Stat5405-HW1 Solution

Pooja Raj Lakshmi

2025-09-06

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(purrr)
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library(stringr)
library(janitor)
```

```
Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
library(readr)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats 1.0.0     v tibble  3.3.0
v ggplot2 3.5.2     v tidyr   1.3.1

-- 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 becor
```

Question 1. Use the trees data in the R package *datasets*.

```r
#loading trees dataset
data(trees)
```

Data Exploration

Dataframe has 31 observations (rows) and 3 numeric variables (columns), i.e. ***Girth***, ***Height*** and ***Volume***.

A quick look at the dataset (`summary()`) shows that:

- **Girth** ranges from about 8 to 21 inches (approximately, with a median close to 12.9.
- **Height** varies between 63 and 87 ft, with most trees clustering around the 70s.
- **Volume** spans from 10 to 77 cubic feet, showing a much wider spread compared to the other variables.

There are no missing values, and all three variables are continuous. Correlation analysis suggests a strong positive relationship between **Girth** and **Volume**, as expected from tree growth patterns, while Height is less strongly correlated with the other two.

This initial exploration confirms that the dataset is clean.

```r
?trees
```

```r
#checking the structure of the dataframe
str(trees)
```

```
'data.frame':   31 obs. of  3 variables:
 $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
 $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```r
#checking head of trees dataset
head(trees)
```

```
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

```r
#checking tail of dataset
tail(trees)
```

```
   Girth Height Volume
26  17.3     81   55.4
27  17.5     82   55.7
28  17.9     80   58.3
29  18.0     80   51.5
30  18.0     80   51.0
31  20.6     87   77.0
```

```r
summary(trees)
```

```
     Girth           Height       Volume     
 Min.   : 8.30   Min.   :63   Min.   :10.20  
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40  
 Median :12.90   Median :76   Median :24.20  
 Mean   :13.25   Mean   :76   Mean   :30.17  
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30  
 Max.   :20.60   Max.   :87   Max.   :77.00  
```

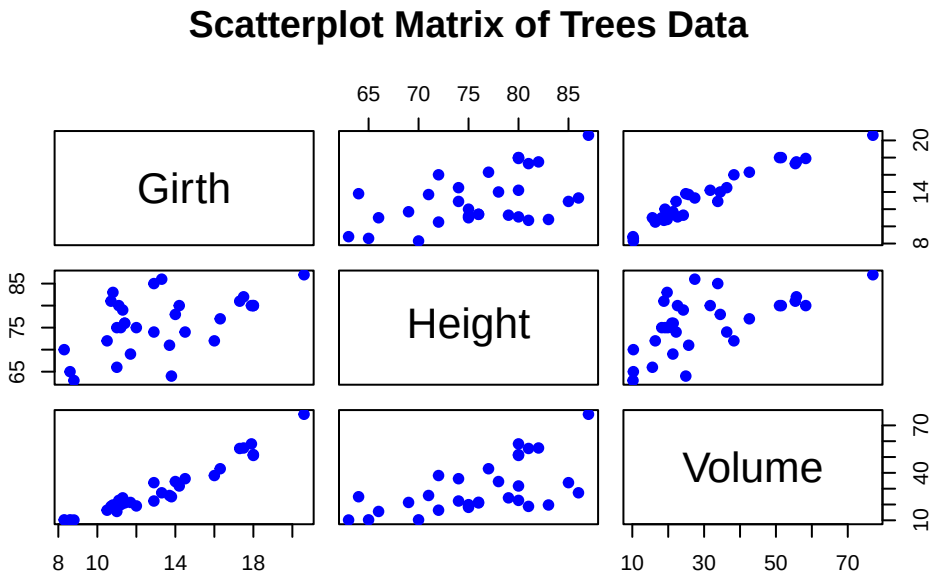
```r
#checking for missing values
colSums(is.na(trees))
```

```
 Girth Height Volume
     0      0      0
```

```r
#correlation coefficient of numerical variables
cor(trees)
```

```
            Girth    Height    Volume
Girth  1.0000000 0.5192801 0.9671194
Height 0.5192801 1.0000000 0.5982497
Volume 0.9671194 0.5982497 1.0000000
```

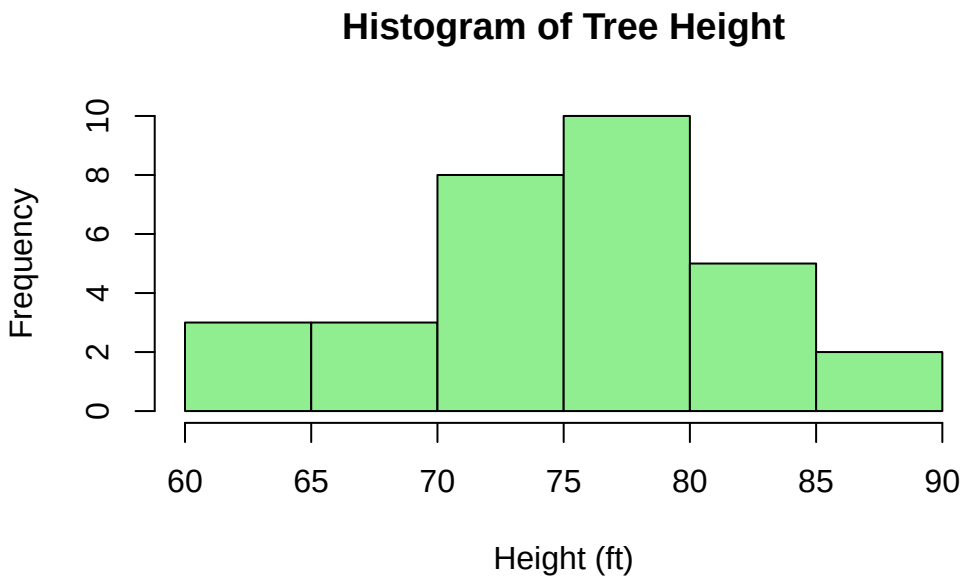
```r
#scatterplot matrix of all the numerical variables
pairs(trees,
      main = "Scatterplot Matrix of Trees Data",
      pch = 19, col = "blue")
```



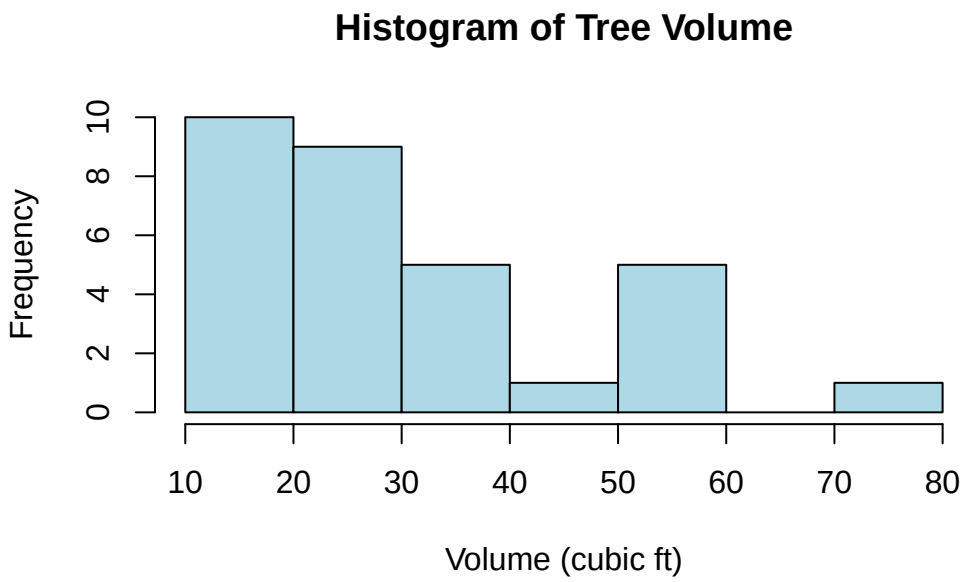
(a) Construct histograms of ***Height***, ***Volume*** and ***Girth***.

Since the Dataset is very small, using 6 Bins (**Sturges' rule**: `k = 1 + log2(n)` → For 31 trees, 6 bins). Also, using frequency and not relative frequency (due to small size of dataset).

```r
# Histogram of Height
hist(trees$Height,
     breaks = 6, #since sample size is small, using 6 bins
     freq = TRUE, #y-axis shows the count of observations in each bin
     right = TRUE, #bins are right closed
     main = "Histogram of Tree Height",
     xlab = "Height (ft)",
     col = "lightgreen",
     border = "black")
```



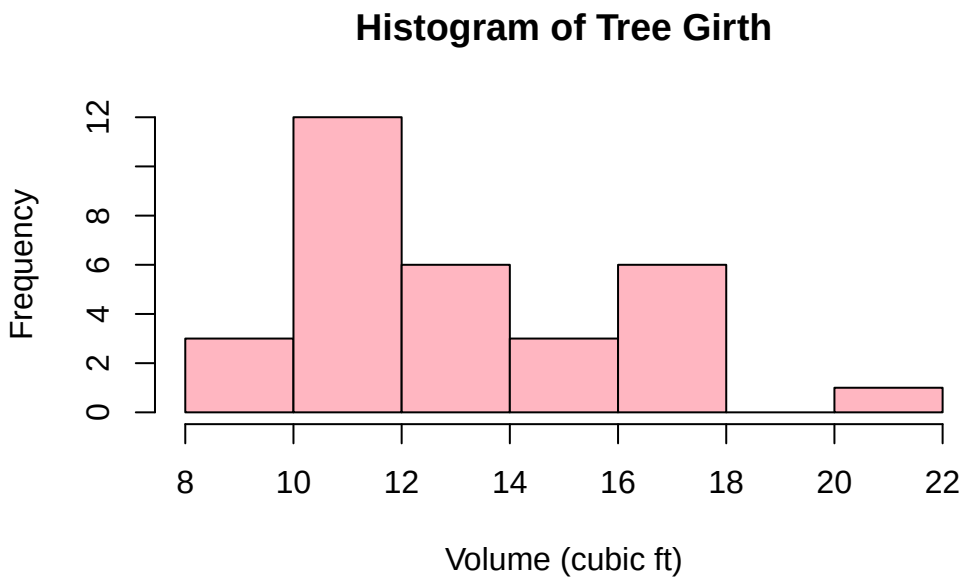

```r
# Histogram of Volume
hist(trees$Volume,
     breaks = 6, #since sample size is small, using 6 Bins
     freq = TRUE,
     right = TRUE, #bins are right closed
     main = "Histogram of Tree Volume",
     xlab = "Volume (cubic ft)",
     col = "lightblue",
     border = "black")
```




```
#actual values between range 60-70 in order to test the correctness of histogram
trees$Volume[trees$Volume >= 60 & trees$Volume <= 70]
```

```
numeric(0)
```

```
# Histogram of Girth
hist(trees$Girth,
     breaks = 6, #since sample size is small, using 6 Bins
     freq = TRUE,
     right = TRUE, #bins are right-closed
     main = "Histogram of Tree Girth",
     xlab = "Volume (cubic ft)",
     col = "lightpink",
     border = "black")
```




```r
#actual values between range 60-70 in order to test the correctness of histogram
trees$Girth[trees$Girth >= 18 & trees$Girth <= 20]
```

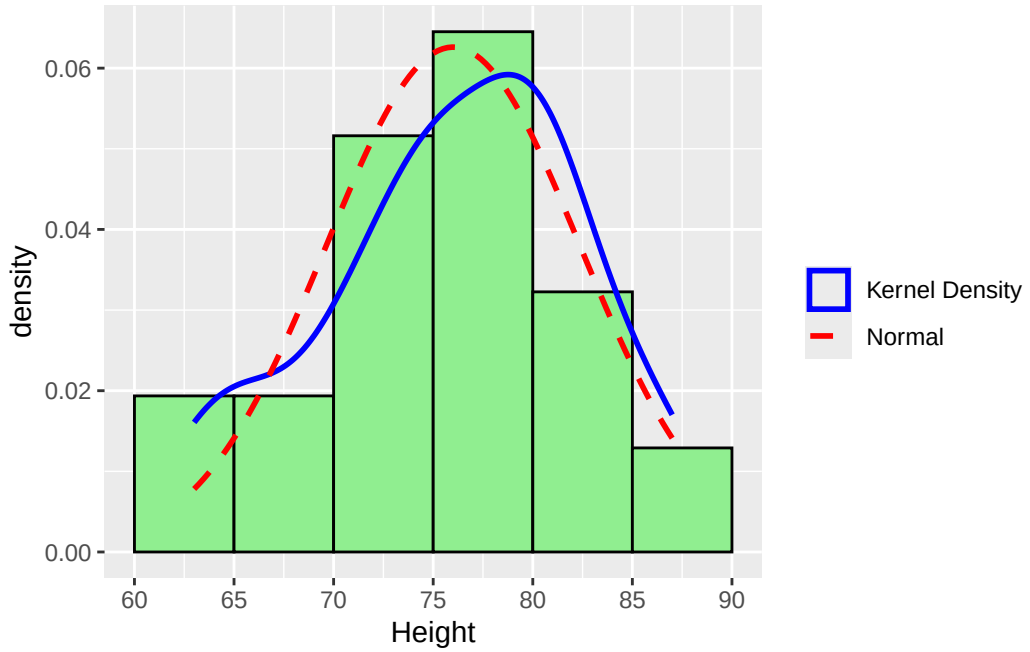
```
[1] 18 18
```

Comment:

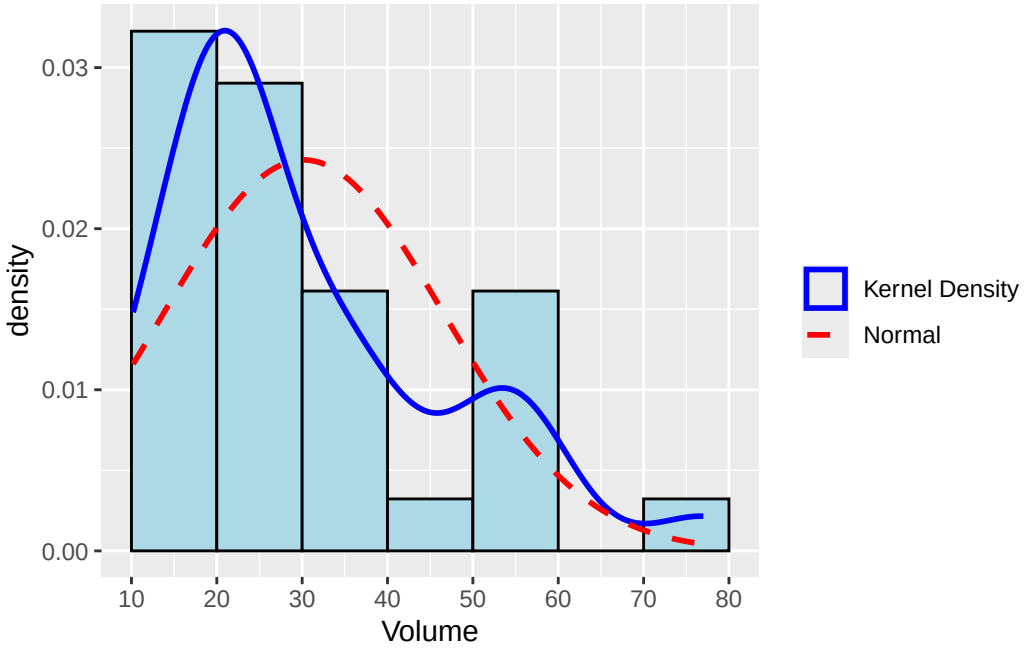
1. **Height** (ft) is roughly uni-modal. Most of the trees cluster in the **70–80 ft** range.
2. **Volume** (cubic ft) is strongly **right skew**—many small volumes and a long upper tail (e.g., one near 77). Very few trees between **60–70**, which matches the raw check returning none.
3. **Girth** (in) is **mildly right skew** with most of the values in **low teens**. Both 18-inch trees fall into (16,18], leaving (18,20] empty; one large tree (~20.6 in) sits in (20,22].
4. The histograms show **Volume** is **right-skewed**, while **Girth** is **concentrated** in the **low teens** with a mild right tail. A scatterplot of Girth vs .Volume (also Height) and the correlation `r 0.97r\approx0.97r 0.97` indicate a **very strong positive relationship between Volume and Girth**.

b. Add a kernel density plot to each histogram in (a), and overlay the histogram with a normal distribution.

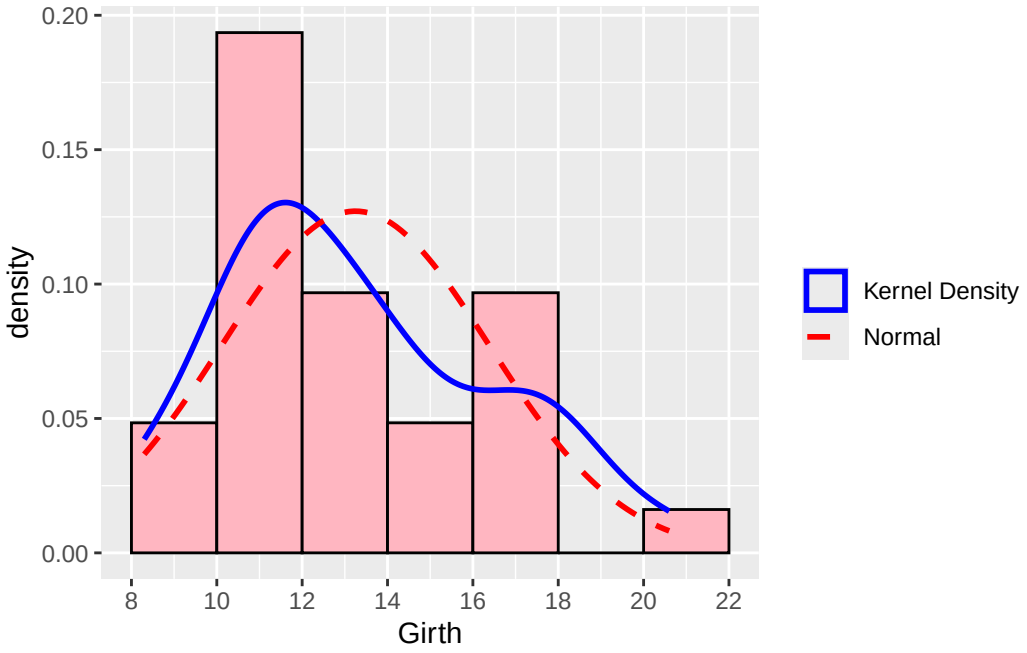
```r
library(ggplot2)
#Height :- kernel density plot of overlay with normal distribution
ggplot(trees, aes(x = Height)) +
  stat_bin(breaks = seq(60, 90, by = 5),
           aes(y=after_stat(density)),
           colour = "black",fill = "lightgreen",closed = "right") +
           scale_x_continuous(breaks = seq(60, 90, by = 5)) +
  geom_density(aes(color="Kernel Density"),linewidth=1) + # Adding kernel density
  geom_function(fun = dnorm, args = list(mean = mean(trees$Height),
     sd = sd(trees$Height)),aes(color="Normal"),linetype="dashed",linewidth=1) +
     scale_color_manual(name="",values=c("blue","red"))# Adding normal density
```




```r
#Volume:- kernel density plot overlay with normal distribution
ggplot(trees, aes(x = Volume)) +
  stat_bin(breaks = seq(10, 80, by = 10), #using 10 bins for better redability
           aes(y=after_stat(density)),
           colour = "black",fill = "lightblue",closed = "right") +
           scale_x_continuous(breaks = seq(10, 80, by = 10)) +
  geom_density(aes(color="Kernel Density"),linewidth=1) + # Adding kernel density
  geom_function(fun = dnorm, args = list(mean = mean(trees$Volume),
     sd = sd(trees$Volume)),aes(color="Normal"),linetype="dashed",linewidth=1) +
     scale_color_manual(name="",values=c("blue","red"))# Adding normal density
```




```r
#Girth:- kernel density plot overlay with normal distribution
ggplot(trees, aes(x = Girth)) +
  stat_bin(breaks = seq(8, 22, by = 2),aes(y=after_stat(density)),
           colour = "black",fill = "lightpink",closed = "right") +
           scale_x_continuous(breaks = seq(8, 22, by = 2)) +
  geom_density(aes(color="Kernel Density"),linewidth=1) + # Adding kernel density
  geom_function(fun = dnorm, args = list(mean = mean(trees$Girth),
     sd = sd(trees$Girth)),aes(color="Normal"),linetype="dashed",linewidth=1) +
     scale_color_manual(name="",values=c("blue","red"))# Adding normal density
```



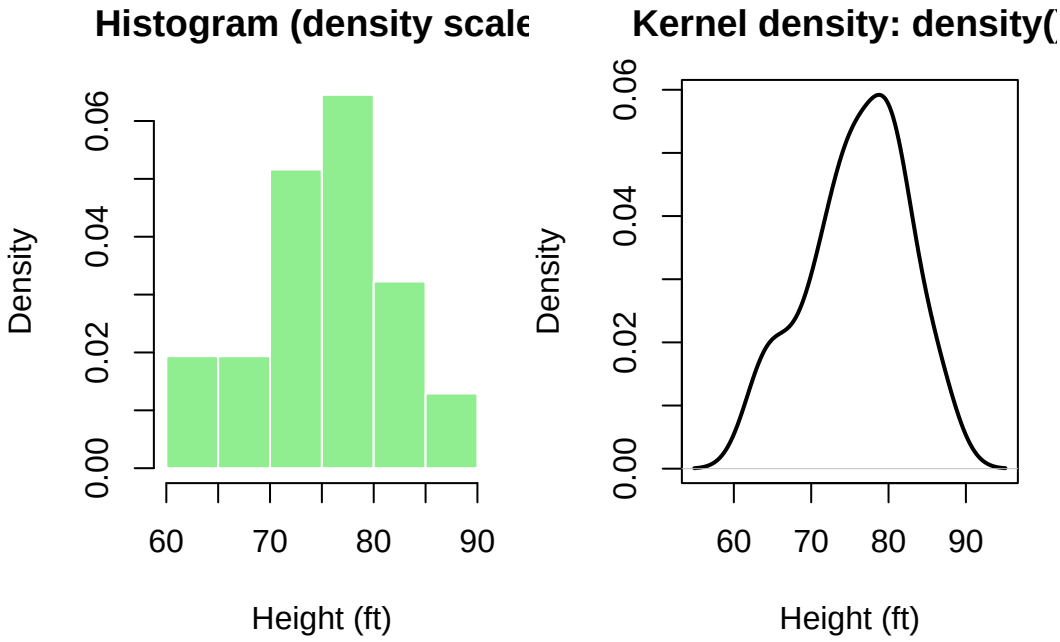
Comments:

1. Height is **well approximated by a normal**. KDE confirms only a **mild right-tail departure**. Normal-based methods are reasonable here.
2. The normal curve is **poor** since its **pulled right** by the long tail (mean > median), so its peak sits **to the right of the KDE's peak**. It **overstates density** around the mid-30s to 40s and **understates** the heavy right tail structure and the high concentration near ~20.
3. Normal is **serviceable** in the center but not ideal in the tail (again, mean > median under right skew).

c. Distinguish between the *density* option in the *hist()* function with the command *density()*.

```r
x <- trees$Height

op <- par(mfrow = c(1, 2), mar = c(4,4,3,1))

# Histogram on density scale (area  1)
hist(x, breaks = 6, freq = FALSE,
     col = "lightgreen", border = "white",
     main = "Histogram (density scale)",
     xlab = "Height (ft)")
```

```r

# KDE as a smooth density curve
plot(density(x), lwd = 2,
     main = "Kernel density: density()",
     xlab = "Height (ft)")
```




```r
par(op)
```

Comments:

1. The `density=True` option in a histogram function modifies the y-axis scaling of a binned frequency plot to represent probability density, where the area of the bars sums to 1.
2. A `density()` function, on the other hand, performs kernel density estimation to create a smoothed, continuous representation of the data's probability density function.

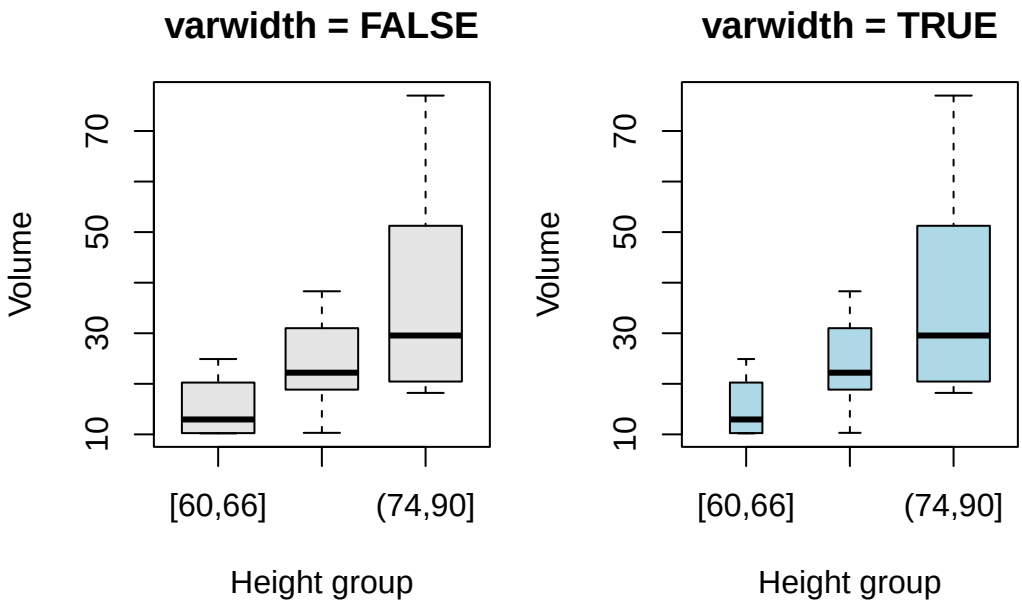
d. Explain what you see when you use the *boxplot()* function as *boxplot(Volume, varwidth = TRUE)*.

When using `boxplot(Volume, varwidth = TRUE)`, the logical argument tells the function to draw the width of each box in proportion to the square root of the number of observations in that group. Allowing for a visual representation of the number of observations contributing to each box.

```r
ht_grp2 <- cut(trees$Height, breaks = c(60,66,74,90), include.lowest = TRUE)
table(ht_grp2)
```

```
ht_grp2
[60,66] (66,74] (74,90]
      4       7      20
```

```r
par(mfrow = c(1,2), mar = c(5,4,3,1))

# Constant-width boxes
boxplot(Volume ~ ht_grp2, data = trees,
        varwidth = FALSE, boxwex = 0.7,
        col = "grey90", border = "black",
        main = "varwidth = FALSE", xlab = "Height group", ylab = "Volume")

# Widths  √(group size)
boxplot(Volume ~ ht_grp2, data = trees,
        varwidth = TRUE, boxwex = 0.7,
        col = "lightblue", border = "black",
        main = "varwidth = TRUE", xlab = "Height group", ylab = "Volume")
```




```r
par(mfrow = c(1,1))
```

```r
ht_grp2 <- cut(trees$Height, breaks = c(60,66,74,90), include.lowest = TRUE)
cbind(n = as.vector(table(ht_grp2)),
      sqrt_scale = round(sqrt(table(ht_grp2))/max(sqrt(table(ht_grp2))), 2))
```

```
         n sqrt_scale
[60,66]  4       0.45
(66,74]  7       0.59
(74,90] 20       1.00
```

Comments:

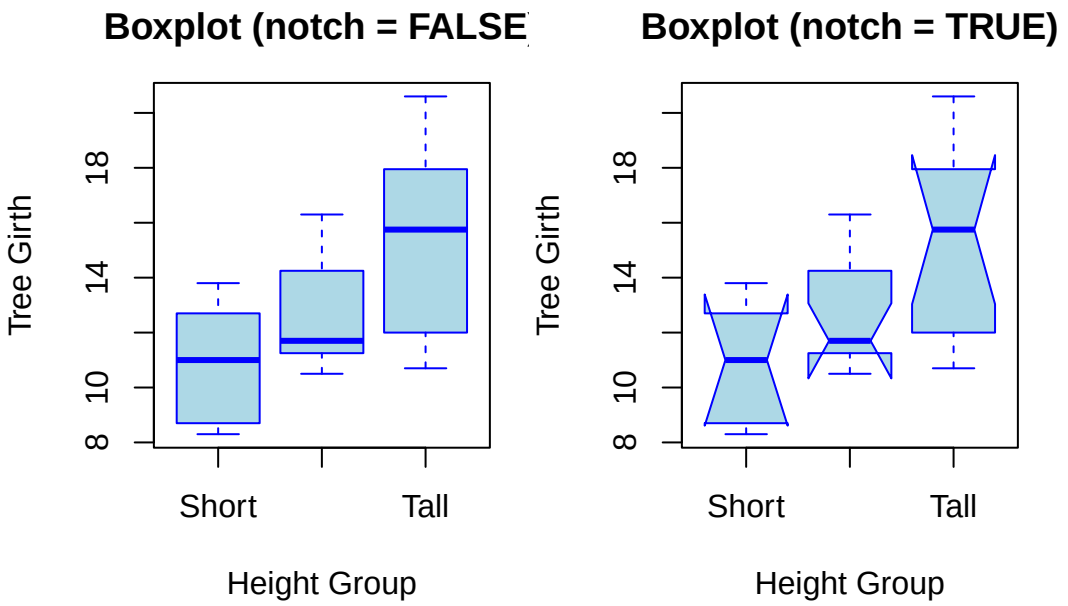
1. **Left (varwidth = FALSE):** All three boxes have the **same width**, so we can compare medians/IQRs, but we can't tell how many trees are in each height group.
2. **Right (varwidth = TRUE):** Box widths are **proportional to sqr root of (group size)**. The **(74,90]** group is visibly **widest** (it has the most trees), **(66,74]** is medium, and **[60,66]** is narrow (fewest trees). Width only encodes *sample size*; it doesn't change any statistics.

e. Explain what you see when you use the *boxplot()* function as *boxplot(Girth, notch = TRUE)*.

The option *notch=T* provides notches on each side of the box. If the notches in two boxplots do not overlap, we can conclude that the two medians differ.

```r
par(mfrow = c(1, 2), mar = c(5,4,3,1))

# (a) standard boxplot (no notch)
boxplot(Girth ~ cut(Height, breaks = 3, labels = c("Short","Medium","Tall")),
        data = trees,
        notch = FALSE,
        xlab = "Height Group",
        ylab = "Tree Girth",
        border = "blue", col = "lightblue",
        main = "Boxplot (notch = FALSE)")

# (b) notched boxplot
boxplot(Girth ~ cut(Height, breaks = 3, labels = c("Short","Medium","Tall")),
        data = trees,
        notch = TRUE,
        xlab = "Height Group",
        ylab = "Tree Girth",
        border = "blue", col = "lightblue",
        main = "Boxplot (notch = TRUE)")
```

```
Warning in (function (z, notch = FALSE, width = NULL, varwidth = FALSE, : some
notches went outside hinges ('box'): maybe set notch=FALSE
```



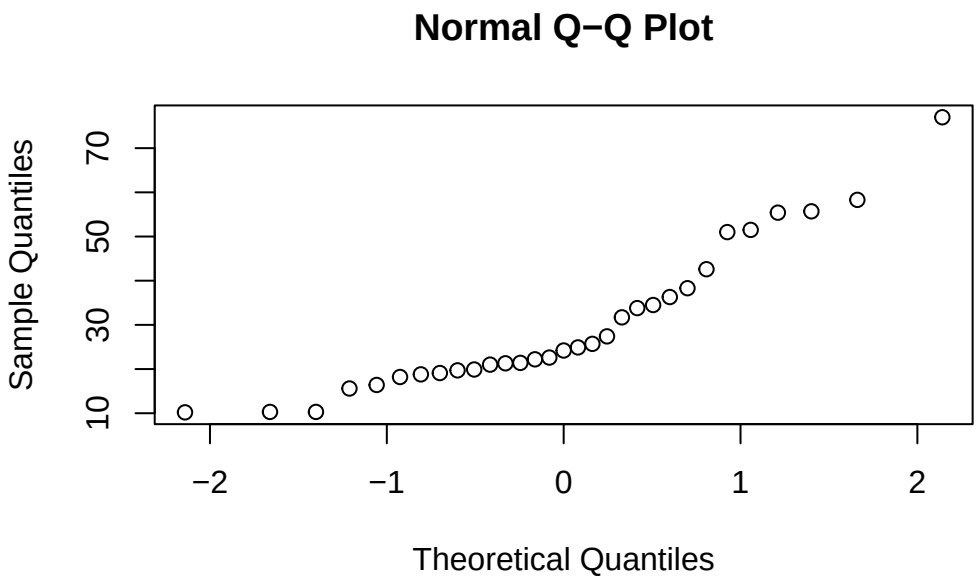
Comments:

The left plot (`notch = FALSE`) shows standard boxplots of tree girth by height group. The right plot (`notch = TRUE`) adds notches around the medians, which represent a 95% confidence interval. Non-overlapping notches (e.g., Short vs Tall) suggest a significant difference in median girth, while overlapping notches (Short vs Medium) indicate no clear difference. Thus, tree girth increases with height, and the notched boxplot makes median comparisons more interpretable.

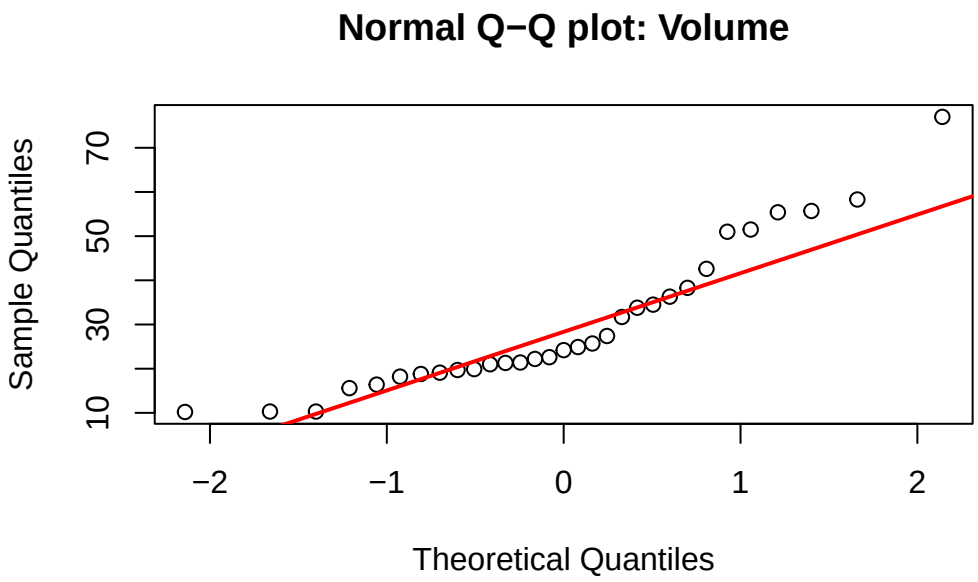
Question 2: Use the trees data in the R package *datasets*. Discuss whether you can assume that ***Volume*** follows a normal distribution, and if not, in what way(s) the data departs from normality. To answer this question, use

a. the normal Q-Q plot,

```r
x <- trees$Volume

# Q-Q plot (separate figure)
qqnorm(trees$Volume)
```




```r
qqnorm(x, main = "Normal Q-Q plot: Volume")
qqline(x, col = "red", lwd = 2)
```



Comments:

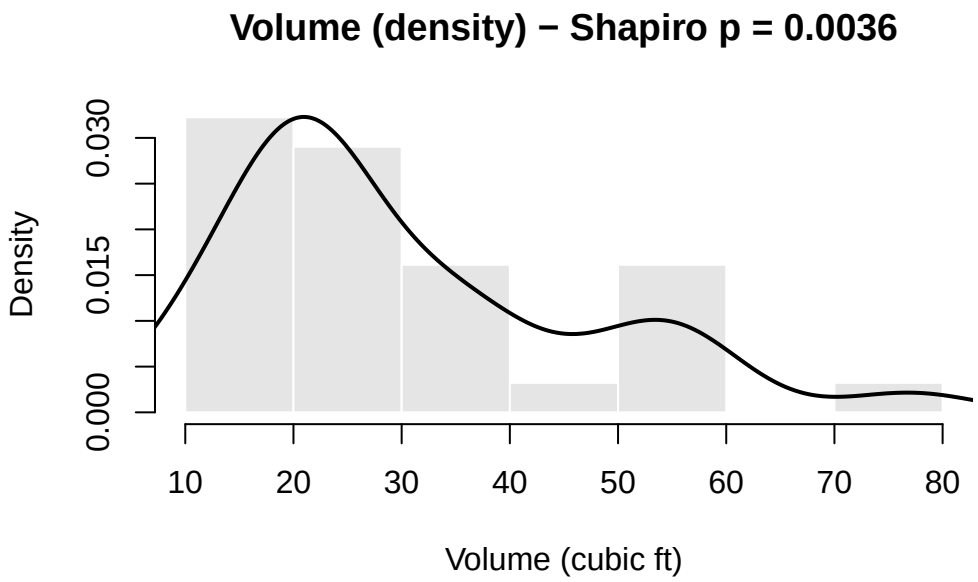
- Points follow the line through the middle but bend into a clear **S-shape**: they fall **below** the line in the lower tail and rise **above** it in the upper tail. The largest value (~**77**) sits well above the line.
- This pattern indicates **positive (right) skew** with a **heavier-than-normal right tail**. A normal distribution would place the points roughly on the straight line across all quantiles.

b. the Shapiro-Wilk test, and

```r
# Shapiro-Wilk test
sw <- shapiro.test(trees$Volume)
sw #printing test output
```

```
    Shapiro-Wilk normality test

data:  trees$Volume
W = 0.88757, p-value = 0.003579
```

```r
# Separate figure: histogram on density scale with KDE
hist(trees$Volume, breaks = 6, freq = FALSE,
     col = "grey90", border = "white",
     main = sprintf("Volume (density) - Shapiro p = %.4f", sw$p.value),
     xlab = "Volume (cubic ft)")
lines(density(trees$Volume), lwd = 2)
```



Comments:

- **Result shown:** `p 0.0036p \approx 0.0036p 0.0036`
- At the 5% (and even 1%) level, we **reject the null hypothesis of normality** for `Volume`. Combined with the Q–Q plot, the departure is due to **right skew / heavy right tail** (not random noise).

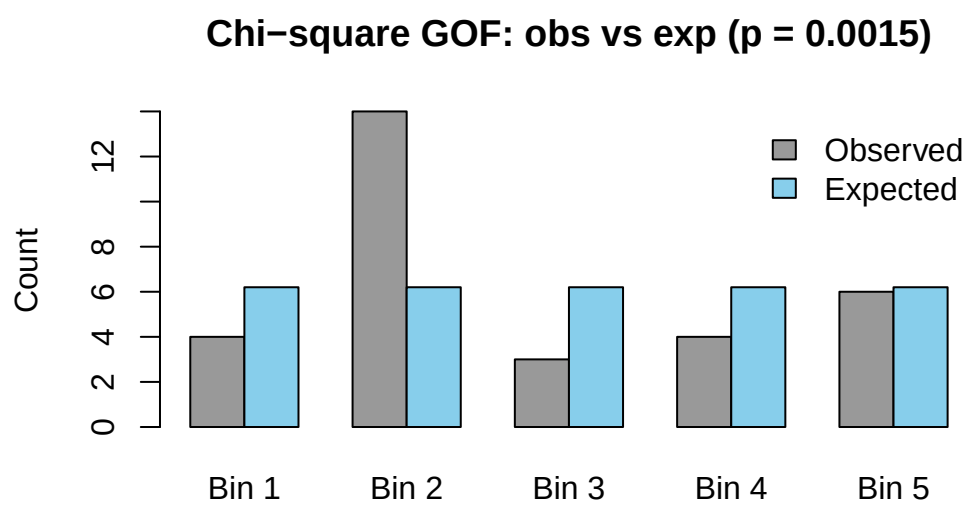
c. the chi-square goodness of fit test for normality.

```r
# fitting normal parameters from data
mu <- mean(x); sigma <- sd(x); n <- length(x)

# using 5 equal-probability bins under the fitted normal
cuts <- qnorm(c(0, .2, .4, .6, .8, 1), mean = mu, sd = sigma)
cuts[1] <- -Inf; cuts[length(cuts)] <- Inf

# observed vs expected counts
obs <- as.numeric(table(cut(x, breaks = cuts, include.lowest = TRUE, right = TRUE)))
exp <- rep(n/5, 5)

# Chi-square stat and p-value (df = k - p - 1 = 5 - 2 - 1 = 2)
chisq_stat <- sum((obs - exp)^2 / exp)
df <- 2
```

```r
p_val <- pchisq(chisq_stat, df = df, lower.tail = FALSE)

# printing results
list(observed = obs, expected = round(exp, 2),
     chisq_stat = unname(chisq_stat), df = df, p_value = unname(p_val))
```

```
$observed
[1]  4 14  3  4  6

$expected
[1] 6.2 6.2 6.2 6.2 6.2

$chisq_stat
[1] 13.03226

$df
[1] 2

$p_value
[1] 0.001479385
```

```r
# Plot observed vs expected (separate figure)
barplot(rbind(obs, exp), beside = TRUE,
        col = c("grey60", "skyblue"),
        names.arg = paste("Bin", 1:5),
        ylab = "Count",
        main = sprintf("Chi-square GOF: obs vs exp (p = %.4f)", p_val))
legend("topright", fill = c("grey60","skyblue"),
       legend = c("Observed", "Expected"), bty = "n")
```



Comments:

- **How the test was set up:** 5 **equal-probability** bins under the fitted Normal(,), so each has the same **expected** count `(~6.2 with n=31n=31n=31).`
- The **observed** bars differ notably from the **expected** bars—e.g., an **excess** of observations in one lower-volume bin and **deficits** in mid/high bins, plus mass in the far right tail. Title shows p 0.0015p 0.0015p 0.0015.
- With df = 2, the small p-value again leads us to **reject normality**; the direction of deviation matches the Q–Q and Shapiro findings.

Overall conclusion

`Volume` **does not follow a normal distribution**. The data **depart from normality via right skew and a long right tail**, driven by a few large trees (e.g., ~77 cu ft).

Question 3: Use the mtcars data from the R package *datasets*.

Data Exploration

Dataframe has 32 observations (rows) and 11 numeric variables (columns).

A quick look at the dataset (`summary()`) shows that:

- **mpg** (fuel efficiency) ranges **10.4–33.9** mpg, median **19.2**.
- **disp** (engine displacement) **71–472** cu in, median **196.3**; **hp 52–335**, median **123**.
- **wt** (weight) **1.51–5.42** (1000 lbs), median **3.33**.
- **drat 2.76–4.93**, median **3.695**; **qsec** (1/4 mile) **14.5–22.9**, median **17.71**.

Categorical: **cyl** {4,6,8} (median 6); **gear** {3,4,5}; **carb** {1–8}.

Correlation patterns (from `cor(mtcars)` and the scatterplot matrix):

- **mpg** shows strong **negative** correlations with **wt** → heavier, larger, and more powerful cars get lower mileage.
- Engine/size variables are strongly **positively** correlated with each other (e.g., **disp–wt** ~0.89, **disp–cyl** ~0.90, **hp–disp** ~0.79) → expect **multicollinearity** if used together in a model.
- **drat** is positively related to **gear** and **am** (manual transmissions tend to higher rear-axle ratios).
- **qsec** is positively associated with **vs** (V/S engine shape indicator) and negatively with **hp** (faster cars have shorter times).

Distributional notes:

- Several continuous variables (**disp, hp, wt**) are **right-skewed**; **mpg** is roughly unimodal with lighter right tail.
- Discrete variables (**cyl, gear, carb**) create visible vertical bands in pair plots.

The dataset is clean with no null values in the column.

```r
#loading mtcars dataset
data(mtcars)
```

```r
?mtcars
```

```r
#checking head of mtcars dataset
head(mtcars)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
```

```
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
#checking head of trees dataset
tail(mtcars)
```

```
               mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

```
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

```
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
#checking for missing values
colSums(is.na(mtcars))
```

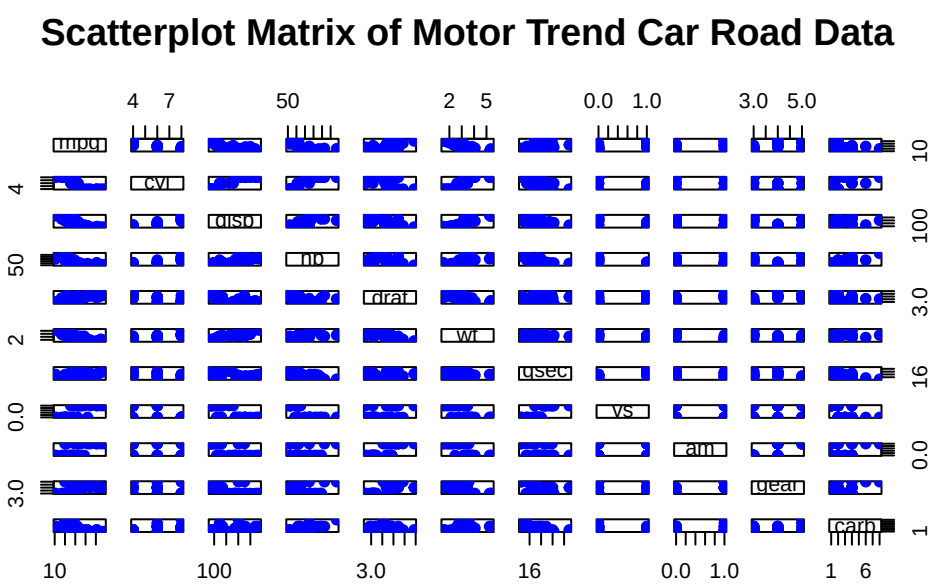
```
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   0    0    0    0    0    0    0    0    0    0    0
```

```
cor(mtcars)
```

```
            mpg        cyl       disp         hp        drat         wt
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
            qsec         vs          am       gear        carb
mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
```

```
vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

```r
#scatterplot matrix of all the numerical variables
pairs(mtcars,
      main = "Scatterplot Matrix of Motor Trend Car Road Data",
      pch = 19, col = "blue")
```

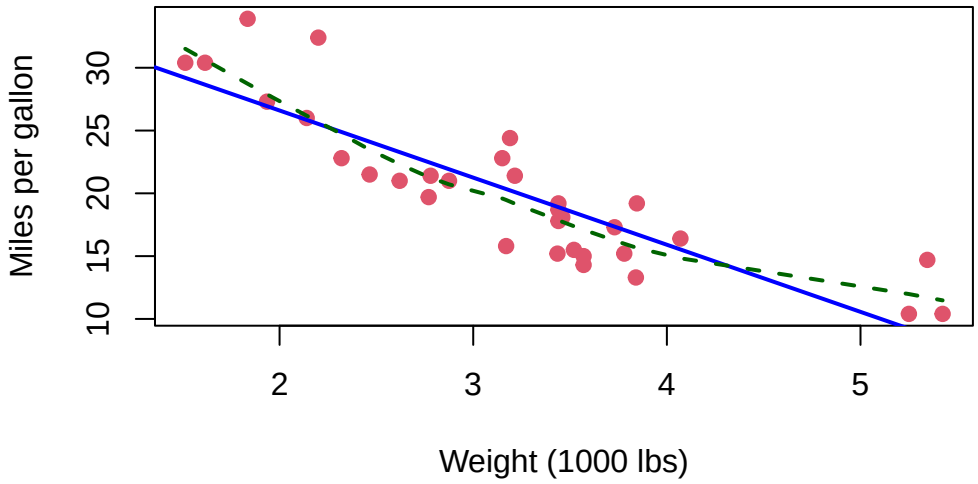


a. Draw a scatterplot of the variable ***mpg*** versus the variable ***wt***. Discuss whether these two variables are associated.

```r
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```r
## Scatterplot: mpg vs wt
with(mtcars, {
  plot(wt, mpg,
       main = NA,
       xlab = "Weight (1000 lbs)",
       ylab = "Miles per gallon",
       pch = 20, col = 2, cex = 1.5)
  abline(lm(mpg ~ wt), col = "blue", lwd = 2)        # linear fit (reference)
  lines(lowess(wt, mpg), col = "darkgreen", lwd = 2, lty = 2)  # smooth fit
})
```




```r
cor(mtcars$wt, mtcars$mpg)
```

```
[1] -0.8676594
```

```r
coef(lm(mpg ~ wt, data = mtcars))
```

```
(Intercept)          wt
  37.285126   -5.344472
```

```r
summary(lm(mpg ~ wt, data = mtcars))$r.squared
```

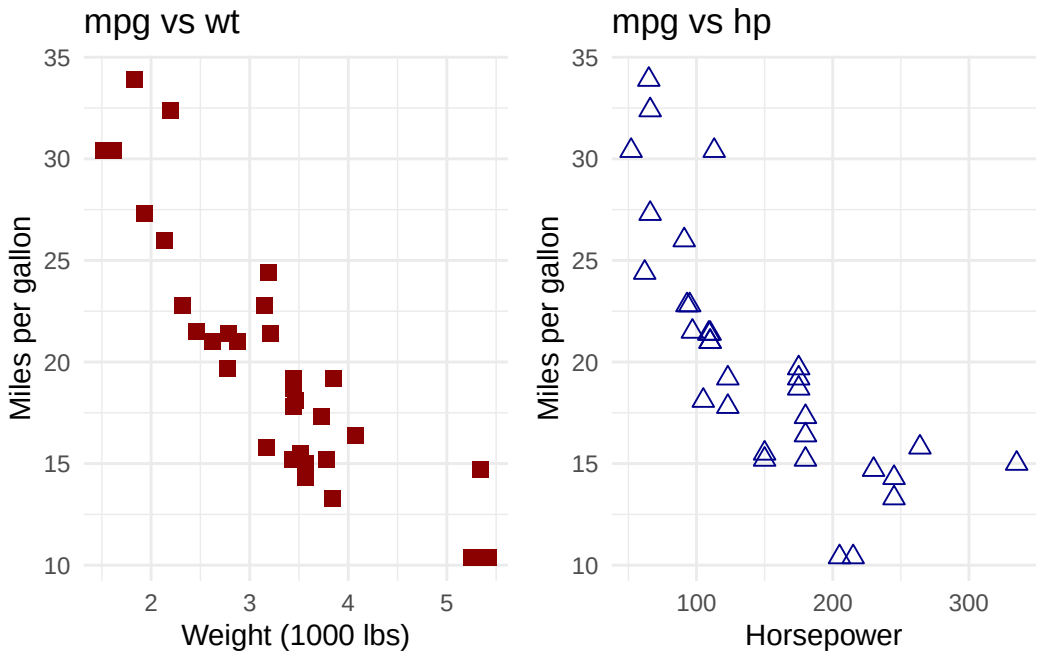
```
[1] 0.7528328
```

Comments:

- The scatterplot of **mpg** versus **wt** shows a **clear negative association**: as car weight increases, fuel economy decreases.
- The fitted regression line and the LOESS smooth nearly coincide, indicating the relationship is **approximately linear** across the observed range.
- The sample **correlation is about −0.87**, and a simple regression of mpg on wt yields a slope of roughly **−5.3 mpg per additional 1000 lbs**, explaining about **75% of the variation** in mpg (**R^2 0.75**).
- **Conclusion:** mpg and wt are **strongly and inversely associated**; heavier cars get fewer miles per gallon.

b. Repeat (a) using the R package *ggplot2* to construct the scatterplot.

```r
# Scatterplot using ggplot
s1 <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(shape = 15, color = "darkred", size = 2.5) +
  labs(title = "mpg vs wt", x = "Weight (1000 lbs)", y = "Miles per gallon") +
  theme_minimal()

s2 <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(shape = 2, color = "darkblue", size = 2.5) +
  labs(title = "mpg vs hp", x = "Horsepower", y = "Miles per gallon") +
  theme_minimal()

# draw them
grid.arrange(s1, s2, nrow = 1)
```

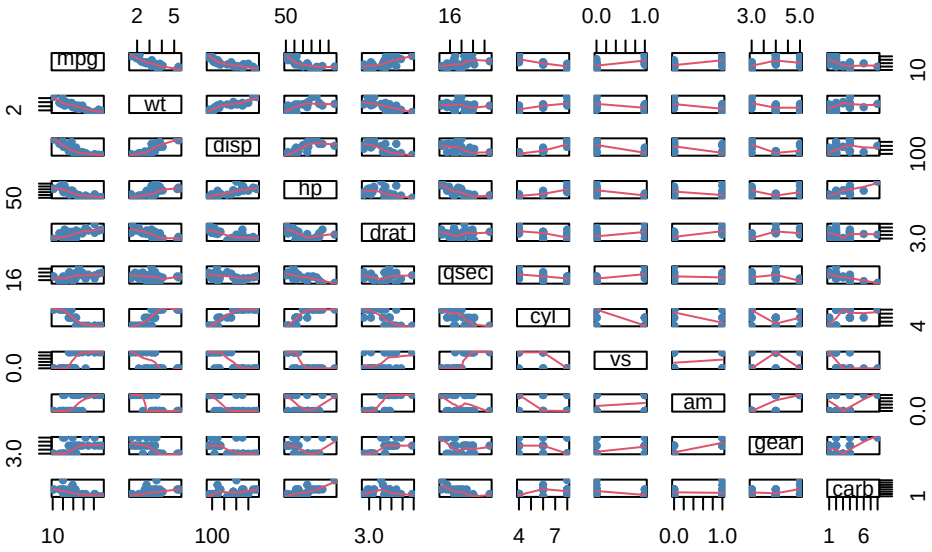


Comment:

1. The dataset shows the classic efficiency trade-off—**heavier and more powerful cars consume more fuel**—with **weight** being the dominant driver of fuel economy among the two.
2. The **left panel is tighter and steeper**, indicating a **stronger relationship with weight** than with **horsepower**. The relationship is **approximately linear** in both cases, with no glaring outliers contradicting the trend.
3. The **right panel** is still clearly downward but **more dispersed**, suggesting a weaker association than weight. Extremely high-hp cars (far right) look like **leverage points** but follow the same trend.

c. Create a matrix scatterplot for the variables in the data. With which variables is the variable ***mpg*** highly associated?

```
## Scatterplot matrix (mpg listed first)
pairs(mtcars[, c("mpg","wt","disp","hp","drat","qsec",
                 "cyl","vs","am","gear","carb")],
      panel = panel.smooth,    # adds a LOWESS smoother in each panel
      pch = 20, col = "steelblue")
```



Comments:

- As was already added in Data Exploration (please navigate to Data Exploration), the scatterplot matrix shows **mpg** has a **clear negative association** with **weight (wt)**, **displacement (disp)**, **number of cylinders (cyl)**, and **horsepower (hp)**—the point clouds slope downward with tight scatter. This means, cars that are **heavier**, have **larger engines** (higher displacement/cylinders), or **more horsepower** tend to achieve **lower mpg**. Among these, **weight** is the single strongest predictor of fuel efficiency in the bivariate plots.

d. Which pair of variables in the mtcars data has the highest correlation? *Hint*: use the *cor()* function.

   As already added in the Data Exploration (Please check Correlation Chunck of Data Exploration of Problem 3):-

   - **mpg** shows strong **negative** correlations with **wt** → heavier, larger, and more powerful cars get lower mileage.
   - Engine/size variables are strongly **positively** correlated with each other (e.g., **disp**–**wt** ~0.89, **disp**–**cyl** ~0.90, **hp**–**disp** ~0.79) → expect **multicollinearity** if used together in a model.
   - **drat** is positively related to **gear** and **am** (manual transmissions tend to higher rear-axle ratios).

- **qsec** is positively associated with **vs** (V/S engine shape indicator) and negatively with **hp** (faster cars have shorter times).

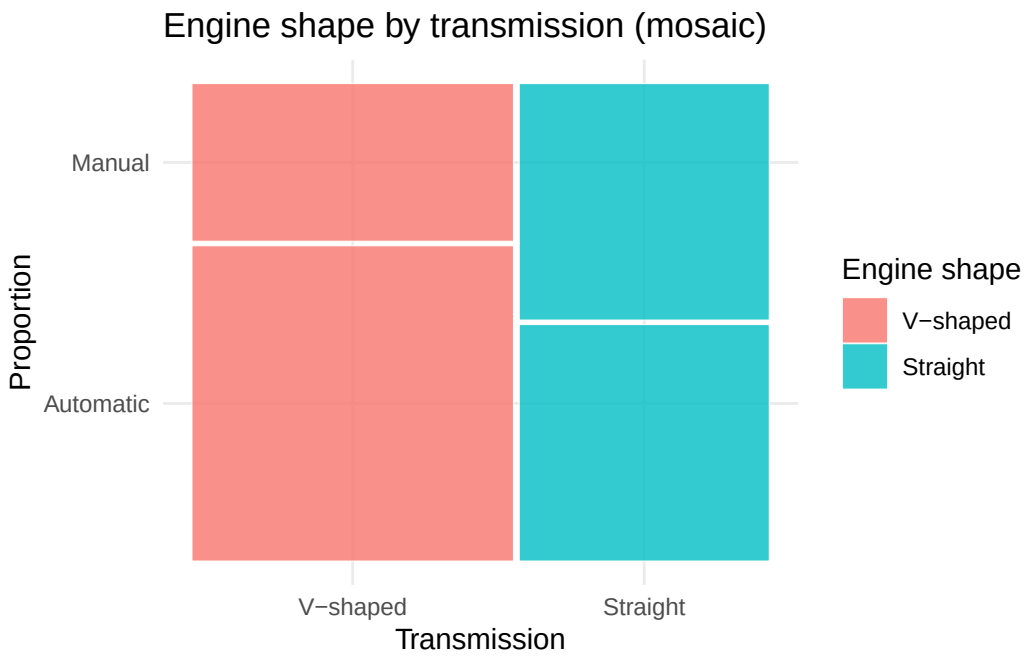
e. Use a spineplot to discuss whether automatic cars use v-shaped engines more often than straight engines.

```
# Spineplot using ggplot
library(ggmosaic)

mt <- transform(
  mtcars,
  am_f = factor(am, levels = c(0,1), labels = c("Automatic","Manual")),
  vs_f = factor(vs, levels = c(0,1), labels = c("V-shaped","Straight"))
)

ggplot(mt) +
  geom_mosaic(aes(x = product(am_f, vs_f), fill = vs_f), color = "white") +
  labs(x = "Transmission", y = "Proportion", fill = "Engine shape",
       title = "Engine shape by transmission (mosaic)") +
  theme_minimal()
```

```
Warning: The `scale_name` argument of `continuous_scale()` is deprecated as of ggplot2
3.5.0.
```

```
Warning: The `trans` argument of `continuous_scale()` is deprecated as of ggplot2 3.5.0.
i Please use the `transform` argument instead.
```

```
Warning: `unite_()` was deprecated in tidyr 1.2.0.
i Please use `unite()` instead.
i The deprecated feature was likely used in the ggmosaic package.
  Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
```



Comments:

1. The **height of each colored segment** within a column gives the **proportion of transmissions within that engine shape**: the lower segment is **Automatic**, the upper is **Manual**.
2. Visually, the **Automatic** slice is **much larger** in the **V-shaped** column than in the **Straight** column, while the **Manual** slice is larger in the **Straight** column.
3. ***Conclusion***: *Yes—**automatic cars use V-shaped engines more often than straight engines**. In contrast, **manual** cars are more commonly **straight** (inline) engines. The mosaic makes this clear: the **Automatic** proportion is high for **V-shaped** and low for **Straight**.*

Question 4: Download historical weather data for four or more Indian cities from kaggle. Create your own interesting visualization(s) of this data, and discuss.

Comments:

Using all the weather data from provided kaggle link.

Data Importation

```r
getwd()
```

```
[1] "/Users/pokeapokemon/playground_pooja/ML"
```

Loading the csv and previewing first few records of the data

```r
library("janitor")
# ------Towards my folder where files exist-----------------------
data_dir  <- "/Users/pokeapokemon/playground_pooja/Stats"
file_list <- list.files(path = data_dir, pattern = "\\.csv$", full.names = TRUE)

# Read each CSV; carry source_path while reading; force 'time' to character
read_one <- function(fp) {
  readr::read_csv(
    fp,
    id = "source_path",
    show_col_types = FALSE,
    col_types = readr::cols(time = readr::col_character())
  )
}

combined_raw <- purrr::map_dfr(file_list, read_one) |> clean_names()

# Parse time -> date robustly
combined_raw <- combined_raw |>
  mutate(
    date = as_date(parse_date_time(
      time,
      orders = c("ymd HMS","ymd HM","ymd H","ymd",
                 "dmy HMS","dmy HM","dmy H","dmy")
    ))
  )

# Add city from filename; then DROP source_path and file_stem
combined_data <- combined_raw %>%
  mutate(
    source_file = basename(source_path),
    file_stem   = tools::file_path_sans_ext(source_file),
    city = case_when(
      str_detect(file_stem, "Bangalore|BangaloreCity") ~ "Bengaluru",
      str_detect(file_stem, "Madras|Chennai")          ~ "Chennai",
```

```
      str_detect(file_stem, "Safdarjung|Delhi")        ~ "Delhi",
      str_detect(file_stem, "Santacruz|Mumbai")        ~ "Mumbai",
      str_detect(file_stem, "Jodhpur")                 ~ "Jodhpur",
      str_detect(file_stem, "Lucknow")                 ~ "Lucknow",

      TRUE ~ file_stem
    )
  ) %>%
  relocate(city, date, .before = 1) %>%
  select(-source_path, -file_stem) %>%   # <- remove scaffolding cols
  # optionally drop the raw time string and/or source_file:
  # select(-time) %>%
  # select(-source_file) %>%
  identity()


View(combined_data)
```

```
weather_data <- combined_data %>%
  select(-any_of(c("source_file", "time")))

View(weather_data)
```

Data Exploration

```
structure(weather_data)
```

```
# A tibble: 71,364 x 6
   city      date        tavg  tmin  tmax  prcp
   <chr>     <date>     <dbl> <dbl> <dbl> <dbl>
 1 Bengaluru 1990-01-01  22.9  19.1  28.4    NA
 2 Bengaluru 1990-01-02  21.7  NA    26.5     0
 3 Bengaluru 1990-01-03  21    16.4  26.5     0
 4 Bengaluru 1990-01-04  20.8  NA    27.4     0
 5 Bengaluru 1990-01-05  20.4  14.2  26.1     0
 6 Bengaluru 1990-01-06  20.4  17.1  24.2    NA
 7 Bengaluru 1990-01-07  18.8  NA    20.5    NA
 8 Bengaluru 1990-01-08  20    16.6  25.1     0
 9 Bengaluru 1990-01-09  21    15.5  NA       0
```

```
10 Bengaluru 1990-01-10  21.2  15    27.7     0
# i 71,354 more rows
```

```
str(weather_data)
```

```
tibble [71,364 x 6] (S3: tbl_df/tbl/data.frame)
 $ city: chr [1:71364] "Bengaluru" "Bengaluru" "Bengaluru" "Bengaluru" ...
 $ date: Date[1:71364], format: "1990-01-01" "1990-01-02" ...
 $ tavg: num [1:71364] 22.9 21.7 21 20.8 20.4 20.4 18.8 20 21 21.2 ...
 $ tmin: num [1:71364] 19.1 NA 16.4 NA 14.2 17.1 NA 16.6 15.5 15 ...
 $ tmax: num [1:71364] 28.4 26.5 26.5 27.4 26.1 24.2 20.5 25.1 NA 27.7 ...
 $ prcp: num [1:71364] NA 0 0 0 0 NA NA 0 0 0 ...
```

```
#checking for missing values
colSums(is.na(weather_data))
```

```
 city  date  tavg  tmin  tmax  prcp
    0     0   410 13367  6270 31099
```

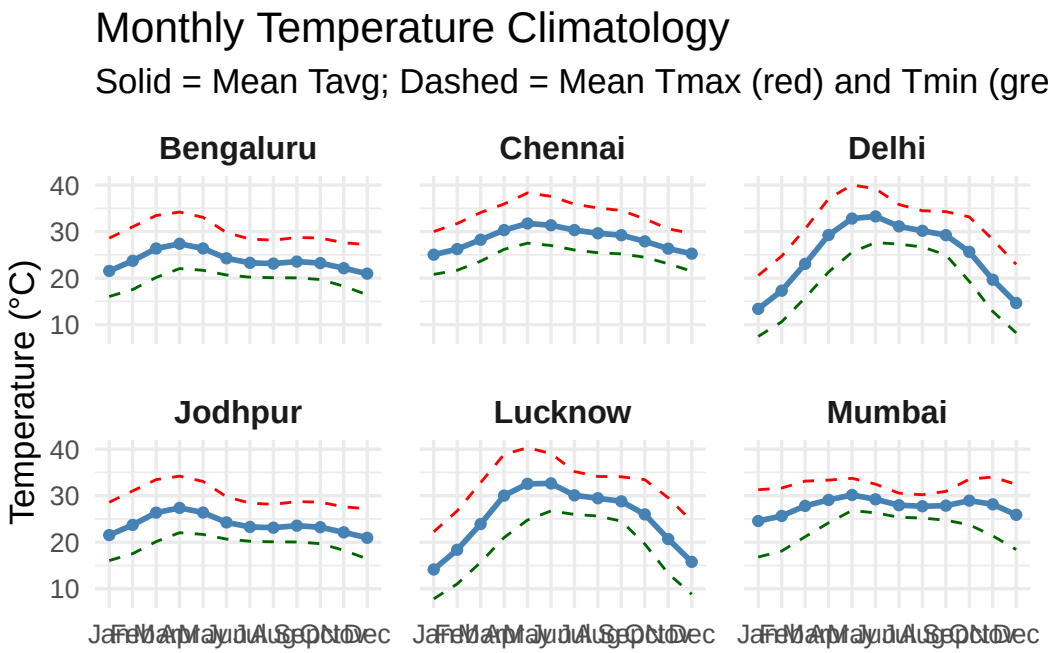
```
summary(weather_data)
```

```
    city                date                tavg            tmin
 Length:71364       Min.   :1990-01-01   Min.   : 5.7   Min.   :-0.60
 Class :character   1st Qu.:1998-02-21   1st Qu.:22.9   1st Qu.:17.90
 Mode  :character   Median :2009-06-20   Median :26.2   Median :20.70
                    Mean   :2009-09-15   Mean   :25.7   Mean   :20.49
                    3rd Qu.:2020-06-20   3rd Qu.:29.1   3rd Qu.:24.50
                    Max.   :2031-12-20   Max.   :39.8   Max.   :34.20
                                         NA's   :410    NA's   :13367
     tmax            prcp
 Min.   : 9.80   Min.   :  0.000
 1st Qu.:28.80   1st Qu.:  0.000
 Median :31.70   Median :  0.000
 Mean   :31.69   Mean   :  5.812
 3rd Qu.:34.70   3rd Qu.:  2.300
 Max.   :48.10   Max.   :470.900
 NA's   :6270    NA's   :31099
```

Data Visualization

Monthly Climatology

```r
# --- Compute monthly climatology ---
clim <- weather_data |>
  mutate(month = month(date, label = TRUE, abbr = TRUE)) |>
  group_by(city, month) |>
  summarise(
    tavg_m = mean(tavg, na.rm = TRUE),
    tmax_m = mean(tmax, na.rm = TRUE),
    tmin_m = mean(tmin, na.rm = TRUE),
    .groups = "drop"
  )

# --- Plot temperature climatology ---
ggplot(clim, aes(month, tavg_m, group = 1)) +
  geom_line(color = "steelblue", linewidth = 1) +
  geom_point(color = "steelblue", size = 1.5) +
  geom_line(aes(y = tmax_m), linetype = "dashed", color = "red") +
  geom_line(aes(y = tmin_m), linetype = "dashed", color = "darkgreen") +
  facet_wrap(~ city, ncol = 3, scales = "fixed") +   # shared y-axis
  labs(
    title = "Monthly Temperature Climatology",
    subtitle = "Solid = Mean Tavg; Dashed = Mean Tmax (red) and Tmin (green)",
    x = NULL, y = "Temperature (°C)"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    strip.text = element_text(face = "bold", size = 12),
    panel.spacing = unit(1, "lines")   # add breathing room between panels
  )
```



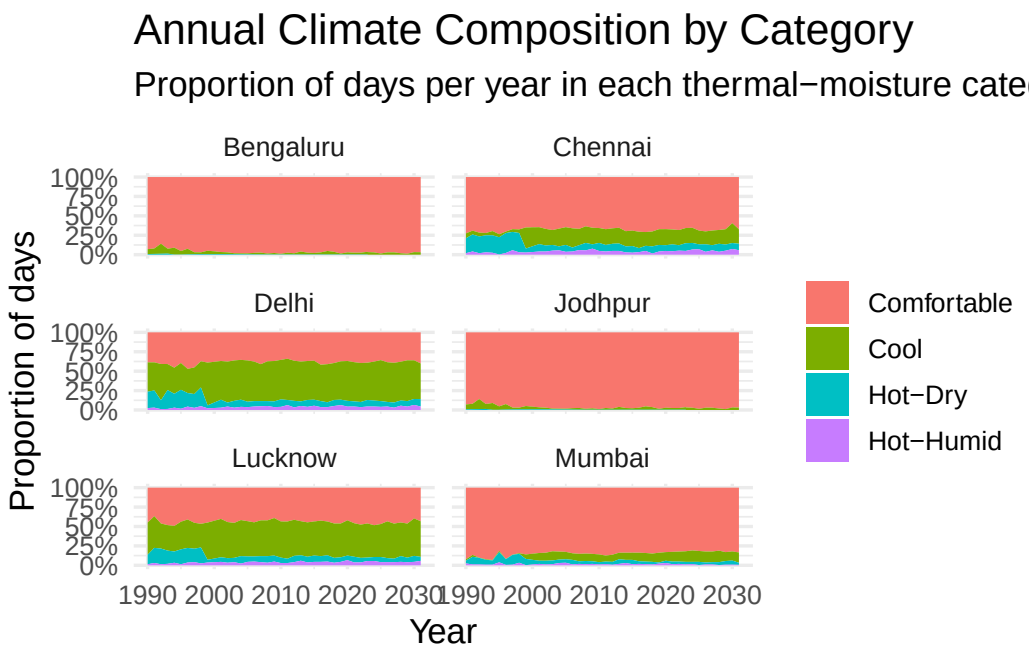
Key Findings

- **Geography drives climate:** inland plains experience **extreme heatwaves and cold winters**, western India stays hot and dry, coastal cities stay **warm but stable**, and Bengaluru remains **mild**.
- The plots clearly separate **continental vs maritime vs plateau climates** in India.
- Pre-monsoon heat stress is a defining feature inland, while coasts experience a much more even thermal profile.

Heat Classification Calendar

```
classified <- weather_data %>%
  mutate(
    year = lubridate::year(date),
    category = case_when(
      tavg >= 30 & prcp > 0       ~ "Hot-Humid",
      tavg >= 30 & prcp == 0      ~ "Hot-Dry",
      tavg >= 20 & tavg < 30      ~ "Comfortable",
      TRUE                        ~ "Cool"
```

```r
    )
  ) %>%
  group_by(city, year, category) %>%
  summarise(days = n(), .groups = "drop")

ggplot(classified, aes(year, days, fill = category)) +
  geom_area(position = "fill") +
  facet_wrap(~ city, ncol = 2) +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Annual Climate Composition by Category",
       subtitle = "Proportion of days per year in each thermal-moisture category",
       x = "Year", y = "Proportion of days", fill = NULL) +
  theme_minimal(base_size = 13)
```



Key Findings

Clear coastal vs. inland contrast :

- **Mumbai, Chennai**: a visible band of **Hot-Humid** days each year, reflecting monsoon overlap with warm temperatures. The Hot-Dry share is small.
- **Delhi, Lucknow, Jodhpur**: much larger **Hot-Dry** share, especially in the late spring/pre-monsoon season; humid heat appears mostly during monsoon peaks.

- **Bengaluru**: dominated by **Comfortable** days due to elevation; only thin slivers of either hot category.
- **Seasonality encoded in composition.** Humid heat spikes in JJAS (monsoon), while dry heat concentrates in pre-monsoon for inland cities.
- **Subtle long-term drift.** In several cities (notably **Delhi** and **Mumbai**) the fraction of hot categories edges up in recent years, hinting at warming and urban heat-island effects, though formal trend tests are recommended.

Limitations:

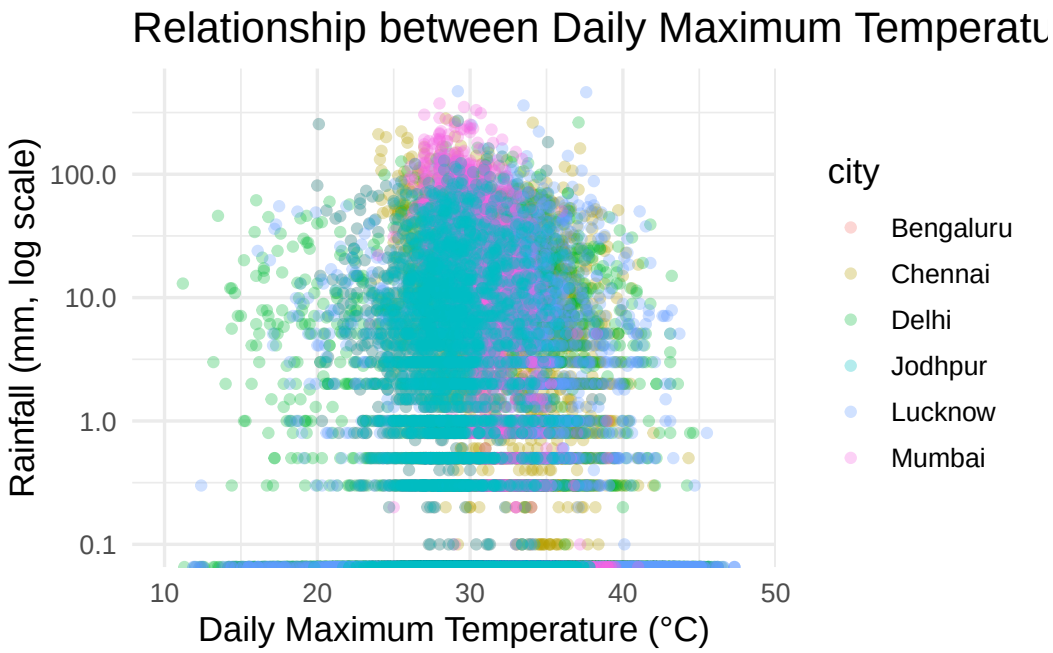
- Rainfall is an **imperfect proxy** for humidity; true humidity or dew point would sharpen the classification.
- Thresholds (`30°C`, `20°C`) are reasonable but can be tuned (e.g., 32°C for coastal cities).
- Proportions don't show intensity (e.g., 35°C vs. 40°C). Complement with heatwave metrics if needed.

Scatter of Tmax vs Rain

```
ggplot(weather_data, aes(tmax, prcp, color = city)) +
  geom_point(alpha = 0.3) +
  scale_y_log10() +  # rain is skewed
  labs(title = "Relationship between Daily Maximum Temperature and Rainfall",
       x = "Daily Maximum Temperature (°C)",
       y = "Rainfall (mm, log scale)") +
  theme_minimal(base_size = 13)
```

```
Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

```
Warning: Removed 34653 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Key Findings

- The **highest maximum temperatures (>40 °C)** cluster near **zero rainfall**, typical of dry-heat conditions in inland cities such as Delhi, Lucknow, and Jodhpur.
- Days with **very high rainfall (>100 mm)** are concentrated around moderate temperatures (~25–32 °C), common in coastal monsoon climates (Mumbai, Chennai).
- **Mumbai and Chennai** (coastal cities) show frequent overlap of moderate-to-high temperatures (28–32 °C) with substantial rainfall, indicating **humid heat**.
- **Delhi, Lucknow, Jodhpur** (inland cities) show a strong pattern of very hot days coinciding with almost no rainfall, reflecting **dry heat**.
- **Bengaluru** clusters in the 20–30 °C range with scattered rainfall, consistent with its milder plateau climate.
- Pre-monsoon (Apr–Jun): high Tmax, low rainfall (dry heat).
- Monsoon (Jun–Sept): moderate Tmax, high rainfall (humid heat).
- Winter: lower Tmax, low rainfall.

City-Level medians

```r
summary_stats <- weather_data %>%
  group_by(city) %>%
  summarise(
    median_tmax = median(tmax, na.rm = TRUE),
    median_prcp = median(prcp, na.rm = TRUE),
    mean_tmax   = mean(tmax, na.rm = TRUE),
    mean_prcp   = mean(prcp, na.rm = TRUE),
    .groups = "drop"
  )

summary_stats
```

```
# A tibble: 6 x 5
  city      median_tmax median_prcp mean_tmax mean_prcp
  <chr>           <dbl>       <dbl>     <dbl>     <dbl>
1 Bengaluru        29.5           0      29.9      4.41
2 Chennai          34             0      33.9      6.24
3 Delhi            33.2           0      31.8      3.66
4 Jodhpur          29.5           0      29.9      4.41
5 Lucknow          33.4           0      32.5      4.54
6 Mumbai           32.4           0      32.3     10.9
```

Key Findings

Coastal cities such as **Mumbai** and **Chennai** combine high median maximum temperatures (~32–34 °C) with the highest mean rainfall values (10.9 mm and 6.2 mm respectively), highlighting their hot–humid climate regimes. In contrast, inland cities like **Delhi**, **Lucknow**, and **Jodhpur** also exhibit high temperature medians (29.5–33.4 °C) but with substantially lower mean rainfall (<5 mm), confirming their tendency toward hot–dry conditions. **Bengaluru** stands out with the lowest temperature median (29.5 °C) and moderate rainfall (~4.4 mm mean), reflecting its plateau location and milder, more comfortable climate.

Rainfall heatmap (month × year)

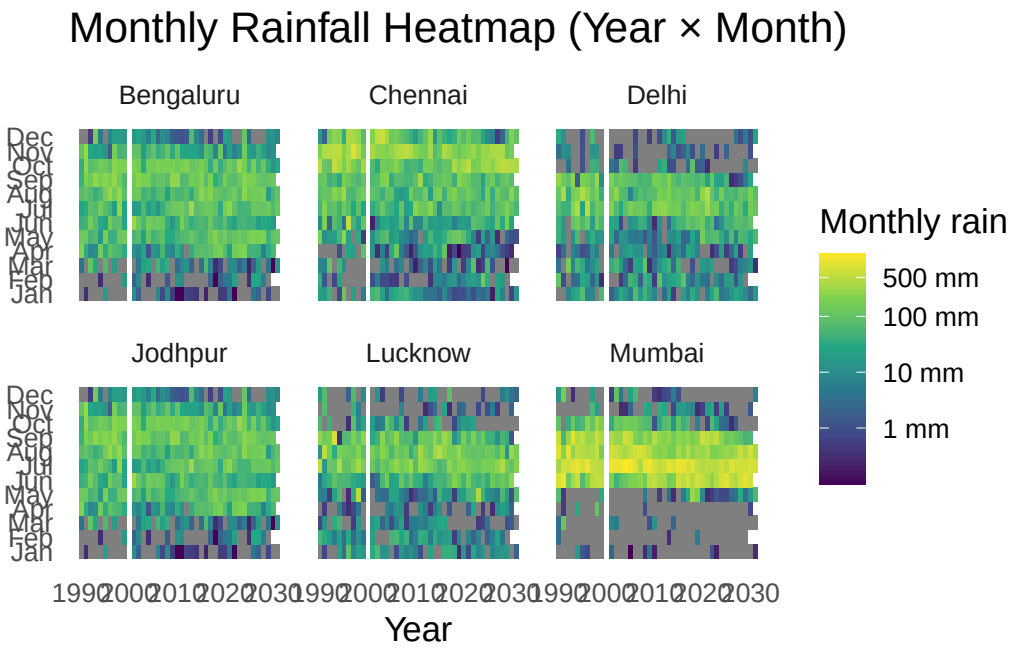
```r
library(dplyr)
library(lubridate)
```

```r
library(ggplot2)
library(scales)
```

```
Attaching package: 'scales'

The following object is masked from 'package:readr':

    col_factor

The following object is masked from 'package:purrr':

    discard
```

```r
rain_heat <- weather_data |>
  mutate(year = year(date), month = month(date)) |>
  group_by(city, year, month) |>
  summarise(rain_mm = sum(prcp, na.rm = TRUE), .groups = "drop")

ggplot(rain_heat, aes(year, month, fill = rain_mm)) +
  geom_tile() +
  scale_y_continuous(breaks = 1:12, labels = month.abb) +
  scale_fill_viridis_c(
    trans  = "log10",
    breaks = c(1, 10, 100, 500),
    labels = label_number(accuracy = 1, suffix = " mm"),   # <- replacement
    name   = "Monthly rain"
  ) +
  facet_wrap(~ city, ncol = 3) +
  labs(title = "Monthly Rainfall Heatmap (Year × Month)",
       x = "Year", y = NULL) +
  theme_minimal(base_size = 13) +
  theme(panel.grid = element_blank())
```

```
Warning in scale_fill_viridis_c(trans = "log10", breaks = c(1, 10, 100, :
log-10 transformation introduced infinite values.
```



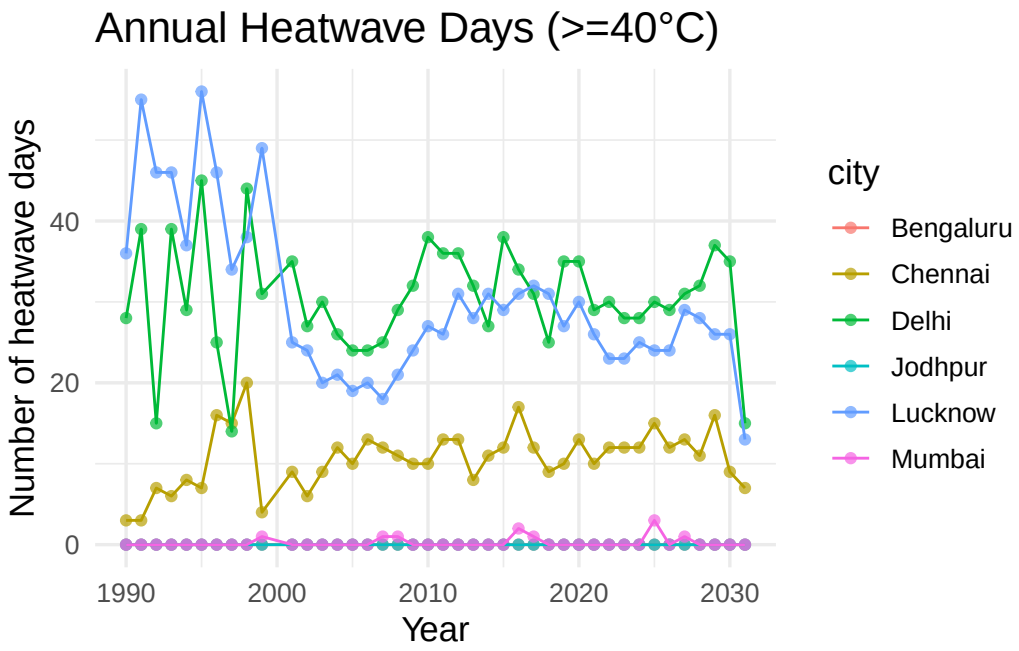
Key Findings

The heatmap makes it clear that while the **monsoon dominates the rainfall climatology**, its **strength fluctuates sharply year to year**. Good rainfall years stand out with bright, saturated colors during the wet months, while bad rainfall years appear faded or purple, showing weak or absent monsoons. This variability is a defining feature of India's climate, with significant implications for agriculture, urban flooding, and drought risk.

Heatwave days per year × city

```
hw_thr <- 40

hw_year <- weather_data |>
  mutate(year = year(date),
         hw = ifelse(!is.na(tmax) & tmax >= hw_thr, 1, 0)) |>
  group_by(city, year) |>
  summarise(hw_days = sum(hw, na.rm = TRUE), .groups = "drop")

ggplot(hw_year, aes(year, hw_days, color = city)) +
  geom_point(alpha = 0.7) +
```

```r
  geom_line() +
  labs(title = paste0("Annual Heatwave Days ( ", hw_thr, "°C)"),
       x = "Year", y = "Number of heatwave days") +
  theme_minimal(base_size = 13)
```

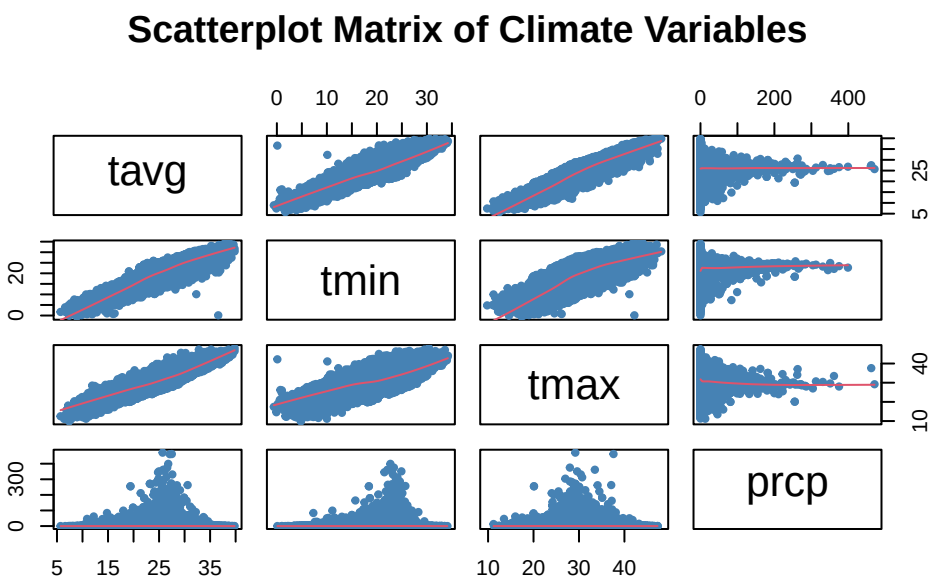


Key Findings

- **Delhi and Lucknow** record the **highest and most frequent heatwave days**, often exceeding 30–40 days annually in the 1990s and 2000s.
- **Jodhpur** also shows consistent heatwaves but with fewer extreme years compared to Delhi and Lucknow.
- **Chennai** averages fewer than 20 heatwave days per year, still showing notable peaks especially in the late 1990s and mid-2010s.
- **Mumbai and Bengaluru** remain near zero across the entire record, rarely if ever crossing the 40 °C threshold due to coastal moderation (Mumbai) and elevation (Bengaluru).
- Lucknow shows particularly high values in the 1990s, but with a slight downward drift afterwards.

Correlation Matrix of Climate Variables

```r
clim_corr <- weather_data |>
  select(tavg, tmin, tmax, prcp)

# scatterplot matrix
pairs(clim_corr,
      panel = panel.smooth,   # adds LOWESS smoother
      pch = 20, col = "steelblue",
      main = "Scatterplot Matrix of Climate Variables")
```



Key Findings

- `tavg` is **strongly linearly correlated** with both `tmin` and `tmax`.
- `tmin` and `tmax` themselves also show a positive relationship, though with slightly more scatter, reflecting day–night variability.
- These strong associations confirm internal consistency in the dataset, since `tavg` is mathematically derived from `tmin` and `tmax`.
- Scatterplots involving `prcp` show a **weak or slightly negative relationship** with temperatures.

- On days with significant rainfall, maximum temperatures tend to be lower (monsoon cooling effect).
- However, rainfall is highly variable, and many days have **zero precipitation**, leading to clusters at the bottom of the plots.
- Temperature variables are fairly continuous and spread, while precipitation is **highly skewed** — most days are dry, with occasional extreme rainfall events.

Conclusion

Together, these findings illustrate that India's climate is defined by two powerful but contrasting forces:

- A **predictable seasonal rhythm of heat and coolness** tied to geography, and
- A **highly variable monsoon system** that dictates rainfall, water resources, and agricultural outcomes.
- For inland regions, **heat stress and rainfall variability** remain the primary risks, while for coastal and plateau regions, **humidity, heavy rains, and flooding** dominate.

Question 5: Describe an example of *unethical* data visualization, discussing with a graphical illustration. The graph can be recreated in R, or downloaded from the source.

Comments: An example of unethical data visualization is using a truncated Y-axis to exaggerate differences in data and mislead viewers. This technique involves starting the vertical axis at a value other than zero, which manipulates the visual representation of the data and can lead to inaccurate conclusions.

The following example demonstrates how a truncated Y-axis can create a misleading bar chart.

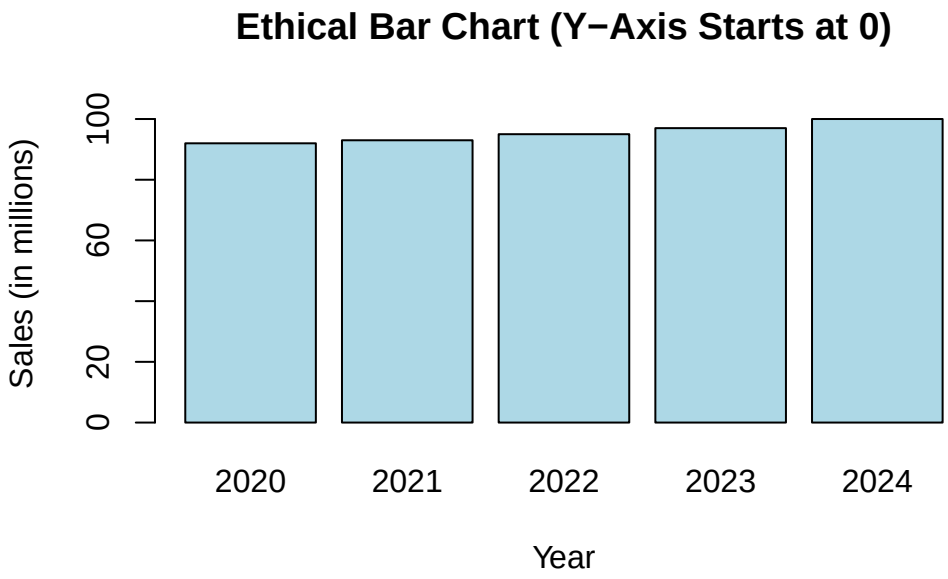
Scenario: Suppose a company wants to show that its sales have significantly increased over the past five years, even if the real growth was modest. The data for sales (in millions of dollars) is as follows:

- **2020:** 92
- **2021:** 93
- **2022:** 95
- **2023:** 97
- **2024:** 100

Ethical bar chart (starts at 0)

An ethical visualization represents the data honestly by starting the Y-axis at zero. This shows the actual, modest growth over the five-year period.

```r
# Creating the data
years <- 2020:2024
sales <- c(92, 93, 95, 97, 100)
data1 <- data.frame(years, sales)

# Create the ethical bar chart starting at 0
barplot(
  height = data1$sales,
  names.arg = data1$years,
  main = "Ethical Bar Chart (Y-Axis Starts at 0)",
  xlab = "Year",
  ylab = "Sales (in millions)",
  ylim = c(0, 105), # Set the y-axis limit to start at 0
  col = "lightblue",
  border = "black"
)
```



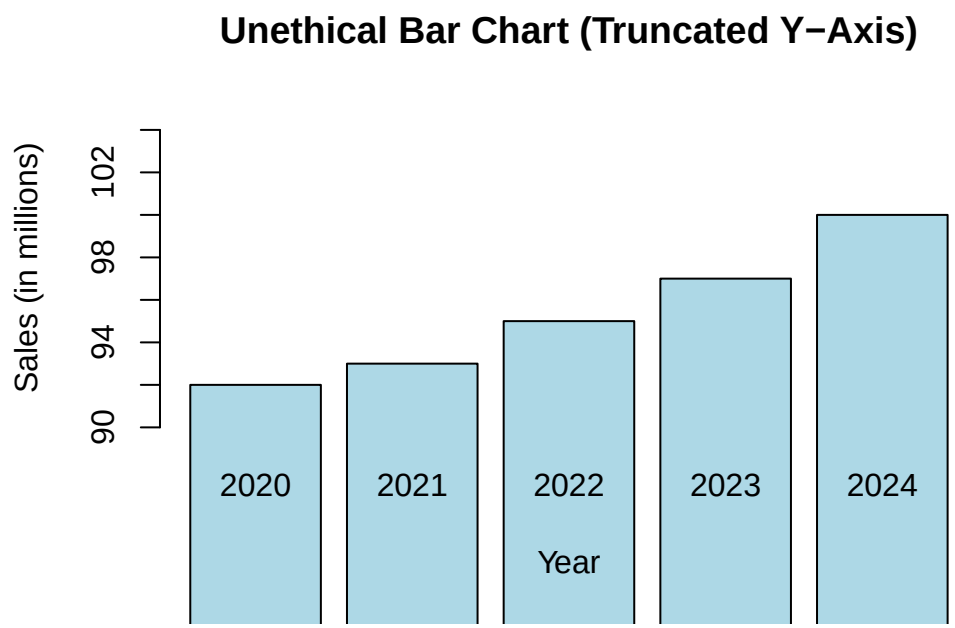
This graph correctly shows that sales have grown steadily but not dramatically. The visual representation is proportional to the actual data values, allowing for an accurate interpretation

of the company's performance.

Unethical bar chart (truncated Y-axis)

An unethical visualization truncates the Y-axis to exaggerate the perceived growth. In this case, starting the axis at 90 makes the sales increase look explosive.

```
# Creating the unethical bar chart with a truncated Y-axis
barplot(
  height = data1$sales,
  names.arg = data1$years,
  main = "Unethical Bar Chart (Truncated Y-Axis)",
  xlab = "Year",
  ylab = "Sales (in millions)",
  ylim = c(90, 105), # Truncated y-axis to exaggerate the effect
  col = "lightblue",
  border = "black"
)
```



Above graph is misleading because it distorts the visual proportions. While sales did increase, the truncated Y-axis makes the difference appear much larger than it is. The viewer may conclude that the company is experiencing massive growth, when in reality, the sales increase was only about 8.7% over the five years ($8 million increase on a base of $92 million). This intentional manipulation of the scale is a form of unethical data visualization.