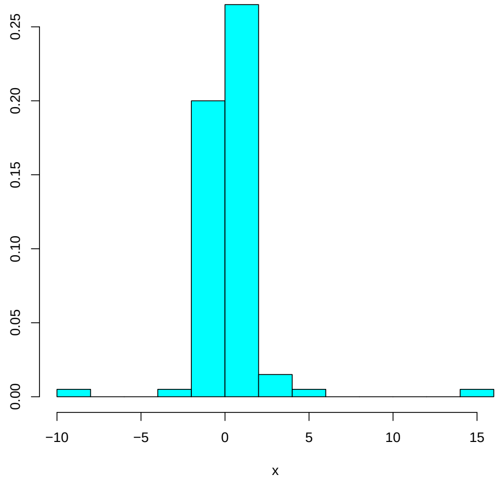
Chapter 1

Introduction

1.1 Generate a random sample $x_1, \ldots, x_{100}$ of data from the t_4 (df=4) distribution using the rt function. Use the MASS::truehist function to display a probability histogram of the sample.

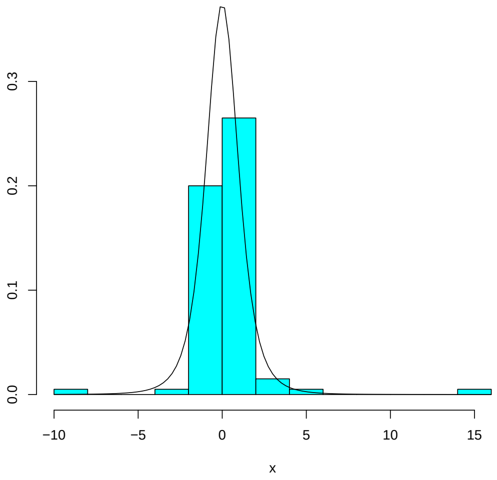
```r
library(MASS)
x <- rt(100, df = 4)
truehist(x)
```



1.2 Add the t_4 density curve (`dt`) to your histogram in Exercise 1.1 using the `curve` function with add=TRUE.

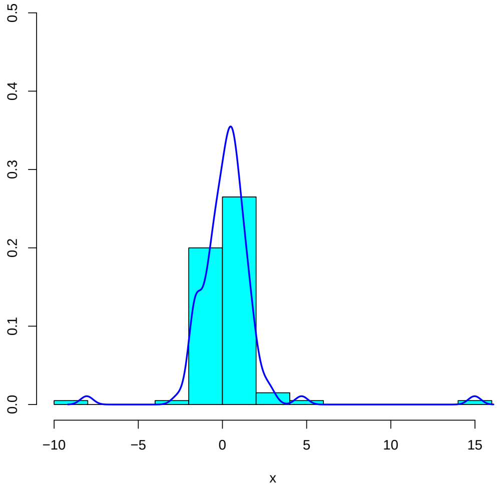
To avoid cutting off the top, we can set the y axis range to match the mode of the density at 0 using ylim in truehist.

```r
# using x from Exercise 1.1
y0 <- dt(0, df = 4)
truehist(x, ylim = c(0, y0))
curve(dt(x, df = 4), add = TRUE)
```



1.3 Add an estimated density curve to your histogram in Exercise 1.1 using `density`. Notice that the density estimate (`density`) is an approximation to the density of the sampled distribution (in this case the t_4 density). (Density estimation and the density function are covered in detail in Chapter 12.)

```r
# using x from Exercise 1.1
truehist(x, ylim = c(0, .5))
lines(density(x), col = 4, lwd = 2)
```



1.4 a. Write an R function f in R to implement the function

$$f(x) = \frac{x-a}{b}$$

that will transform an input vector x and return the result. The function should take three input arguments: x, a, b.

The return statement is optional; the last expression evaluated is returned.

```
f <- function(x, a, b) {
  return ((x - a) / b)
}

# try the function
f(10, 3, 2)

## [1] 3.5
```

b. To transform x to the interval $[0, 1]$ we subtract the minimum value and divide by the range:

```
y <- f(x, a = min(x), b = (max(x) - min(x))
```

Generate a random sample of Normal($\mu = 2, \sigma = 2$) data using `rnorm` and use your function f to transform this sample to the interval $[0, 1]$. Print a `summary` of both the sample x and the transformed sample y to check the result.

```
n <- 100
x <- rnorm(n, mean = 2, sd = 2)
y <- f(x, a = min(x), b = max(x) - min(x))

# one by one - harder to compare
summary(x)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.8001  0.4539  2.2470  1.9429  3.3832  8.2517

summary(y)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2944  0.4567  0.4292  0.5595  1.0000

# nicer for comparison to create a data frame
dat <- data.frame(x = x, y = y)
summary(dat)

##        x                  y
##  Min.   :-2.8001   Min.   :0.0000
##  1st Qu.: 0.4539   1st Qu.:0.2944
##  Median : 2.2470   Median :0.4567
##  Mean   : 1.9429   Mean   :0.4292
##  3rd Qu.: 3.3832   3rd Qu.:0.5595
##  Max.   : 8.2517   Max.   :1.0000
```

1.5 Refer to Exercise 1.4. Suppose that we want to transform the x sample so that it has mean zero and standard deviation one (*studentize* the sample). That is, we want

$$z_i = \frac{x_i - \bar{x}}{s}, \quad i = 1, \ldots, n,$$

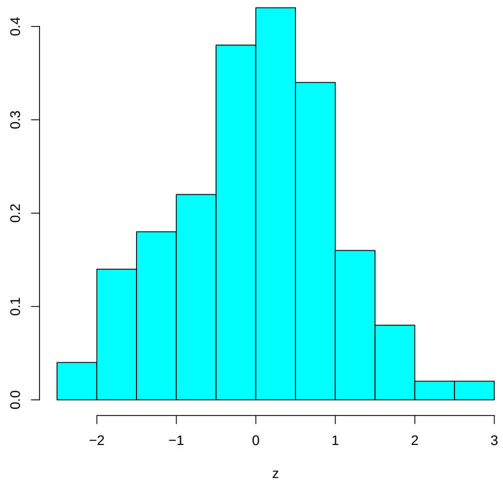
where s is the standard deviation of the sample. Using your function f this is

```
z <- f(x, a = mean(x), b = sd(x))
```

Display a summary and histogram of the studentized sample z. It should be centered exactly at zero. Use ‘sd(z)‘ to check that the studentized sample has standard deviation exactly 1.0.

```
# x sample from Exercise 1.4
z <- f(x, a = mean(x), b = sd(x))
summary(z)
```

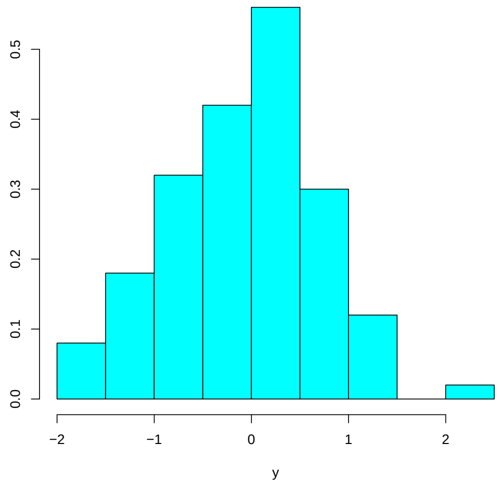
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.0890 -0.6558  0.1339  0.0000  0.6344  2.7786

truehist(z)
```



```
sd(z)

## [1] 1
```

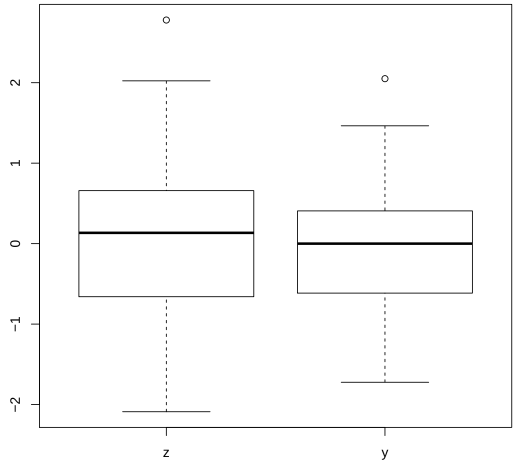
1.6 Using your function f of Exercise 1.4, center and scale your Normal($\mu = 2, \sigma = 2$) sample by subtracting the sample median and dividing by the sample interquartile range (IQR). Compare your results to Exercise 1.5.

```
# x sample from Exercise 1.4
y <- f(x, a = median(x), b = IQR(x))
summary(y)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.7229 -0.6121  0.0000 -0.1038  0.3879  2.0499

truehist(y)
```



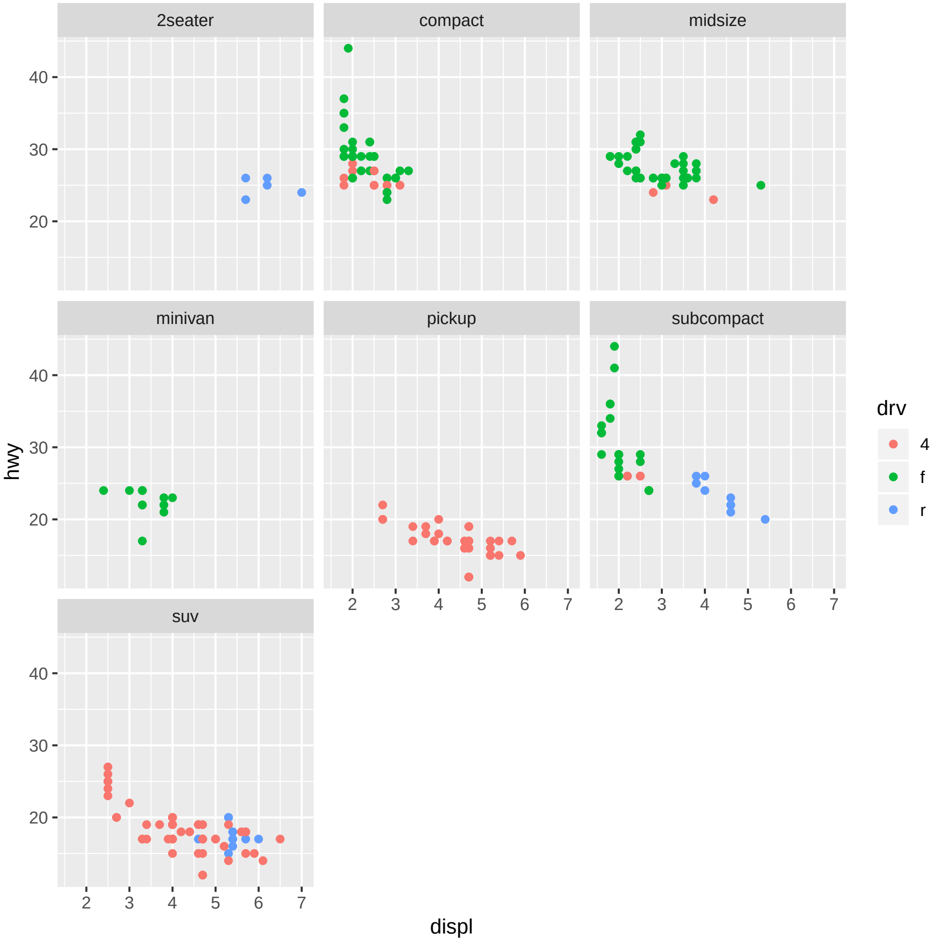
This transformation of the sample is centered close to zero with smaller range than the studentized sample. The sample median of y is zero but the sample mean of y is not zero. We can also compare with parallel boxplots.

```
df <- data.frame(z = z, y = y)
boxplot(df)
```



1.7 (ggplot) Refer to Example 1.14 where we displayed an array of scatterplots using `ggplot` with facet_wrap. One of the variables in the mpg data is drv, a character vector indicating whether the vehicle is front-wheel drive, rear-wheel drive, or four-wheel drive. Add color = drv in aes and display the revised plot. Your scatterplots should now have the three levels of drv coded by color and the plot should have automatically generated a legend for drv color.

```
library(ggplot2)

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang

ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  facet_wrap(~ class)
```



1.8 (RStudio and knitr) This exercise is intended to serve as an introduction to report writing with R Markdown. Install the *knitr* package if it is not installed. Create an html report using R Markdown and knitr in RStudio. The report should include the code and output of Examples 1.12 and 1.14 with appropriate headings and a brief explanation of each example.

- The knitr package should be installed from the Packages tab in RStudio.
- From the File menu in RStudio, select "New File" and "R Markdown ..." to open a basic template.
- Modify the title and author name.
- Replace the examples with text and code chunks for Examples 1.12 and 1.14.

- Save the file with file extension .Rmd (R Markdown).
- Click "Knit" on the editor toolbar to display the report.
- If needed, the html report is automatically saved in the current working directory.
- The report can be generated at any time from within RStudio by knitting the Rmd source file.

Note: It is not necessary to use labels in code chunks. For example, the label **cars** is not needed in

~~~
```{r cars}
summary(cars)
```
~~~

and

~~~
```{r}
summary(cars)
```
~~~

produces the same output.