl> <dbl> <dbl>            <dbl>
##  1 Alabama              AL    South        4779736   135  2.82            4.78
##  2 Alaska               AK    West          710231    19  2.68            0.710
##  3 Arizona              AZ    West         6392017   232  3.63            6.39
##  4 Arkansas             AR    South        2915918    93  3.19            2.92
##  5 California           CA    West        37253956  1257  3.37           37.3
##  6 Colorado             CO    West         5029196    65  1.29            5.03
##  7 Connecticut          CT    Northeast    3574097    97  2.71            3.57
##  8 Delaware             DE    South         897934    38  4.23            0.898
##  9 District of Columbia DC    South         601723    99 16.5             0.602
## 10 Florida              FL    South       19687653   669  3.40           19.7
## # i 41 more rows
```

part 4

```
murders %>%
  pull(population) %>% log %>% mean %>% exp
```

```
## [1] 3675209
```

Importing data

Paths and the working directory

```
filename <- "murders.csv"
dir <- system.file("extdata", package = "dslabs")
fullpath <- file.path(dir, filename)
file.copy(fullpath, "murders.csv")
```

```
## [1] FALSE
```

Working directory

```
wd <- getwd()
```

ggplot2

```
library(ggplot2)
```

Now let us develop a graph based step-by-step by calling `ggplot` objects

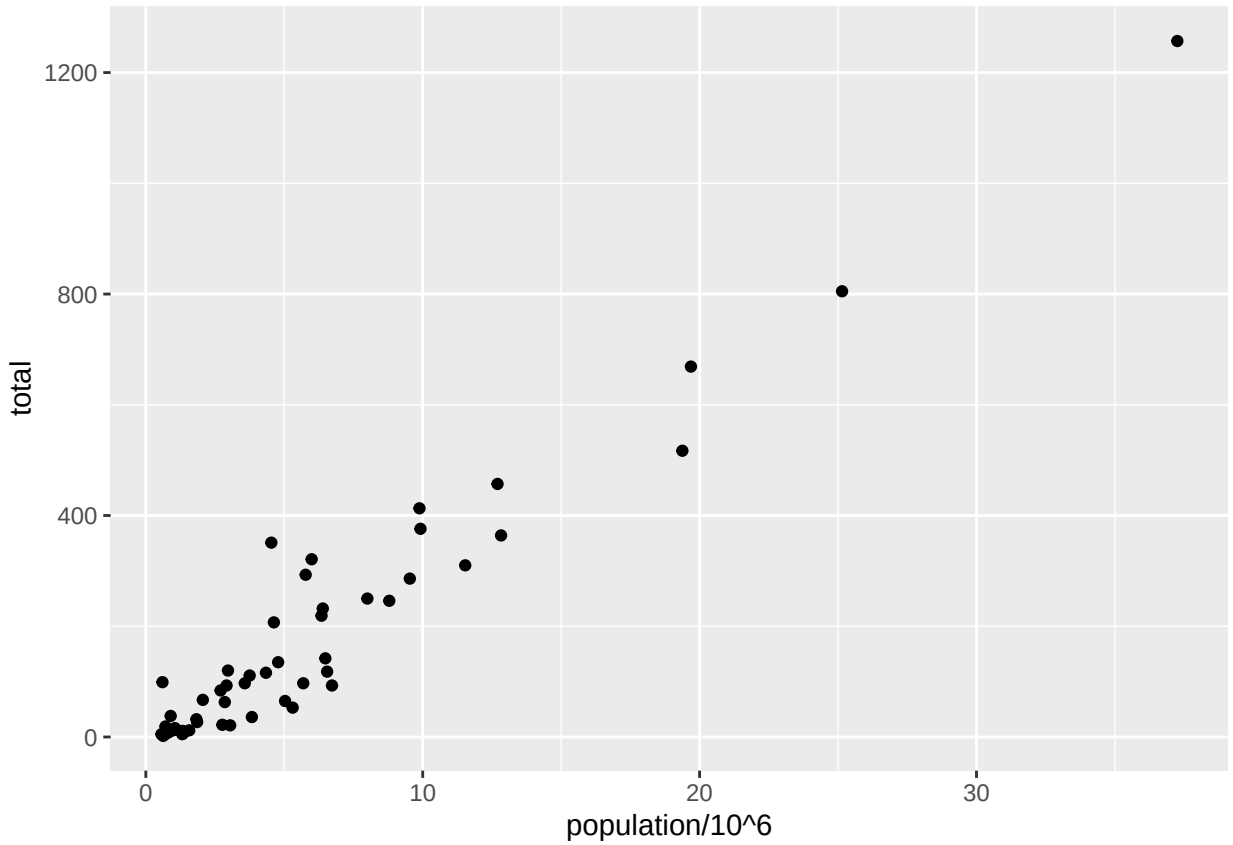
```
#first define the data
murders %>% ggplot() #alternatively, ggplot(data= murders)
```

No geometry is defined therefore it produces only a blank gray canvas. Now let us assign the plot to an object

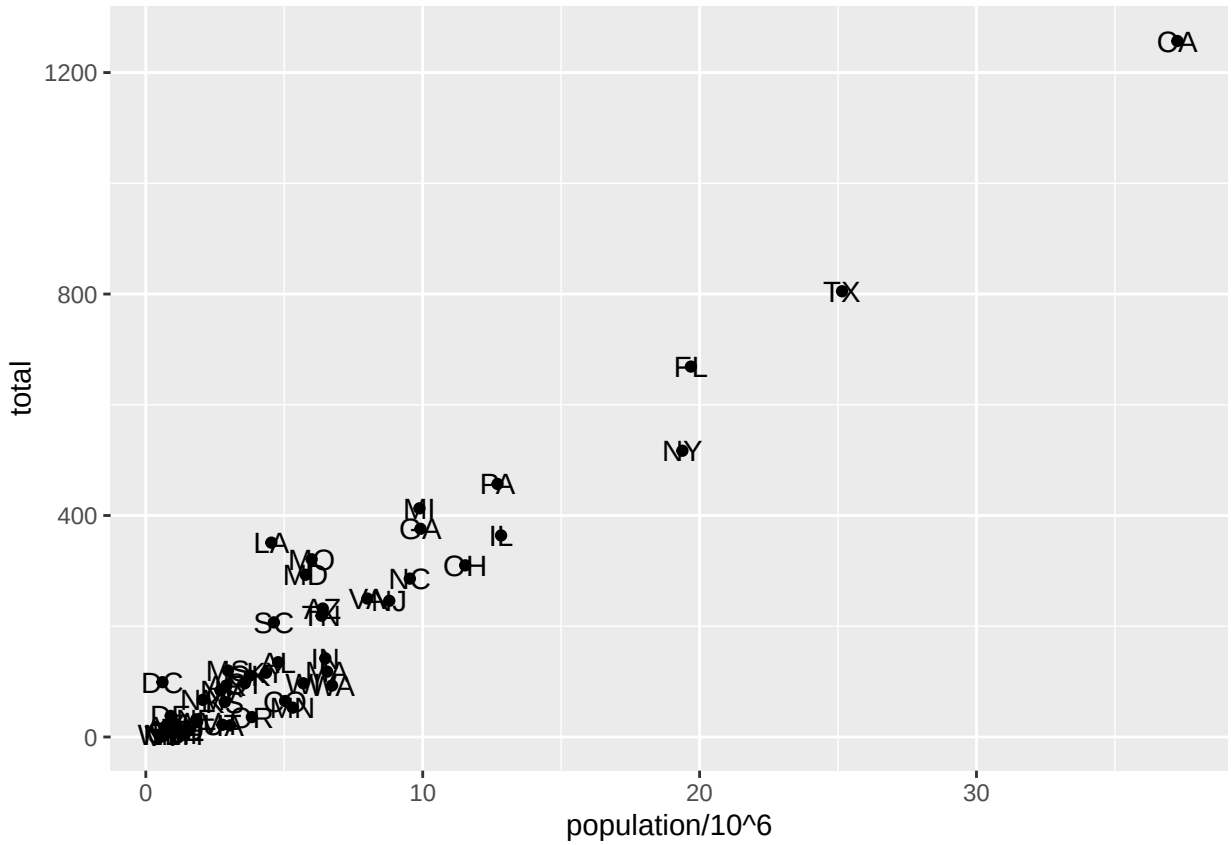
```
p <- murders %>% ggplot()
class(p)
```

```
## [1] "gg"     "ggplot"
```

```
p <- murders %>% ggplot() +
  geom_point(aes(x=population/10^6, y=total))

p
```

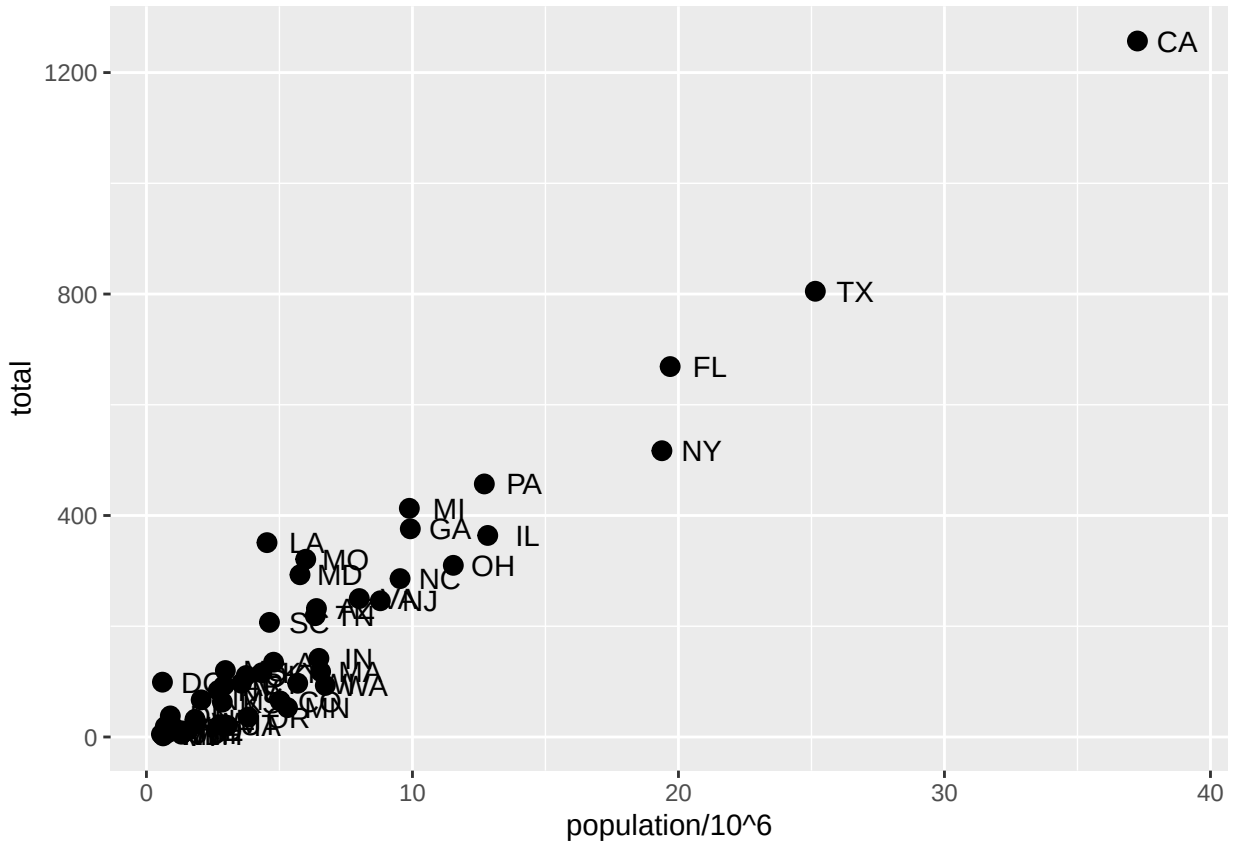



```
#now label the points
p+geom_text(aes(x=population/10^6, y=total, label= abb))
```



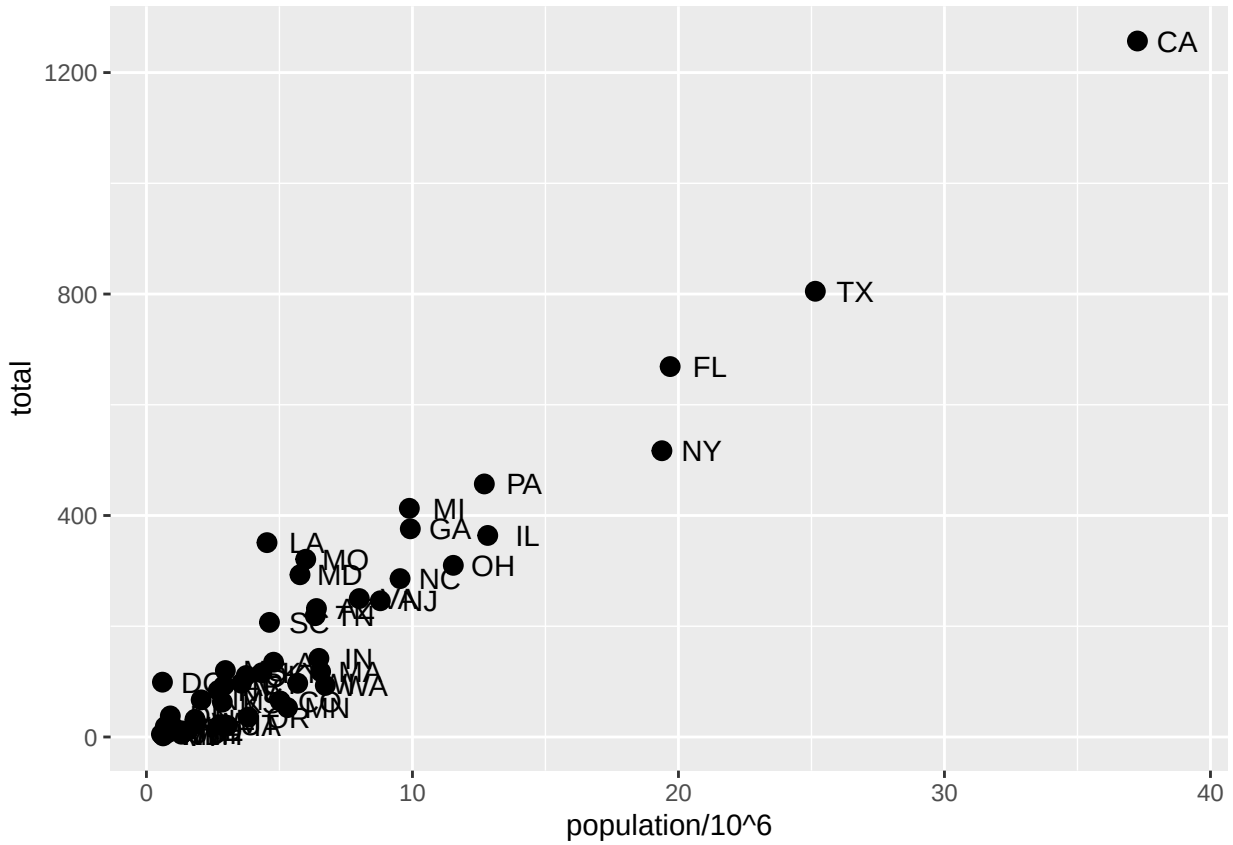
Final graph with all codes

```
p <- murders %>% ggplot() +
  geom_point(aes(x=population/10^6, y=total), size = 3) +
  geom_text(aes(x=population/10^6, y=total, label=abb), nudge_x = 1.5)
p
```



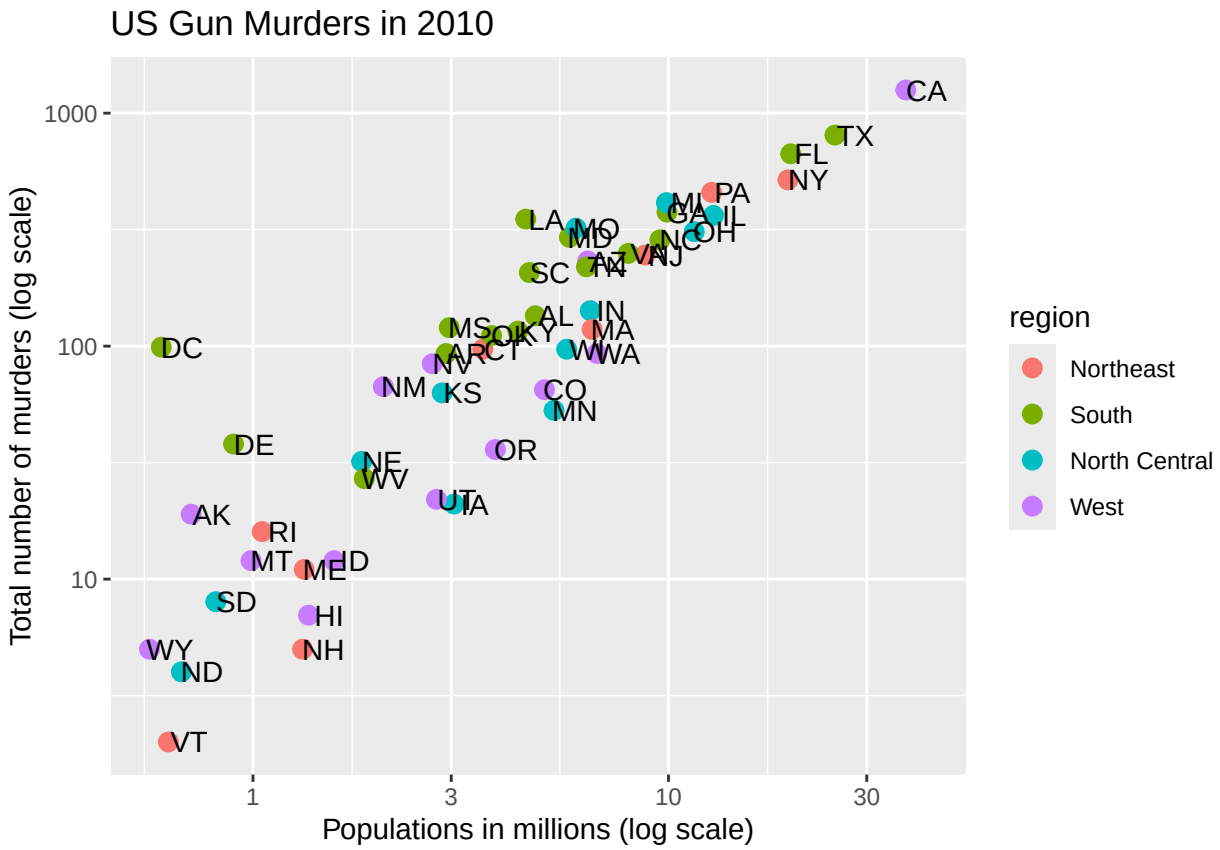
Previously we defined the mapping `aes()` twice, locally. However, let us now define it *globally*.

```
p <- murders %>% ggplot(aes(population/10^6, total, label = abb)) +
  geom_point(size=3) +
  geom_text(nudge_x = 1.5)

p
```



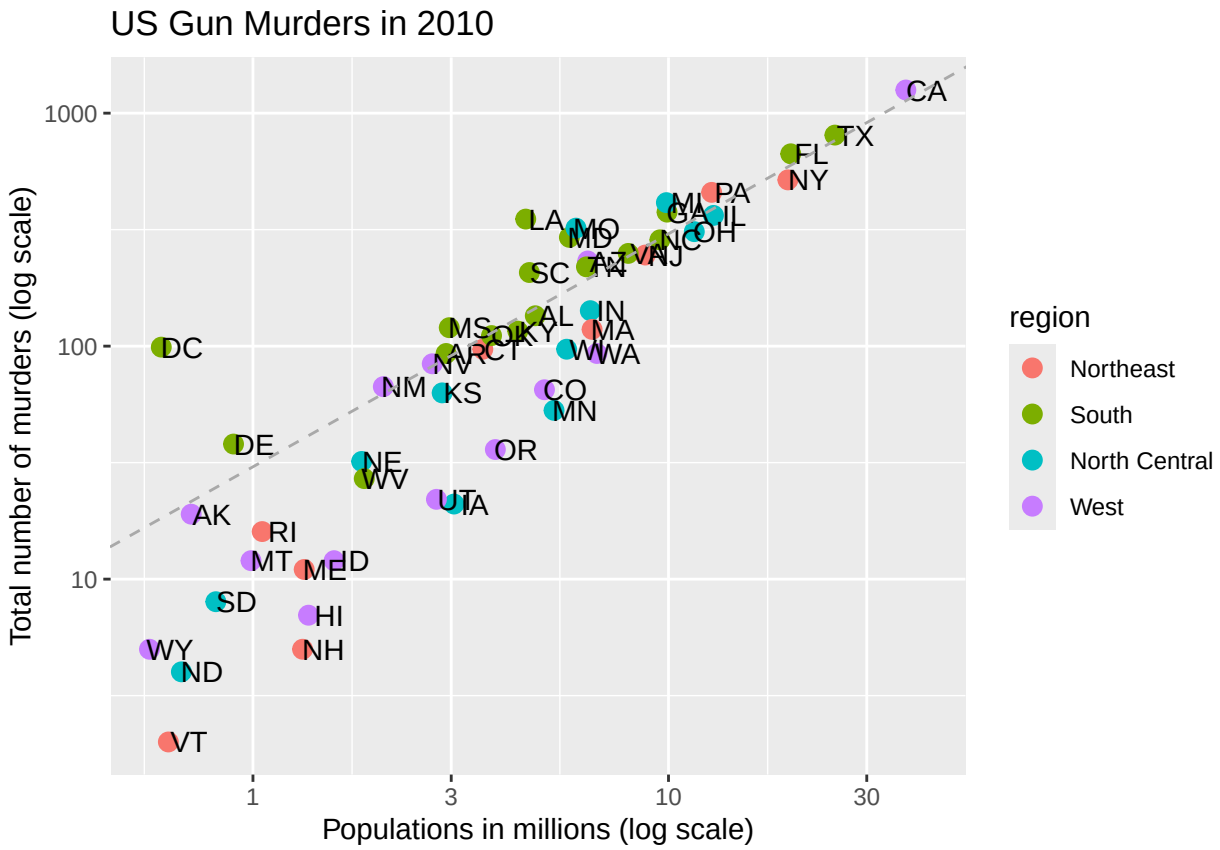
Scales

```
p <- murders %>% ggplot(aes(population/10^6, total, label = abb)) +
  geom_point(aes(col=region),size=3) +
  geom_text(nudge_x =0.05)+
  scale_x_log10() + #scale_x_continuous(trans = "log10") +
  scale_y_log10()+  #scale_y_continuous(trans = "log10")
  xlab("Populations in millions (log scale)") +
  ylab("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010")

p
```



Annotation, shapes and adjustments

```
r <- murders %>%
  summarise(rate = sum(total)/sum(population)*10^6) %>%
  pull(rate)

#Now add the line into the graph
p+geom_abline(intercept = log10(r), lty=2, color = "darkgrey")
```

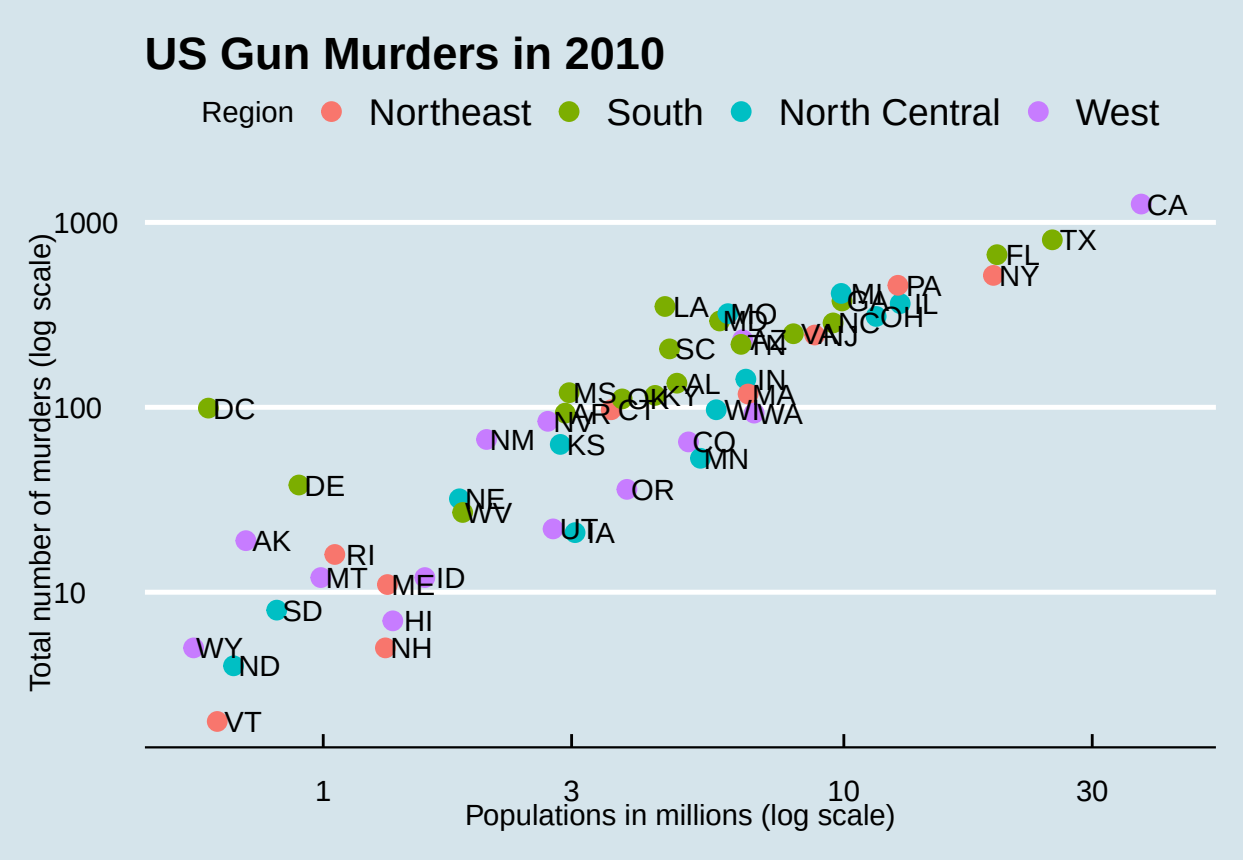



```
p <- p+scale_color_discrete(name="Region") #changes color legend title
```

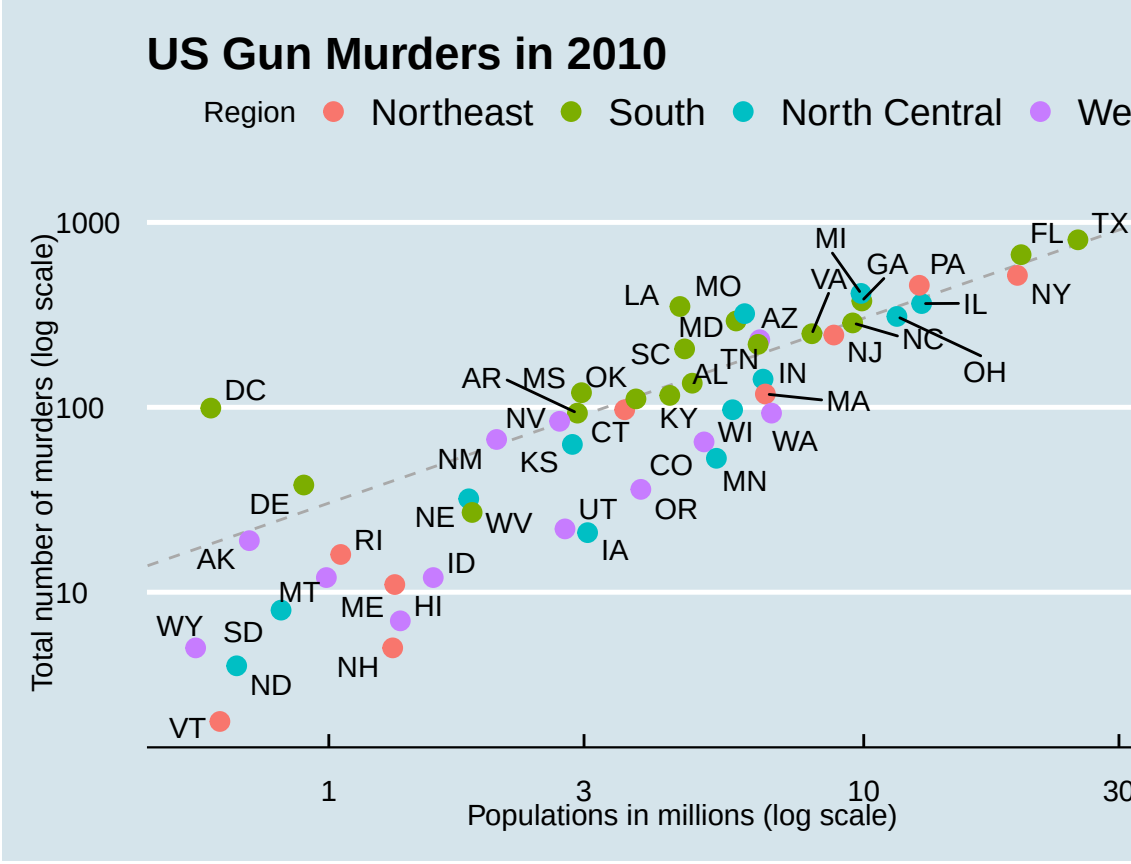
Add on packages for ggplot

```
library(ggthemes)
library(ggrepel)
```

```
p<- p+theme_economist()
p
```



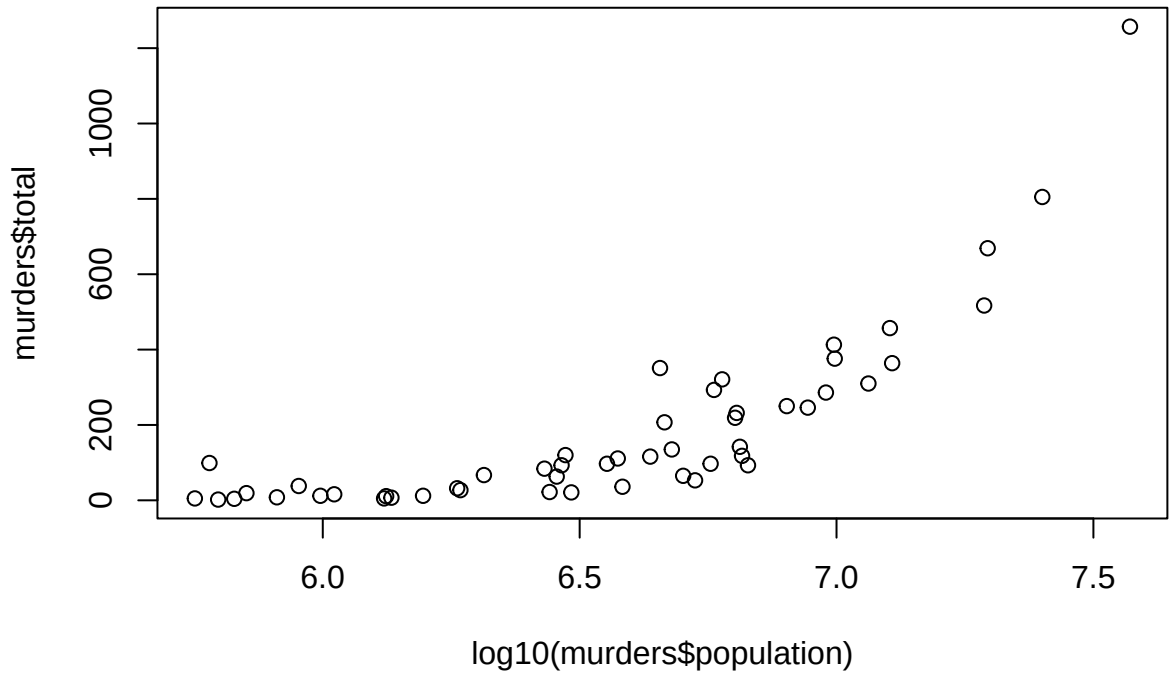

```
r <- murders %>%
  summarise(rate = sum(total)/sum(population)*10^6) %>%
  pull(rate)

p <- murders %>% ggplot(aes(population/10^6, total, label=abb))+
  geom_abline(intercept = log10(r), lty=2, color = "darkgrey") +
  geom_point(aes(col=region), size=3) +
  geom_text_repel() +
  scale_x_log10() +
  scale_y_log10() +
  xlab("Populations in millions (log scale)") +
  ylab("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010") +
  scale_color_discrete(name = "Region") +
  theme_economist()
p
```



Putting it all together

`qplot` was decprecated in ggplots2 3.4.0; therefore, my suggestion is to use `plot` per se.

```
plot(log10(murders$population), murders$total)
```



```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

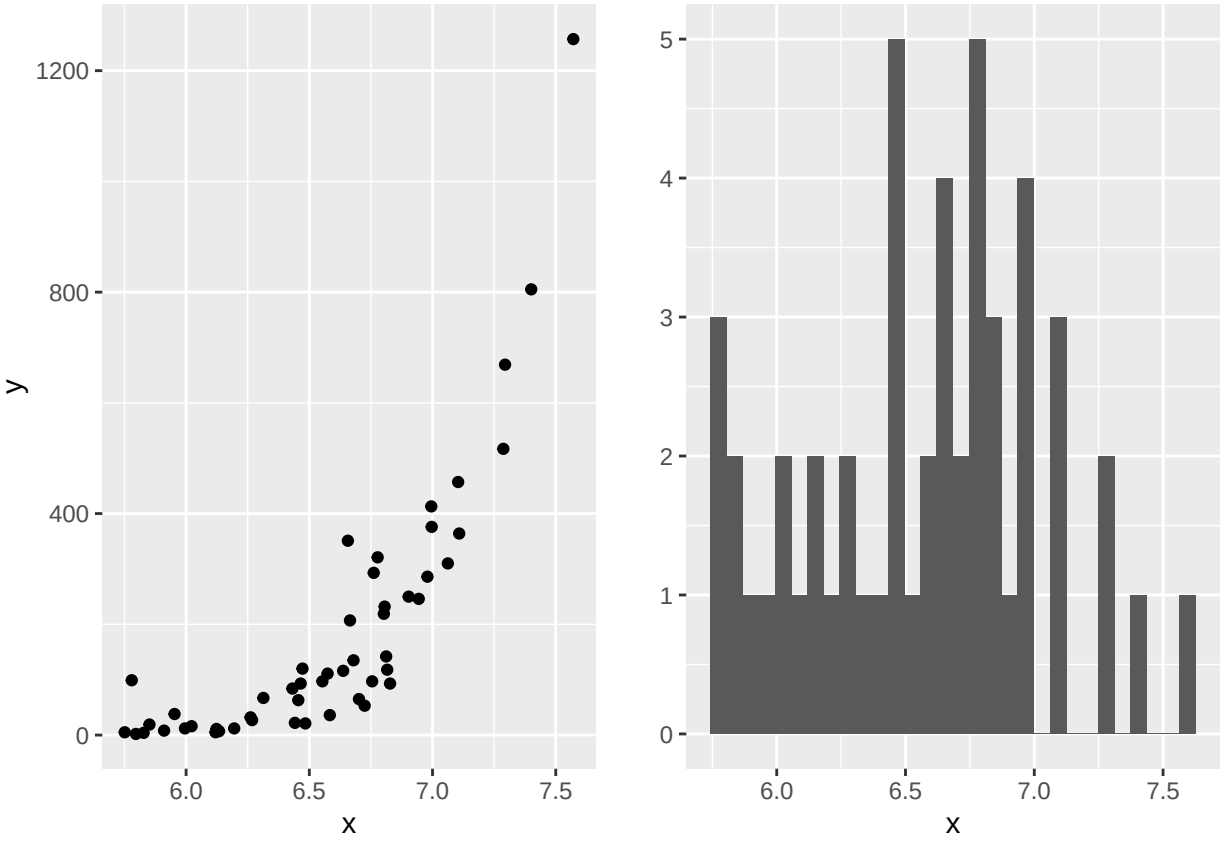
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
x <- log10(murders$population)
y <- murders$total
p1 <- qplot(x,y)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
p2 <- qplot(x)
grid.arrange(p1, p2, ncol=2)
```

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



Visualising data distributions

Frequency table

```
heights %>% count(sex) %>% mutate(proportion = n/sum(n))
```

```
##      sex   n proportion
```

```
## 1 Female 238  0.2266667
## 2   Male 812  0.7733333
```

```
index <- heights$sex == "Male"
x <- heights$height[index]

c(mean = mean(x), st_dev = sd(x))
```

```
##      mean    st_dev
## 69.314755  3.611024
```

Scaling

```
z <- scale(x)
mean(abs(z)<2)
```

```
## [1] 0.9495074
```

Quantile-quantile plots

```
pnorm(-1.96)
```

```
## [1] 0.0249979
```

qnorm

```
qnorm(0.975)
```

```
## [1] 1.959964
```

```
qnorm(.975, mean = 5, sd=2)
```

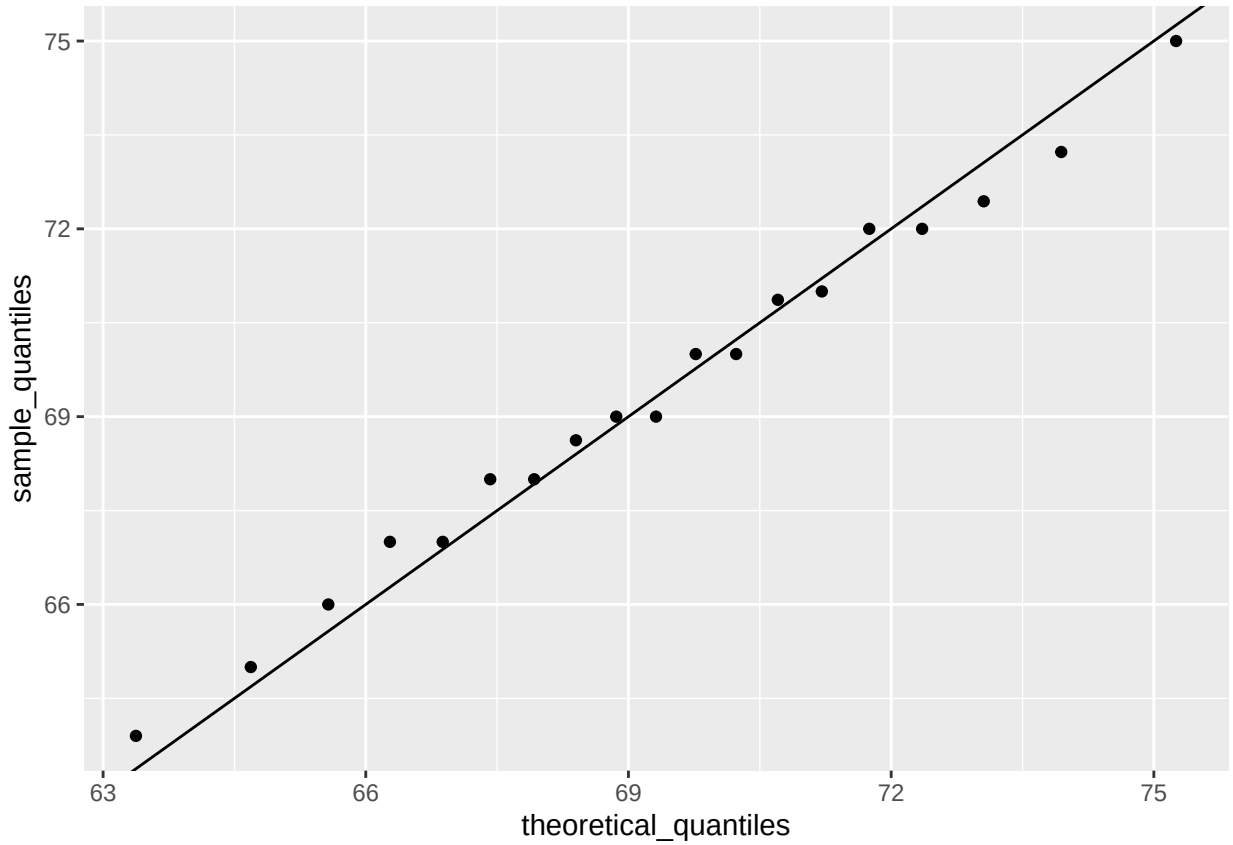
```
## [1] 8.919928
```

```
p <- seq(.05, .95, .05)
sample_quantiles <- quantile(x, p)
sample_quantiles
```

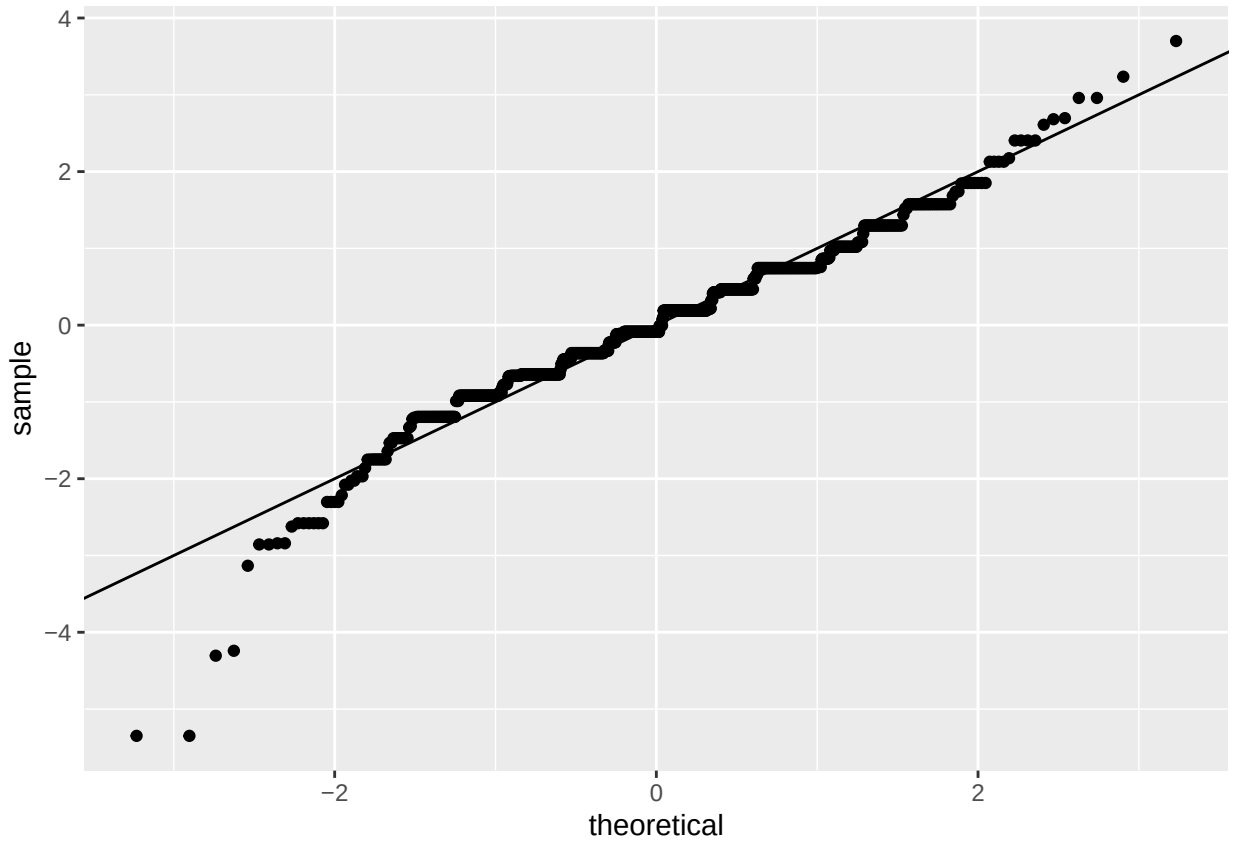
```
##       5%      10%      15%      20%      25%      30%      35%      40%
## 63.90079 65.00000 66.00000 67.00000 67.00000 68.00000 68.00000 68.62236
##      45%      50%      55%      60%      65%      70%      75%      80%
## 69.00000 69.00000 70.00000 70.00000 70.86614 71.00000 72.00000 72.00000
##      85%      90%      95%
## 72.44000 73.22751 75.00000
```

```
theoretical_quantiles <- qnorm(p, mean(x), sd(x))
```

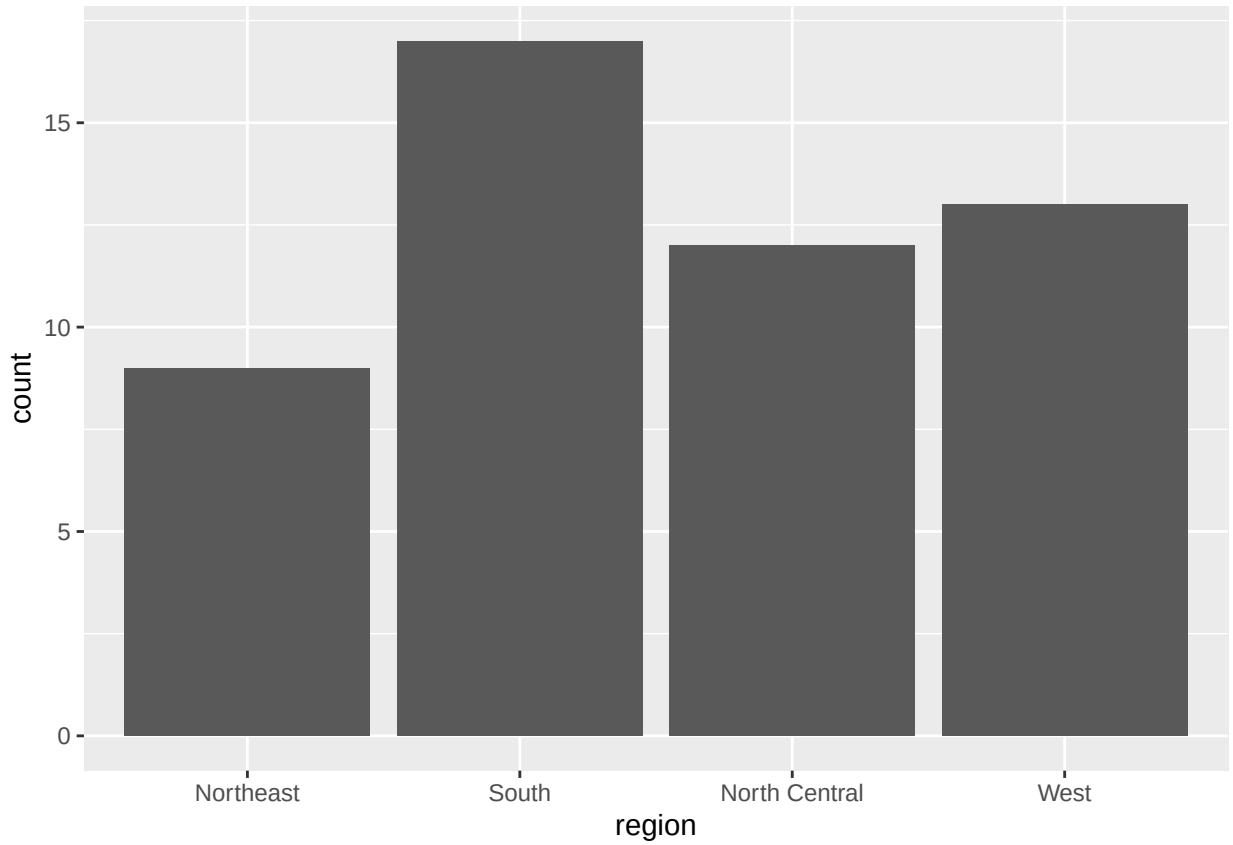
```
qplot(x=theoretical_quantiles, y=sample_quantiles) + geom_abline()
```




```
heights %>% filter(sex== "Male") %>% ggplot(aes(sample=scale(height))) +
                                        geom_qq() + geom_abline()
```



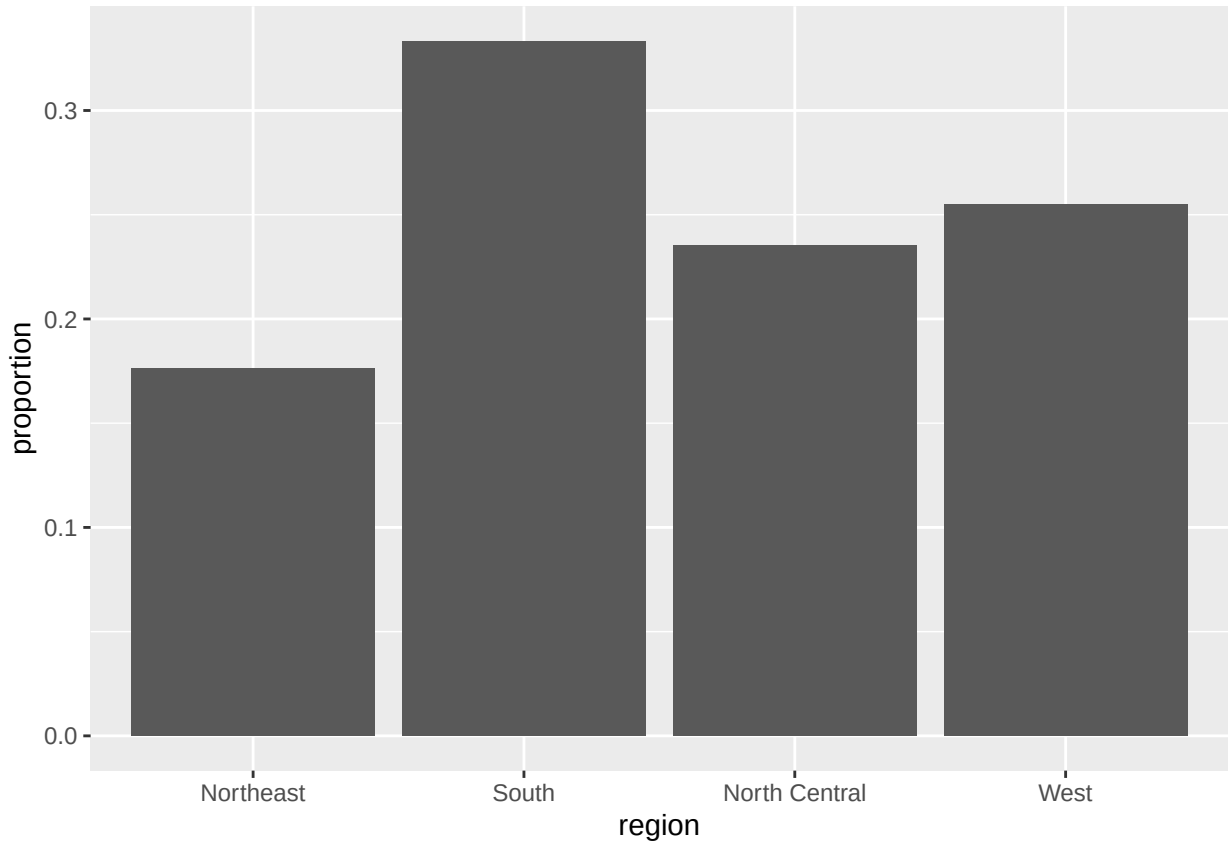

```
murders %>% ggplot(aes(region))+geom_bar()
```



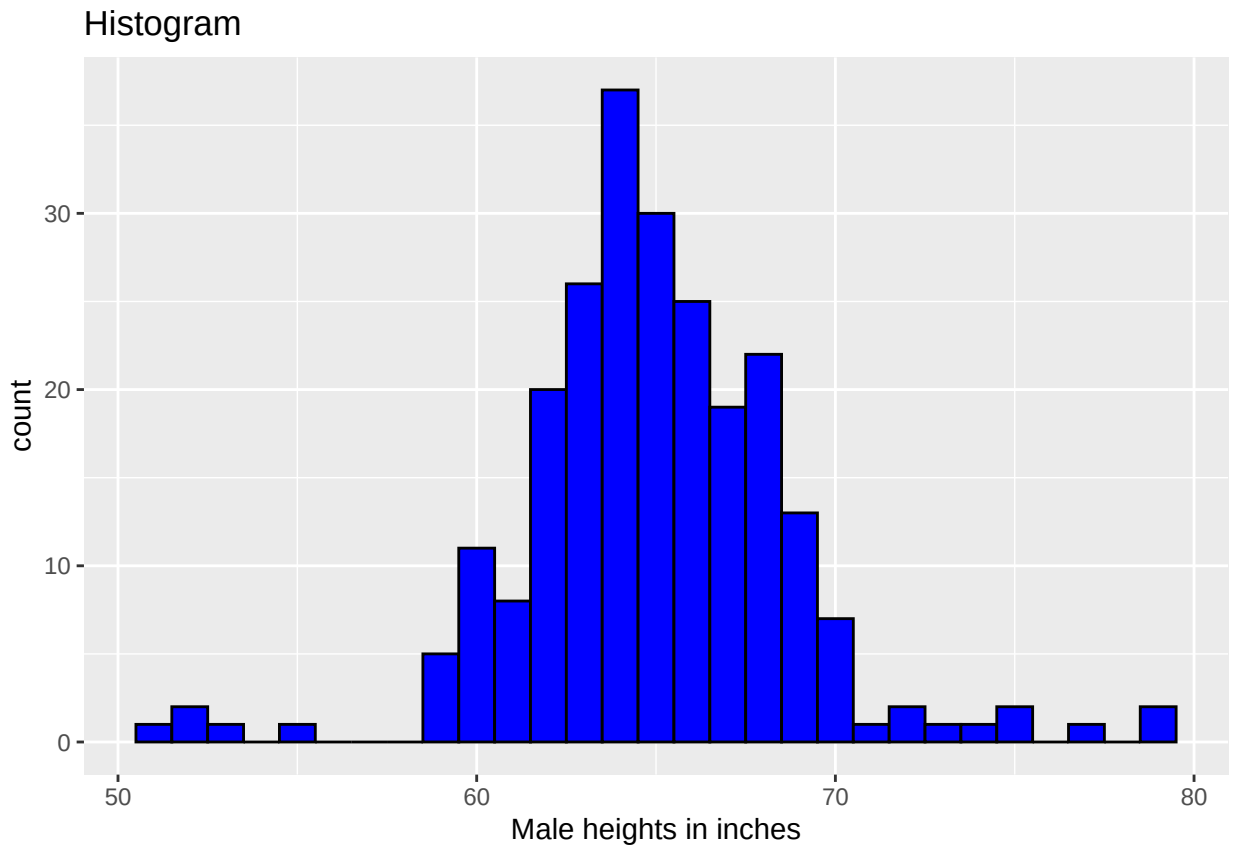

```
tab <- murders %>% count(region) %>% mutate(proportion = n/sum(n))
tab
```

```
##          region  n proportion
## 1     Northeast  9  0.1764706
## 2         South 17  0.3333333
## 3 North Central 12  0.2352941
## 4          West 13  0.2549020
```

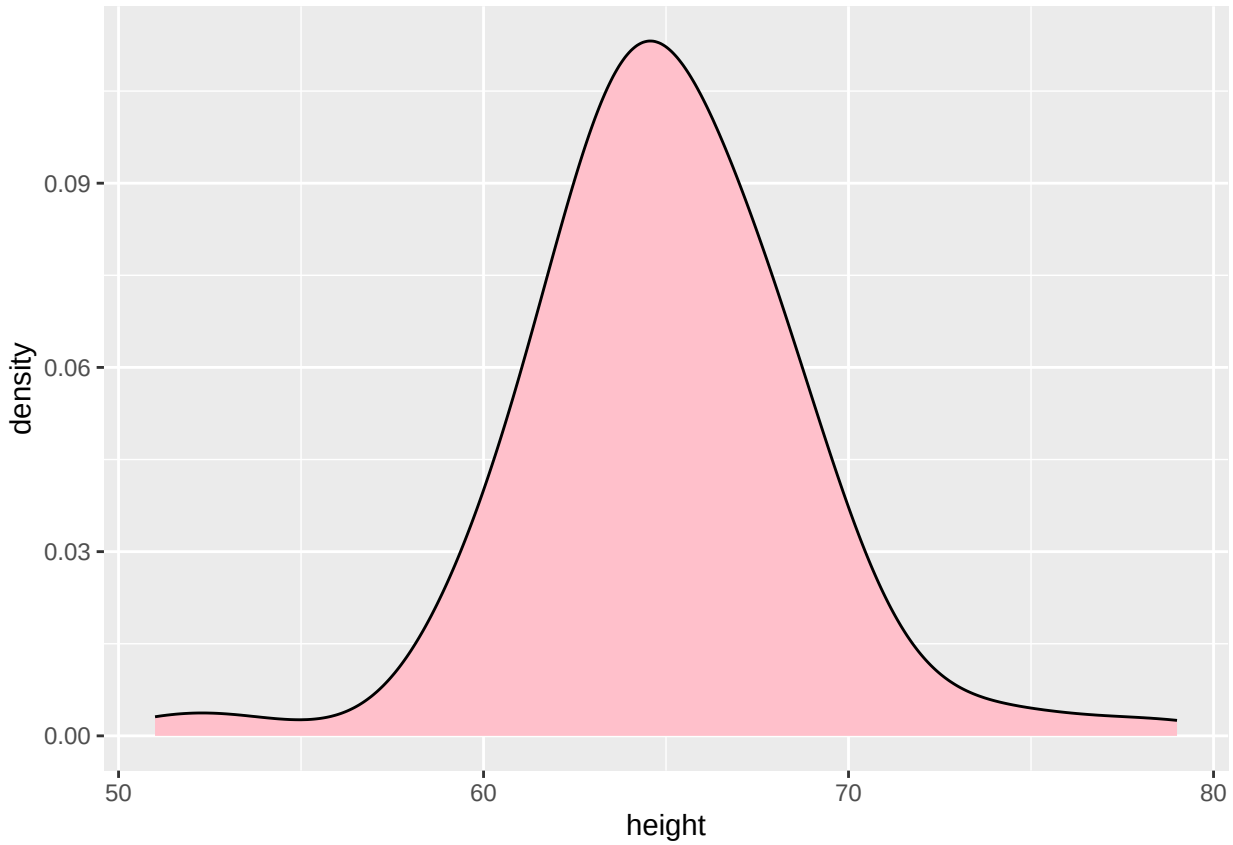
```
tab %>% ggplot(aes(region, proportion))+geom_bar(stat = "identity")
```



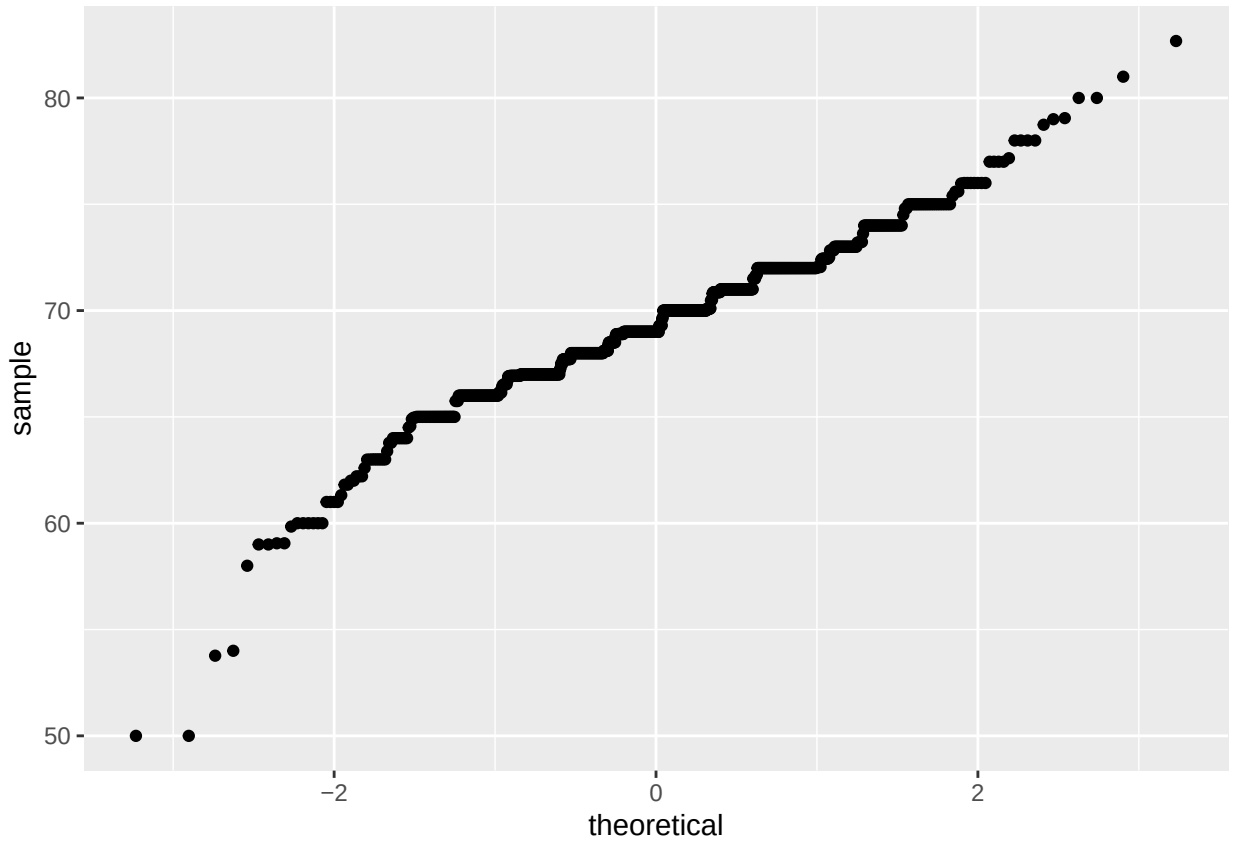

```
heights %>% filter(sex == "Female") %>% ggplot(aes(height)) +
  geom_histogram(binwidth = 1, fill="blue", col = "black")+
  xlab("Male heights in inches")+
  ggtitle("Histogram")
```



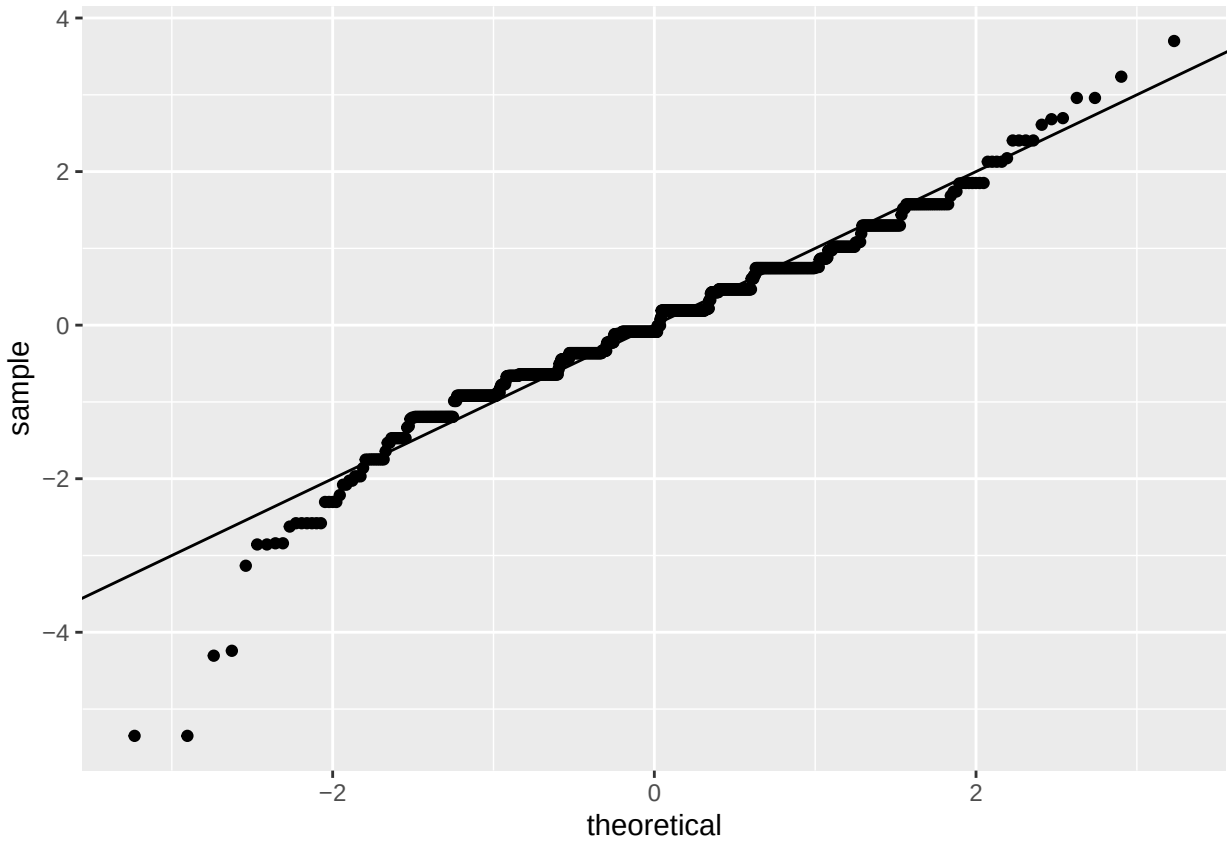

```
heights %>% filter(sex == "Female") %>%
  ggplot(aes(height)) + geom_density(fill="pink", adjust = 2) #adjust the smoothness
```




```
heights %>% filter(sex == "Male") %>%
  ggplot(aes(sample=height)) + geom_qq()
```




```
heights %>% filter(sex == "Male") %>% ggplot(aes(sample=scale(height))) +
  geom_qq()+
  geom_abline()
```



Data visualisation in practice

```
data(gapminder)
gapminder %>% as_tibble()
```

```
## # A tibble: 10,545 x 9
##    country   year infant_mortality life_expectancy fertility population      gdp
##    <fct>    <int>            <dbl>           <dbl>     <dbl>      <dbl>    <dbl>
##  1 Albania   1960            115.             62.9      6.19    1636054 NA
##  2 Algeria   1960            148.             47.5      7.65   11124892  1.38e10
##  3 Angola    1960            208              36.0      7.32    5270844 NA
##  4 Antigua~  1960             NA              63.0      4.43      54681 NA
##  5 Argenti~  1960             59.9            65.4      3.11   20619075  1.08e11
##  6 Armenia   1960             NA              66.9      4.55    1867396 NA
##  7 Aruba     1960             NA              65.7      4.82      54208 NA
##  8 Austral~  1960             20.3            70.9      3.45   10292328  9.67e10
##  9 Austria   1960             37.3            68.8      2.7     7065525  5.24e10
## 10 Azerbai~  1960             NA              61.3      5.57    3897889 NA
## # i 10,535 more rows
## # i 2 more variables: continent <fct>, region <fct>
```

Comparison of child mortality in five countries

1. Sri Lanka versus Turkiye
2. Poland versus South Korea
3. Pakistan versus Vietnam
4. Malaysia versus Russia
5. Thailand versus South Africa

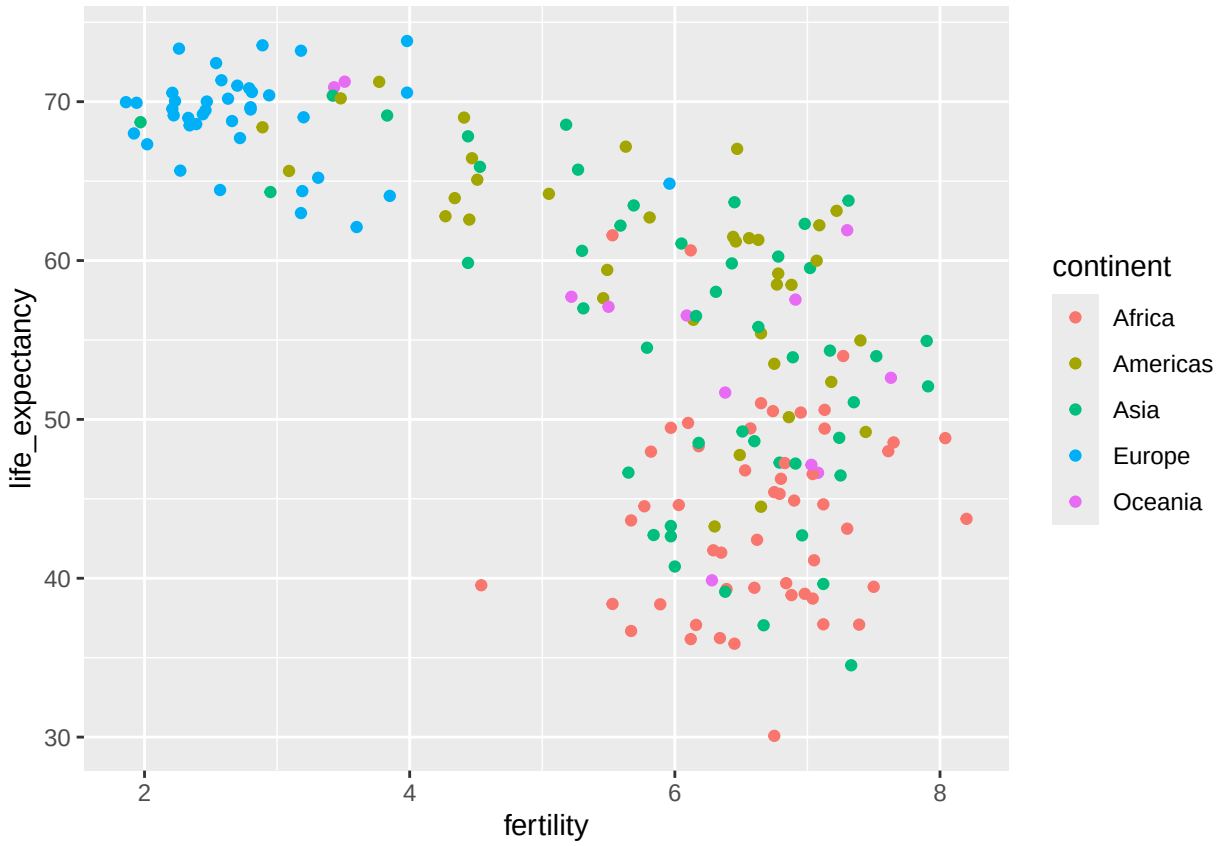
```r
gapminder %>% filter(year == 2015 & country %in% c("Sri Lanka", "Turkey",
                                                   "Poland", "South Korea",
                                                   "Pakistan", "Vietnam",
                                                   "Malaysia", "Russia",
                                                   "Thailand", "South Africa")) %>%
  select(country, infant_mortality)
```

```
##          country infant_mortality
## 1   South Korea              2.9
## 2      Malaysia              6.0
## 3      Pakistan             65.8
## 4        Poland              4.5
## 5        Russia              8.2
## 6  South Africa             33.6
## 7     Sri Lanka              8.4
## 8      Thailand             10.5
## 9        Turkey             11.6
## 10      Vietnam             17.3
```

```r
gapminder %>% filter(year == 2015 & country %in% c("Poland", "South Korea")) %>%
  select(country, infant_mortality)
```

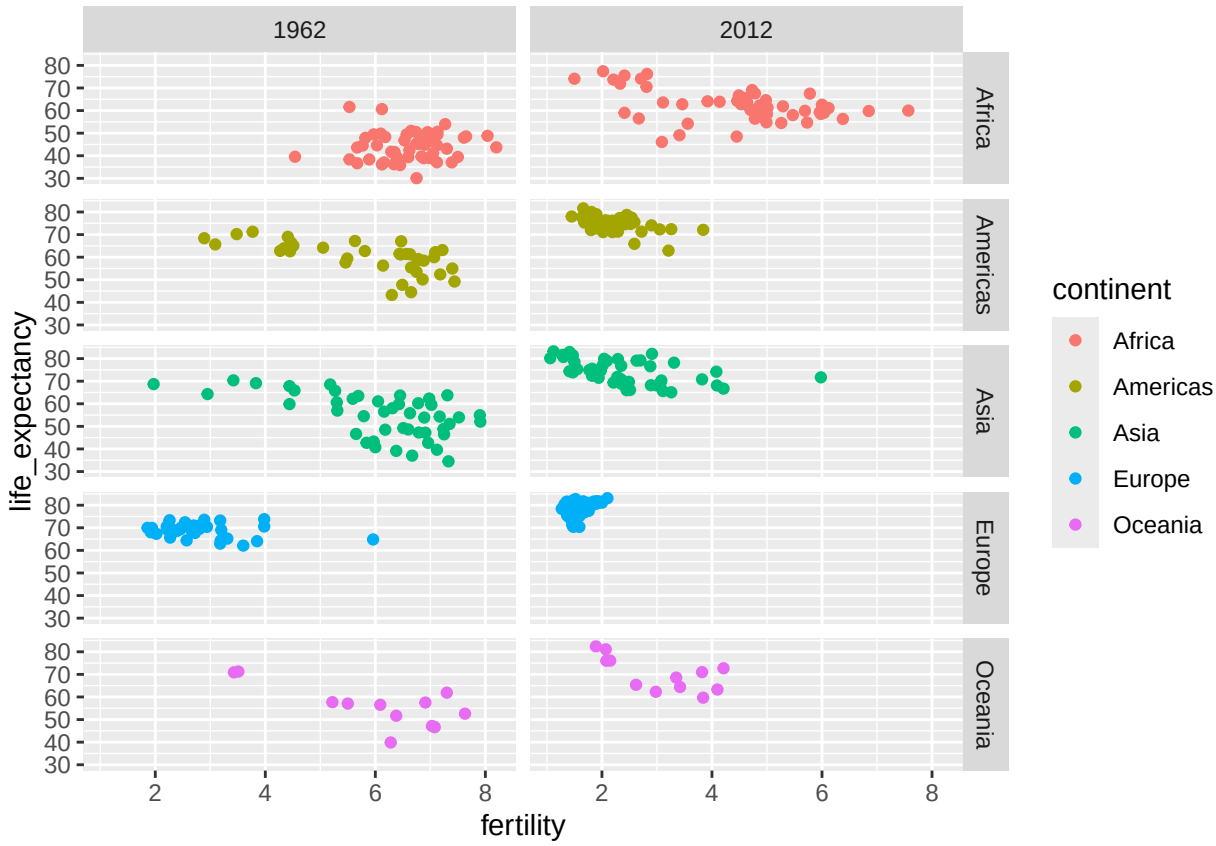
```
##       country infant_mortality
## 1 South Korea              2.9
## 2      Poland              4.5
```

```r
filter(gapminder, year==1962) %>%
  ggplot(aes(fertility, life_expectancy, color=continent))+
  geom_point()
```

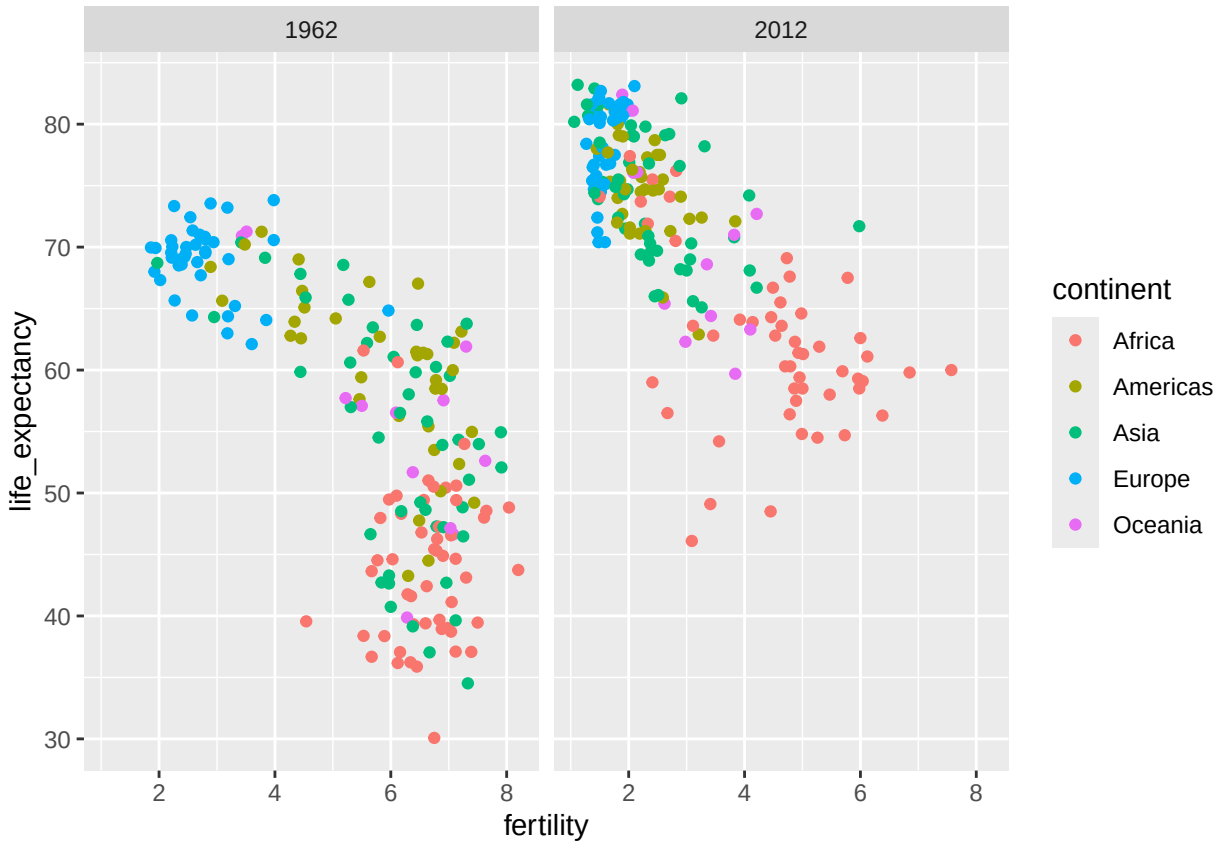


Faceting

```
filter(gapminder, year %in% c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, color=continent))+
  geom_point()+
  facet_grid(continent~year)
```



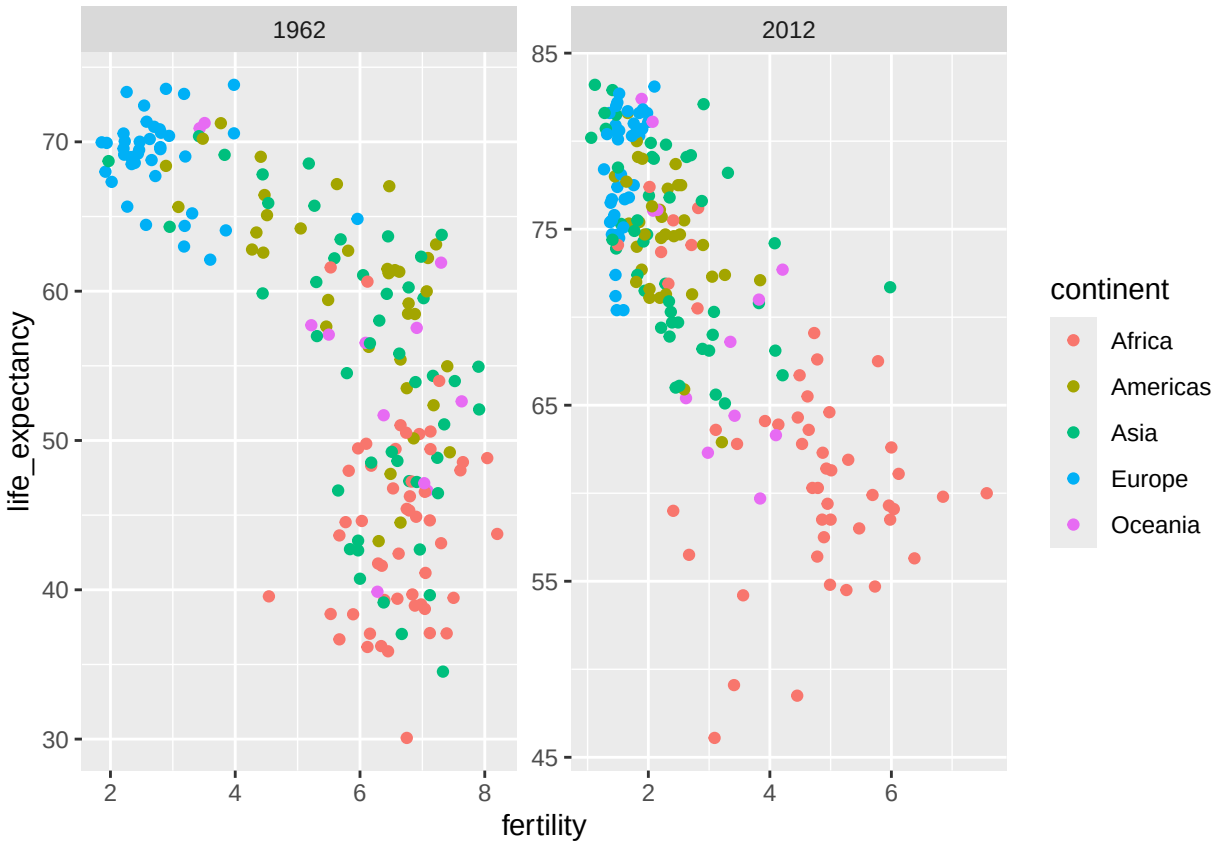

```
filter(gapminder, year %in% c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, color=continent))+
  geom_point()+
  facet_grid(.~year)
```



facet_wrap

```
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot(aes(fertility, life_expectancy, col=continent)) +
  geom_point()+
  facet_wrap(~year)
```

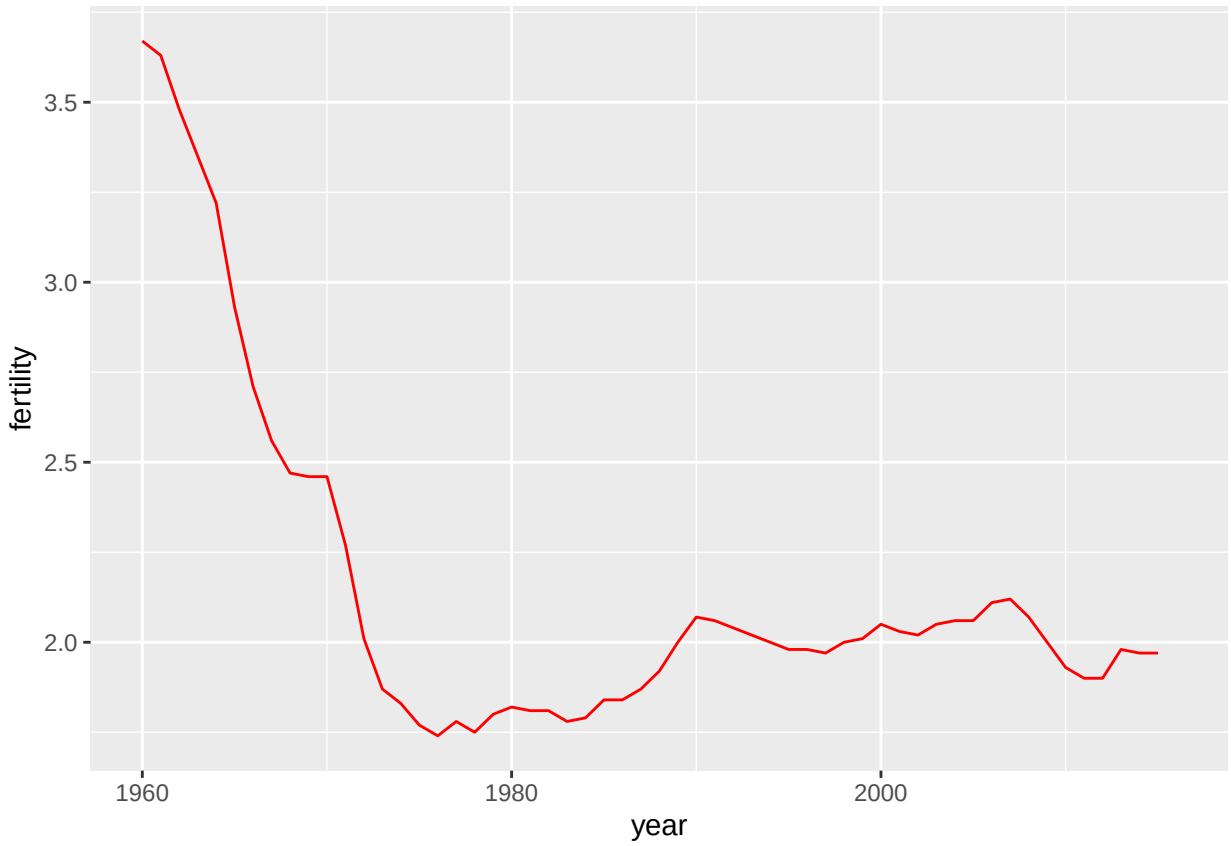

```
gapminder %>% filter(year %in% c(1962, 2012))%>%
  ggplot(aes(fertility, life_expectancy, col=continent)) +
  geom_point()+
  facet_wrap(.~year, scales = "free")
```



Time series plots

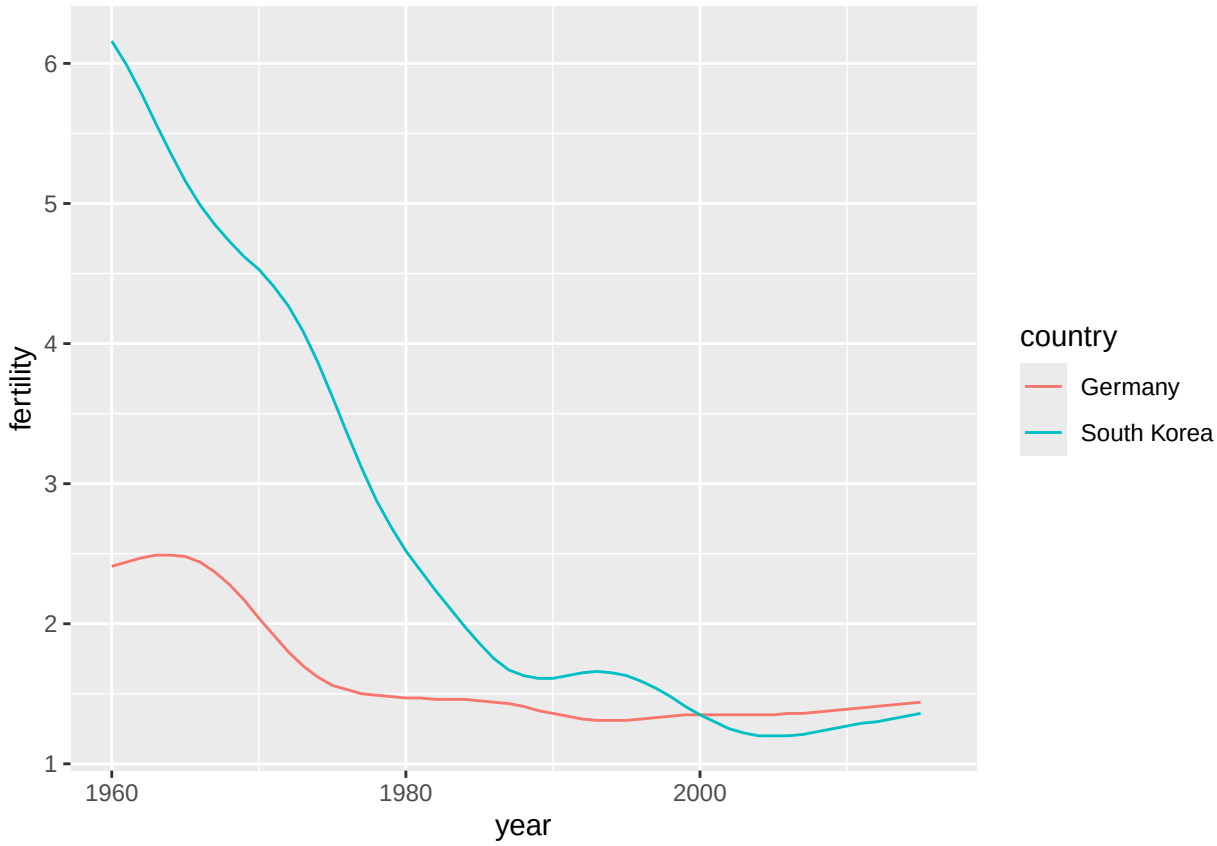
```
gapminder %>%
  filter(country == "United States") %>%
  ggplot(aes(year, fertility))+
  geom_line(colour="red")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



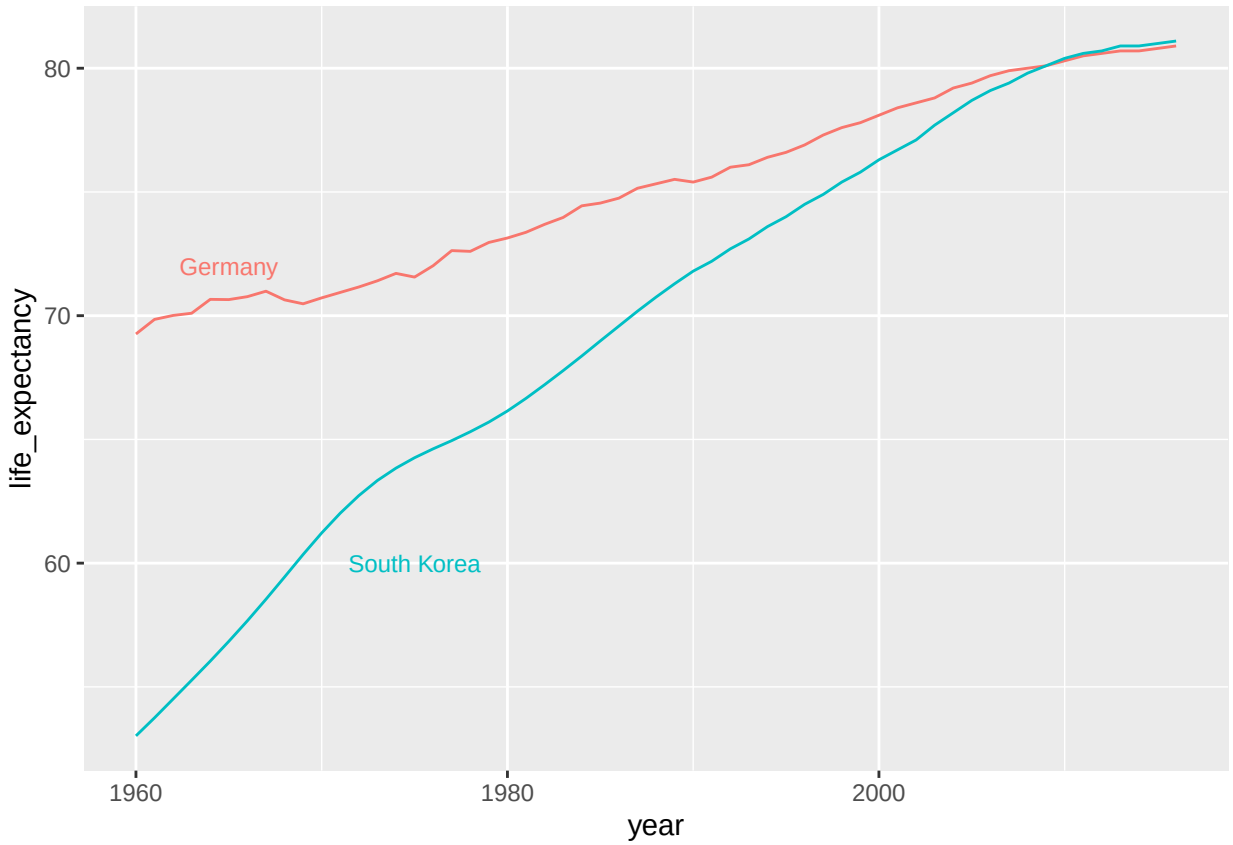

```
countries <- c("South Korea", "Germany")

gapminder %>% filter(country %in% countries) %>%
  ggplot(aes(year, fertility, col = country))+
  geom_line()
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

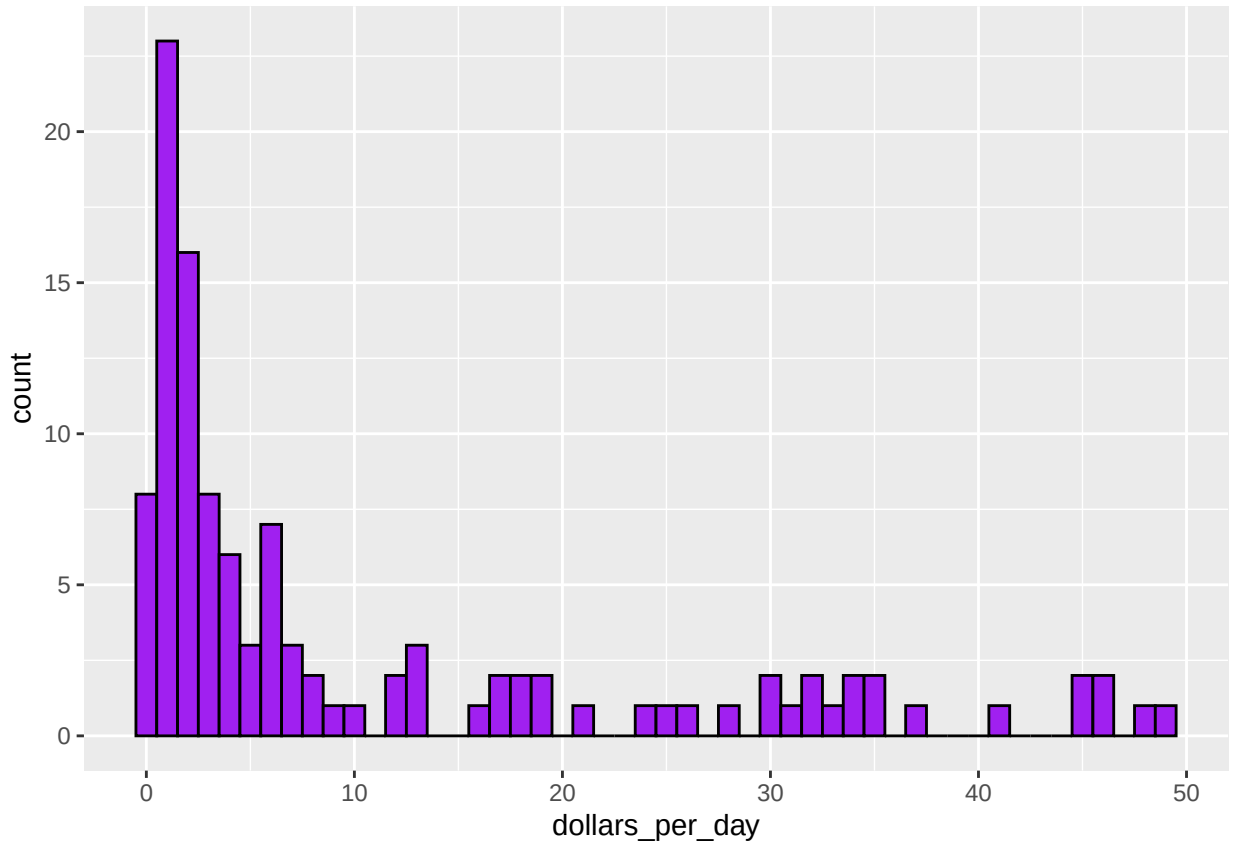


Labels instead of legends

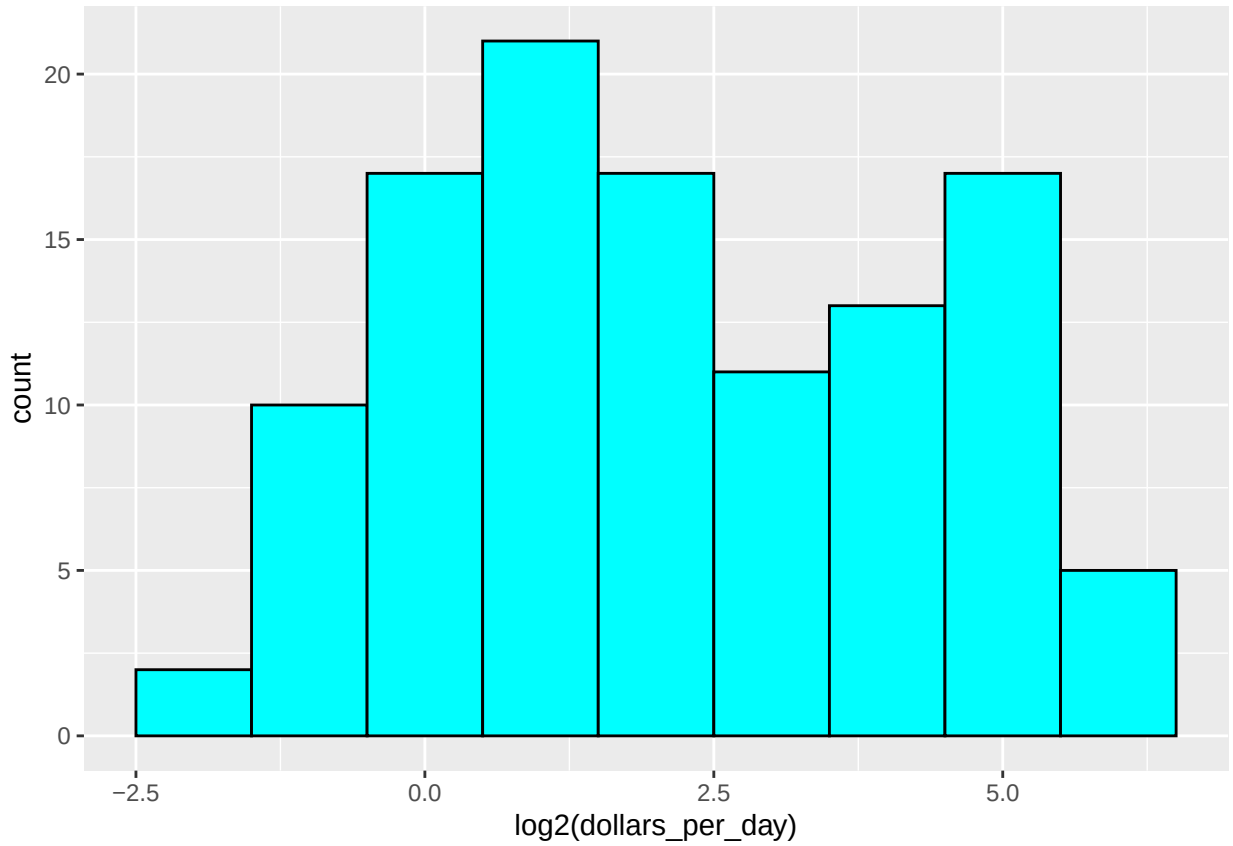
```
labels <- data.frame(country=countries, x = c(1975, 1965), y = c(60, 72))

gapminder %>%
  filter(country %in% countries) %>%
  ggplot(aes(year, life_expectancy, col=country))+
  geom_line()+
  geom_text(data = labels, aes(x,y, label=country), size=3)+
  theme(legend.position = "none")
```




```
gapminder <- gapminder %>% mutate(dollars_per_day = gdp/population/365)
past_year <- 1970
gapminder %>% filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(dollars_per_day))+
  geom_histogram(binwidth = 1, fill="purple", color = "black")
```



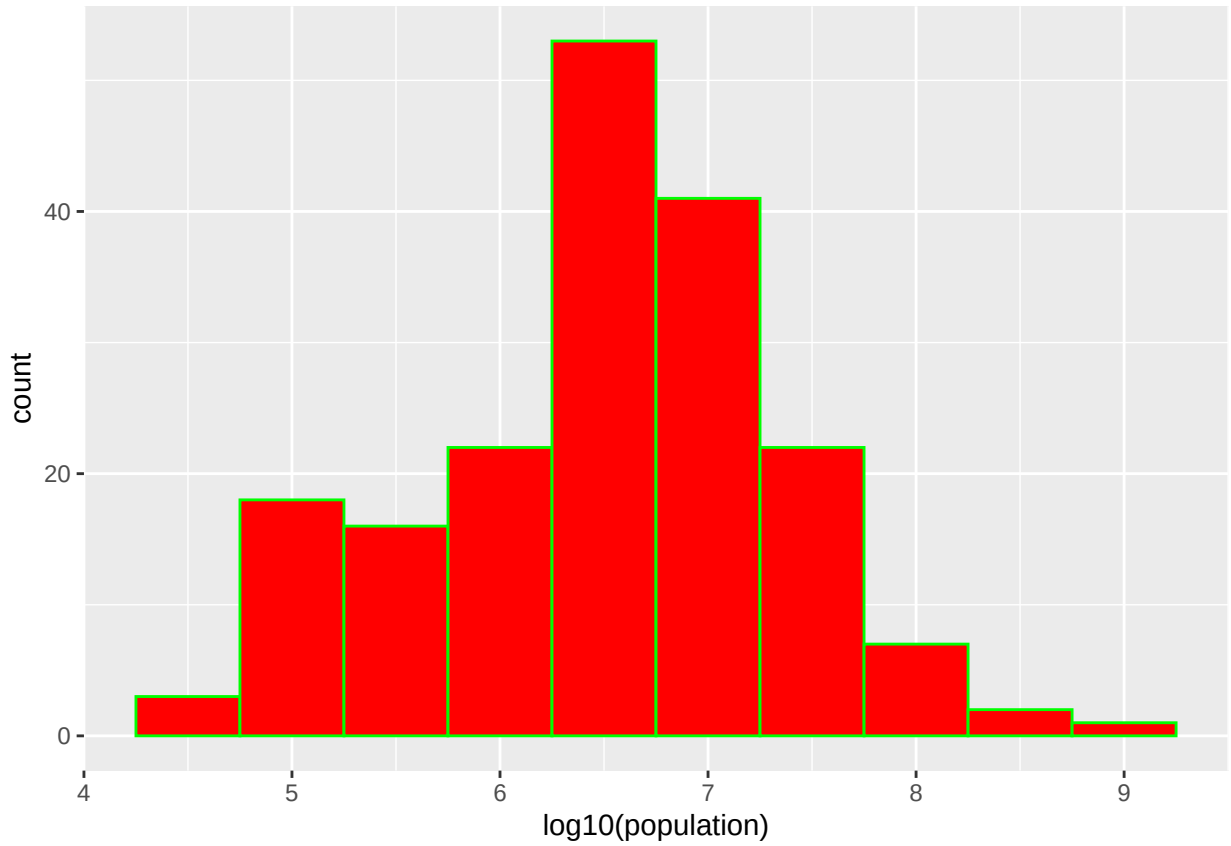

```
gapminder <- gapminder %>% mutate(dollars_per_day = gdp/population/365)
past_year <- 1970
gapminder %>% filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(log2(dollars_per_day)))+
  geom_histogram(binwidth = 1, fill="cyan", color = "black")
```



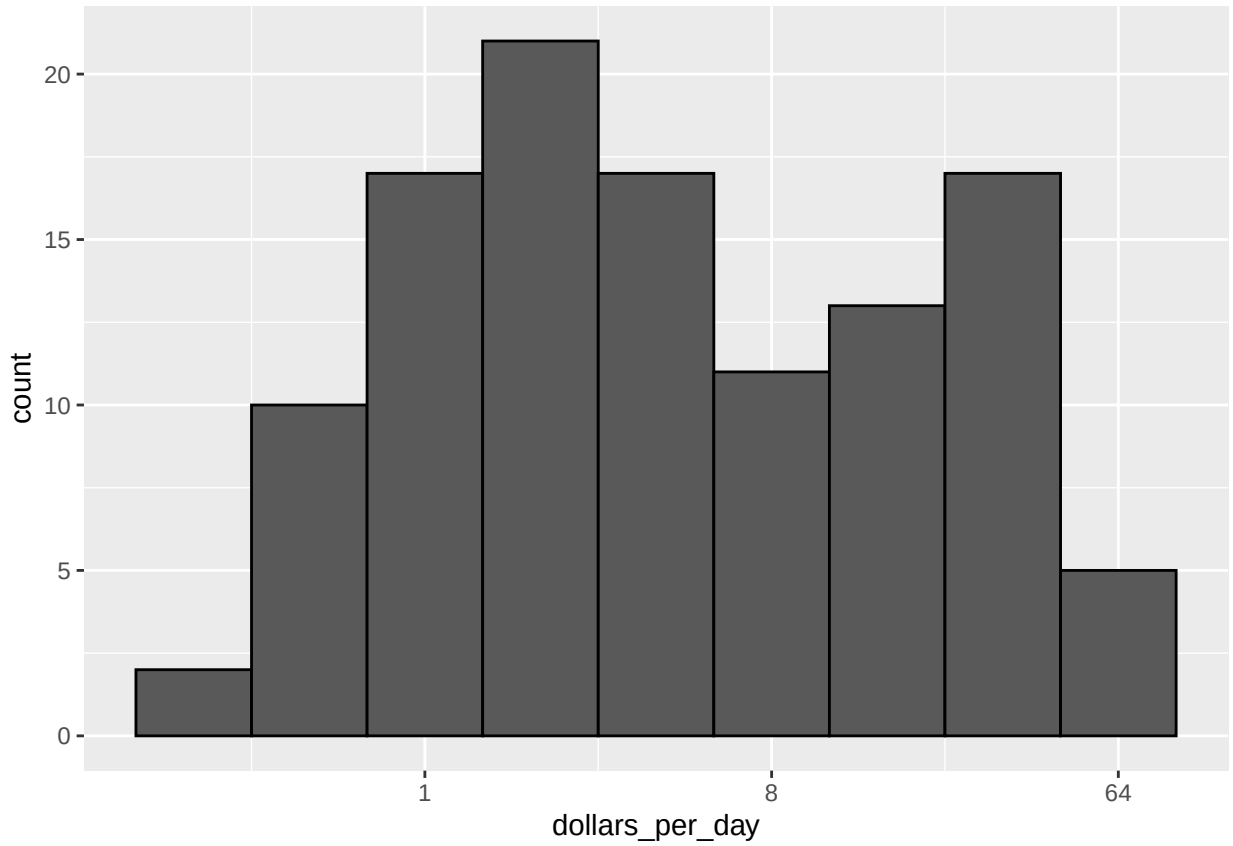

```
filter(gapminder, year == past_year) %>%
  summarise(min = min(population), max = max(population))
```

```
##     min       max
## 1 46075 808510713
```

```
gapminder %>%
  filter(year == past_year) %>%
  ggplot(aes(log10(population)))+
  geom_histogram(binwidth = 0.5, fill="red" ,color = "green")
```

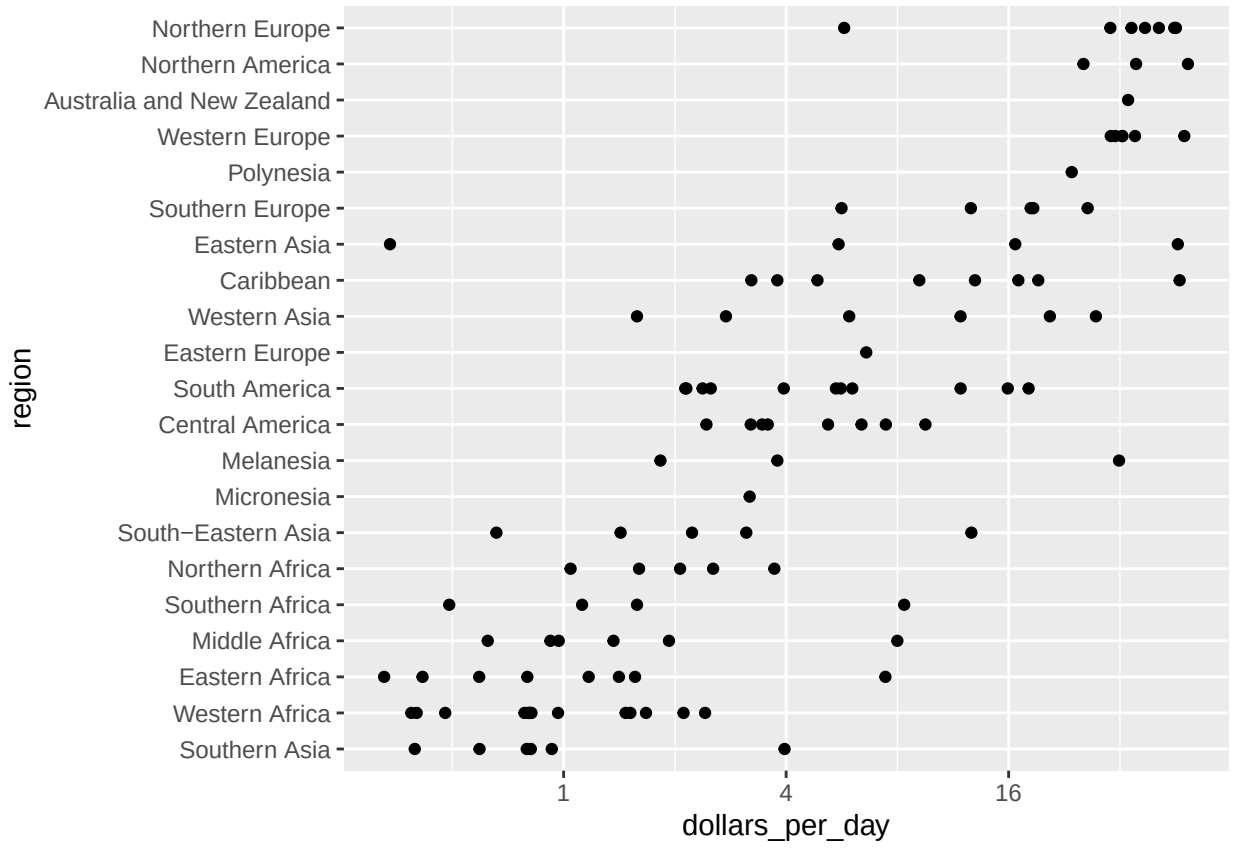



```
gapminder <- gapminder %>% mutate(dollars_per_day = gdp/population/365)
past_year <- 1970
gapminder %>% filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(dollars_per_day))+
  geom_histogram(binwidth = 1, color = "black") +
  scale_x_continuous(trans = "log2")
```




```
gapminder <- gapminder %>%
  mutate(group = case_when(
    region %in% c("Western Europe", "Northern Europe","Southern Europe",
                  "Northern America","Australia and New Zealand") ~ "West",
    region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",
    region %in% c("Caribbean", "Central America",
                  "South America") ~ "Latin America",
    continent == "Africa" & region != "Northern Africa" ~ "Sub-Saharan",
    TRUE ~ "Others"))
```

```
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  mutate(region = reorder(region, dollars_per_day, FUN = median)) %>%
  ggplot(aes(dollars_per_day, region))+
  geom_point()+
  scale_x_continuous(trans="log2")
```


Northern Europe
Northern America
Australia and New Zealand
Western Europe
Polynesia
Southern Europe
Eastern Asia
Caribbean
Western Asia
Eastern Europe
South America
Central America
Melanesia
Micronesia
South−Eastern Asia
Northern Africa
Southern Africa
Middle Africa
Eastern Africa
Western Africa
Southern Asia
region
1
4
16
dollars_per_day