Summarizing data and scaling features

Søren Højsgaard

Contents

1 Summarizing data

1.1 Example data: potatoes

Weight and size of 20 potatoes. Weight in grams; size in millimeter. There are two sizes: 'length' is the longest length and 'width' is the shortest length across a potato.

```
options("digits"=3)
dat <- doBy::potatoes
head(dat)
```

```
##   weight length width
## 1     22     38    29
## 2     41     46    34
## 3     24     40    31
## 4     16     33    28
## 5      7     25    21
## 6     40     48    35
```

```
dim(dat)
```

```
## [1] 20  3
```

```
y <- dat$weight ## grams
y
```

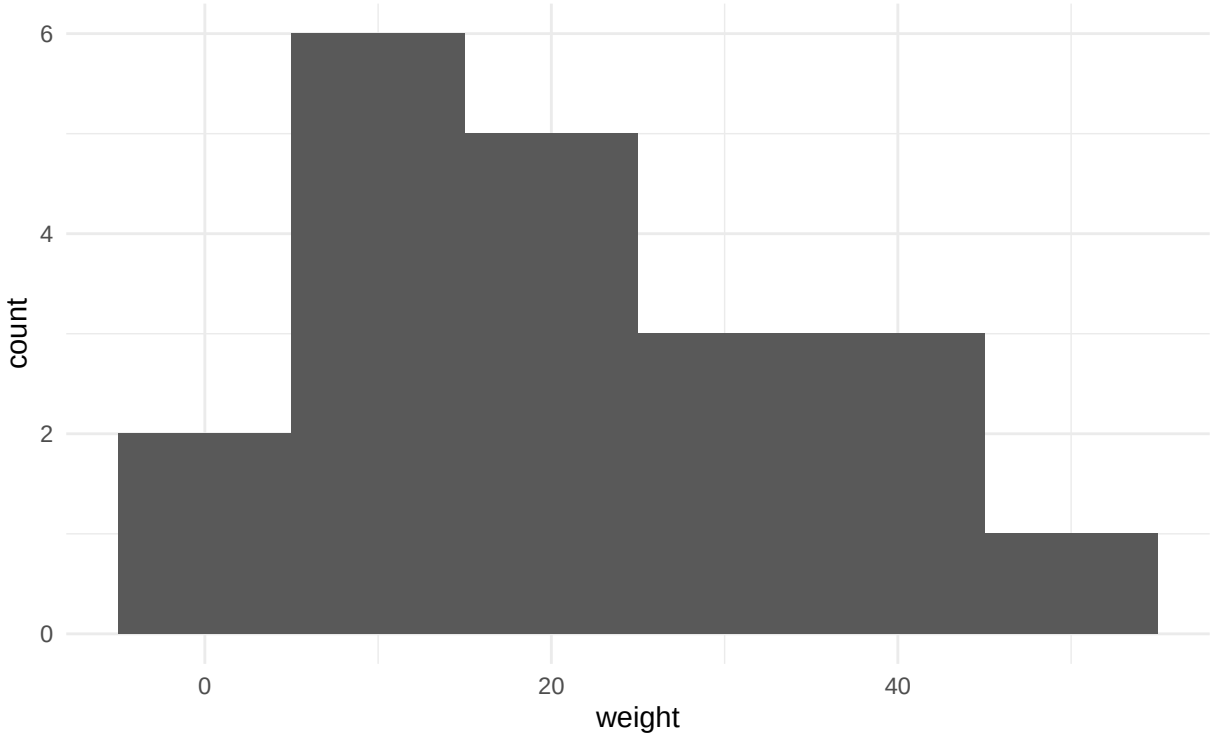
```
##  [1] 22 41 24 16  7 40 18 26  4 11  9 29 46 40  3 23  6 12 26  7
```

1.2 Looking at data

```
sort(y)
```

```
##  [1]  3  4  6  7  7  9 11 12 16 18 22 23 24 26 26 29 40 40 41 46
```

```
## hist(y) ## Default
ggplot(dat, aes(x=weight)) + geom_histogram(binwidth=10) + theme_minimal()
```



1.3 Measures of location and spread

1.3.1 A language for data

We have $n = 20$ observations in the vector y. Denote these symbolically by

$$y_1, y_2, y_3, \ldots, y_{20} \tag{1}$$

and they read `y one`, `y two` etc.

For the sum $y_1 + y_2 + y_3 + \cdots + y_{20}$ we write

$$y_\cdot = \Sigma_{i=1}^{20} y_i = y_1 + y_2 + y_3 + \cdots + y_{20} \tag{2}$$

and the left hand side reads "y dot" and $\Sigma_{i=1}^{20} y_i$ reads "the sum of y_i as i goes from 1 to n". The symbol $y_\cdot$ is of course an arbitrary name for the sum.

If we divide $y_\cdot$ by the number of observations n (here $n = 20$) we get the mean (or average). It is common to write $\bar{y}$ (pronounced "y bar") for the average:

$$\bar{y} = \frac{1}{20} y_\cdot = \frac{1}{20} \sum_{i=1}^{20} y_i \tag{3}$$

1.3.2 Measures of location

There are many different measures of location. The mean (or avarage) is the most commonly used.

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

```
mean(y) ## sum(y) / length(y) ## Same thing
```

```
## [1] 20.5
```

Notice: The unit of the mean is the same as the unit of the data, here grams.

1.3.3 Measures of spread

There are many different measures of spread. The most common measure of spread is the sample standard deviation.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

(For technical / historical) reasons we divide by $n-1$ rather than n.

Notice: The unit of the standard deviation is the same as the unit of the data, here grams.

```
sd(y) ## var(y) ## The variance
```

```
## [1] 13.5
```

The sample variance is

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{5}$$

1.3.4 Interpretation - midpoint of data and the 4sd-rule

A practical interpretation of $\bar{y}$ and s is as follows:

Suppose data is nearly symmetrical mound shaped. Then the sample mean $\bar{y}$ is close to the midpoint of the sorted data vector which is the median.

```
sort(y)
```

```
##  [1]  3  4  6  7  7  9 11 12 16 18 22 23 24 26 26 29 40 40 41 46
```

```
median(y)
```

```
## [1] 20
```

```
mean(y)
```

```
## [1] 20.5
```

Moreover, about 95% of the observations are is within two standard deviations of the mean. If we say that 95% of the observations are "practically all observation'' then practically all observations fall in this interval

Hence the largest minus the smallest value is roughly 4 standard deviations (we call this the "4sd-rule")

```
(max(y) - min(y)) / 4
```

```
## [1] 10.8
```

```
sd(y)
```

```
## [1] 13.5
```

1.4 Standardizing variables - z-scores, feature scaling

The z-score is given by

$$z_i = \frac{y_i - \bar{y}}{s} \tag{6}$$

Notice: The z-score is a dimensionless quantity: Numerator and denominator are both in grams. The z-score has mean zero and standard deviation one.

```
z <- (y - mean(y)) / sd(y)
z |> head()
```

```
## [1]  0.111  1.519  0.259 -0.334 -1.001  1.445
```

```
mean(z)
```

```
## [1] -1.52e-17
```

```
sd(z)
```

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

Suppose we change the unit from grams to kilograms. Then mean and standard deviation also changes unit.

```
y2 <- y / 1000 ## kilograms
mean(y)
```

```
## [1] 20.5
```

```
mean(y2)
```

```
## [1] 0.0205
```

```
sd(y)
```

```
## [1] 13.5
```

```
sd(y2)
```

```
## [1] 0.0135
```

However, the z-score is unaffected by change in scale and location:

```
z <- (y - mean(y)) / sd(y);
z |> head()
```

```
## [1]  0.111  1.519  0.259 -0.334 -1.001  1.445
```

```
z2 <- (y2 - mean(y2)) / sd(y2)
z2 |> head()
```

```
## [1]  0.111  1.519  0.259 -0.334 -1.001  1.445
```

Hence: mean and standard deviation depends on the scale on which the variables are measured but z-scores do not.

1.5 Covariance and correlation

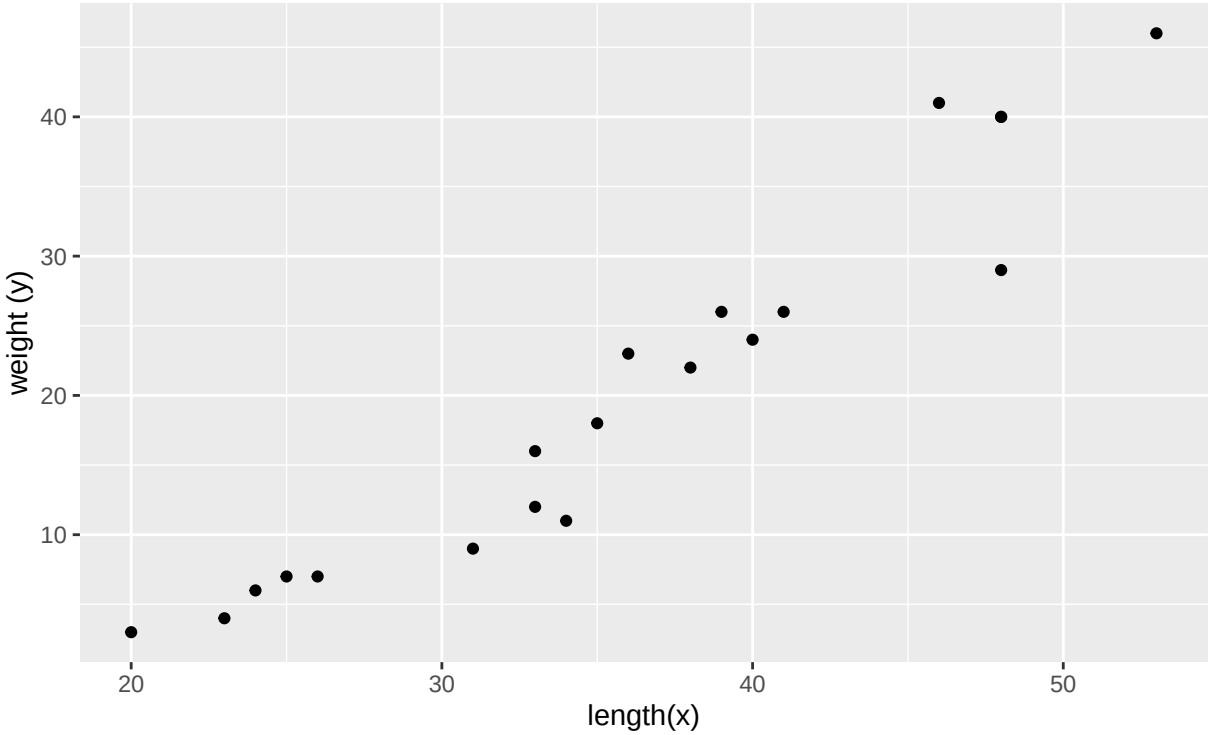
Next we include the length of potatoes. This leads to the notion of covariance and correlation.

```
x <- doBy::potatoes$length ## millimeters
x2 <- x / 10 ## centimeters
```

Hence we now write s_x and s_y to distinguish the standard deviations etc.

Cleary these measurements "co–vary''

```
## plot(x, y) ## Old style
ggplot(dat, aes(x=length, y=weight)) +
  geom_point() + labs(x = "length(x)", y = "weight (y)")
```



A measure of how they co–vary is the (sample) covariance between x and y

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{7}$$

Notice: The unit of the covariance is the product of the units of the data, here millimeters times grams. This is not very interpretable, and the covariance will change if units are changed.

Notice: If we replace y_i by x_i and $\bar{y}$ by $\bar{x}$ above then we get the sample covariance between x and x which is the (sample) variance of x.

Closely related to the covariance is the correlation coefficient:

$$\rho_{xy} = \frac{s_{xy}}{s_x s_y} \tag{8}$$

Notice: The correlation coefficient is a dimensionless quantity. In both numerator and denominator we have a product of a millimeter and a gram.

The correlation is always in the interval ± 1. If the correlation is -1 or 1 there is a perfect linear (negative/positive) association between x and y. If the correlation is 0 there is no linear association at all.

```
cov(x, y)
```

```
## [1] 123
```

```
cov(x2, y2)
```

```
## [1] 0.0123
```

Another view of a correlation is that it is the covariance between the standardized variables z_x and z_y and hence the correlation is unaffected by change in scale and location:

```
z_y <- (y-mean(y))/sd(y)
z_x <- (x-mean(x))/sd(x)
cov(z_x, z_y)
```

```
## [1] 0.961
```

```
cor(x, y)
```

```
## [1] 0.961
```

1.6 Transformation of data - more generally (optional)

Take four numbers, a, b, c and d and create new variables u_i and v_i as

$$u_i = a + bx_i, \quad v_i = c + dy_i \tag{9}$$

This corresponds to changing the scale and location of the data.

Then for the new variables we have

$$\bar{u} = a + b\bar{x}, \quad s_u = bs_x, \quad s_{uv} = bds_{xy} \tag{10}$$

But

$$z_u = z_x; \quad z_v = z_y; \quad \rho_{uv} = \rho_{xy} \tag{11}$$

Hence: mean, variance, standard deviation and covariance depends on the scale on which the variables are measured but z-scores and correlation does not.

1.7 Computations

```
cov(dat)
```

```
##        weight length width
## weight    182  123.1  83.0
## length    123   90.2  58.2
## width      83   58.2  45.9
```

```
cor(dat)
```

```
##        weight length width
## weight  1.000  0.961 0.908
## length  0.961  1.000 0.905
## width   0.908  0.905 1.000
```

```
dat2 <- dat |> scale()
dat2 |> head()
```

```
##      weight length  width
## [1,]  0.111  0.205  0.332
## [2,]  1.519  1.048  1.070
## [3,]  0.259  0.416  0.627
## [4,] -0.334 -0.321  0.185
## [5,] -1.001 -1.164 -0.849
## [6,]  1.445  1.259  1.218
```

```
dat2 |> cov()
```

```
##        weight length width
## weight  1.000  0.961 0.908
## length  0.961  1.000 0.905
## width   0.908  0.905 1.000
```

1.8 When the Unit of Measurement Matter and do Not Matter (optional)

1.8.1 When the Unit of Measurement Doesn't Matter

In statistical analysis, the scale or unit of measurement can sometimes be crucial, and in other cases, it might not matter. Here's a detailed explanation to clarify these differences:

Linear Regression Models: - The coefficients of a linear regression model reflect the change in the dependent variable for a one-unit change in the predictor variable, holding all other variables constant. While the interpretation of coefficients changes with the scale of measurement, the underlying model and the conclusions derived from it remain valid regardless of whether measurements are in Celsius or Fahrenheit, liters or gallons, etc. - However, the numerical values of the coefficients will change if the units change, but the fit of the model and the predictions will be consistent. This property makes it possible to compare models or understand relationships without worrying about the units.

1.8.2 When the Unit of Measurement Does Matter

Statistical Methods Sensitive to Scale: - In some analyses, the absolute scale of the variables matters. For example, in methods like Principal Component Analysis (PCA) or clustering, the scale can significantly impact the results because these methods rely on distances or variances which are directly influenced by the measurement units.

Standardization: - To address the issue of scale, standardization is often used. Standardization involves rescaling the data so that each variable has a mean of 0 and a standard deviation of 1. This process, often done by computing Z-scores, helps in making variables comparable. Standardizing ensures that variables contribute equally to the analysis, preventing variables with larger scales from dominating the results.

1.8.3 Examples of Analyses Requiring Standardization

1. **Principal Component Analysis (PCA)**:
   - PCA is used for dimensionality reduction. If variables are not standardized, those with larger scales can disproportionately influence the principal components, leading to biased results.
2. **Cluster Analysis**:
   - In clustering algorithms (e.g., K-means), the distance between data points is a crucial aspect. Variables with larger scales can dominate the distance metrics, skewing the clustering results.
3. **Multivariate Regression Models**:
   - In multivariate regression, having predictors on different scales can affect the stability and interpretation of the coefficients. Standardizing the predictors can help in comparing the relative importance of each predictor.

1.8.4 Summary

- **Unit-Invariant Analyses**: Linear regression models (though interpretation of coefficients depends on units).
- **Scale-Sensitive Analyses**: PCA, clustering, and multivariate regression models.
- **Solution for Scale Sensitivity**: Standardization of variables to have mean 0 and standard deviation 1.

2 A digression: More on programming in R

Consider income data from doBy package:

```
dat <- doBy::income
dat |> head()
```

```
##   inc educ race
## 1  16   10    b
## 2  18    7    b
## 3  26    9    b
## 4  16   11    b
## 5  34   14    b
## 6  22   12    b
```

We want to standardize the numerical variables to have mean 0 and standard deviation 1.

First we create a function that standardizes a vector:

```
standardize <- function(x) {
  (x - mean(x)) / sd(x)
}
z <- 1:10
standardize(z)
```

```
##  [1] -1.486 -1.156 -0.826 -0.495 -0.165  0.165  0.495  0.826  1.156  1.486
```

```
## quiz: When does this function fail?
```

We can make it more safe to use by adding a check:

```
standardize <- function(x) {
  if (is.numeric(x) && length(x) > 1) {
    (x - mean(x)) / sd(x)
  } else {
    stop("Input must be a numeric vector of length > 1")
  }
}
```

There is room for improvement. If all elements of the vector are the same, the standard deviation is zero, and we get a division by zero error. We can add a check for this:

```
standardize <- function(x) {
  if (is.numeric(x) && length(x) > 1) {
    if (sd(x) == 0) {
      stop("Standard deviation is zero")
```

```
    } else {
      (x - mean(x)) / sd(x)
    }
  } else {
    stop("Input must be a numeric vector of length > 1")
  }
}
```

We can now use this function to standardize the income data:

```
dat <- doBy::income
dat_std <- dat
for (j in 1:ncol(dat_std)) {
  if (is.numeric(dat_std[[j]])) {
    dat_std[[j]] <- standardize(dat_std[[j]])
  }
}
dat |> head()
```

```
##   inc educ race
## 1  16   10    b
## 2  18    7    b
## 3  26    9    b
## 4  16   11    b
## 5  34   14    b
## 6  22   12    b
```

```
dat_std |> head()
```

```
##      inc   educ race
## 1 -1.041 -0.936    b
## 2 -0.944 -1.981    b
## 3 -0.557 -1.284    b
## 4 -1.041 -0.588    b
## 5 -0.171  0.457    b
## 6 -0.751 -0.239    b
```

Let us wrap this into a function:

```
standardize_data <- function(dat) {
  dat_std <- dat
  for (j in 1:ncol(dat_std)) {
    if (is.numeric(dat_std[[j]])) {
      dat_std[[j]] <- standardize(dat_std[[j]])
    }
  }
  return(dat_std)
}

dat2 <- standardize_data(dat)
dat |> head()
```

```
##   inc educ race
## 1  16   10    b
## 2  18    7    b
## 3  26    9    b
## 4  16   11    b
## 5  34   14    b
## 6  22   12    b
```

```
dat2 |> head()
```

```
##      inc   educ race
## 1 -1.041 -0.936    b
## 2 -0.944 -1.981    b
## 3 -0.557 -1.284    b
## 4 -1.041 -0.588    b
## 5 -0.171  0.457    b
## 6 -0.751 -0.239    b
```

Alternative implementation (remember that a dataframe is a list)

```
out <- lapply(dat,
       function(x) {
         if (is.numeric(x)) standardize(x) else x
       })
```

```r
lapply(out, head)
```

```
## $inc
## [1] -1.041 -0.944 -0.557 -1.041 -0.171 -0.751
##
## $educ
## [1] -0.936 -1.981 -1.284 -0.588  0.457 -0.239
##
## $race
## [1] "b" "b" "b" "b" "b" "b"
```

```r
## or with a pipe
out <- dat |> lapply(function(x) {
         if (is.numeric(x)) standardize(x) else x
       })
```

```r
standardize_data <- function(dat) {
  dat |> mutate(across(where(is.numeric), standardize) )
}
standardize_data(dat)
```

```
##        inc   educ race
## 1  -1.0412 -0.936    b
## 2  -0.9445 -1.981    b
## 3  -0.5575 -1.284    b
## 4  -1.0412 -0.588    b
## 5  -0.1705  0.457    b
## 6  -0.7510 -0.239    b
## 7   0.2165  1.154    b
## 8   0.2165  1.154    b
## 9  -1.0412 -1.284    b
## 10 -0.8477 -0.936    b
## 11  1.3774  1.154    b
## 12 -0.5575 -0.239    b
## 13 -0.8477 -0.936    b
## 14 -0.3640  0.805    b
## 15 -0.8477 -0.936    b
## 16 -0.3640  2.199    b
## 17 -0.2673  1.154    h
## 18 -1.0412 -0.588    h
## 19 -0.8477 -0.936    h
## 20  0.9904  1.154    h
## 21 -0.3640 -0.239    h
## 22 -0.5575 -0.936    h
## 23 -0.8477 -1.633    h
## 24  0.1197 -0.239    h
## 25 -0.2673 -0.936    h
## 26 -0.7510 -0.588    h
## 27 -0.8477 -0.936    h
## 28  0.8937  0.457    h
## 29 -0.2673 -0.239    h
## 30 -0.3640 -0.588    h
## 31 -0.3640  0.457    w
## 32  0.5067  0.457    w
## 33  0.1197 -1.981    w
## 34  2.2481  1.850    w
## 35  0.6034 -0.936    w
## 36  0.0230 -0.239    w
## 37 -0.3640 -0.239    w
## 38  1.8611  1.154    w
## 39  0.5067  1.154    w
## 40 -0.0738 -0.588    w
## 41  0.1197 -0.588    w
## 42  0.3132 -0.239    w
## 43 -0.3640 -0.936    w
## 44  1.0872  0.805    w
## 45 -0.6542 -1.284    w
## 46  2.4416  1.502    w
## 47  0.4100  1.154    w
## 48  0.6034  1.154    w
## 49  0.6034  0.457    w
## 50 -0.7510 -0.588    w
## 51 -0.5575 -0.239    w
```

```
## 52  0.4100  1.154    w
## 53 -0.7510 -1.284    w
## 54 -0.6542 -1.284    w
## 55  1.2807  0.457    w
## 56  1.1839  1.154    w
## 57 -0.6542 -0.936    w
## 58  0.6034  0.109    w
## 59 -0.2673 -0.936    w
## 60 -0.1705  1.154    w
## 61  0.7002  1.850    w
## 62 -0.6542 -0.239    w
## 63 -0.7510  0.457    w
## 64 -0.8477  0.109    w
## 65 -0.3640  0.457    w
## 66 -0.6542  0.109    w
## 67  3.9895  1.850    w
## 68 -0.7510 -0.936    w
## 69  2.1514  1.154    w
## 70 -0.9445 -0.239    w
## 71 -0.5575 -0.239    w
## 72  3.2156  0.457    w
## 73 -0.4607 -0.239    w
## 74 -0.2673 -0.239    w
## 75  0.0230  0.457    w
## 76  0.3132 -0.239    w
## 77 -0.7510 -0.239    w
## 78 -0.9445 -0.936    w
## 79 -0.6542 -0.239    w
## 80  0.8937  2.547    w
```

2.1 Programming exercise

1. Inpired by the work above, create a function, write a function that converts income in USD to DKK (exchange rate: 1 USD = 7 DKK), and converts years of education to months of education.

2. What is the correlation between income and years of education before and after the transformation?

3. Fit a linear regression model to predict income from years of education before and after the transformation. Compare the coefficients.