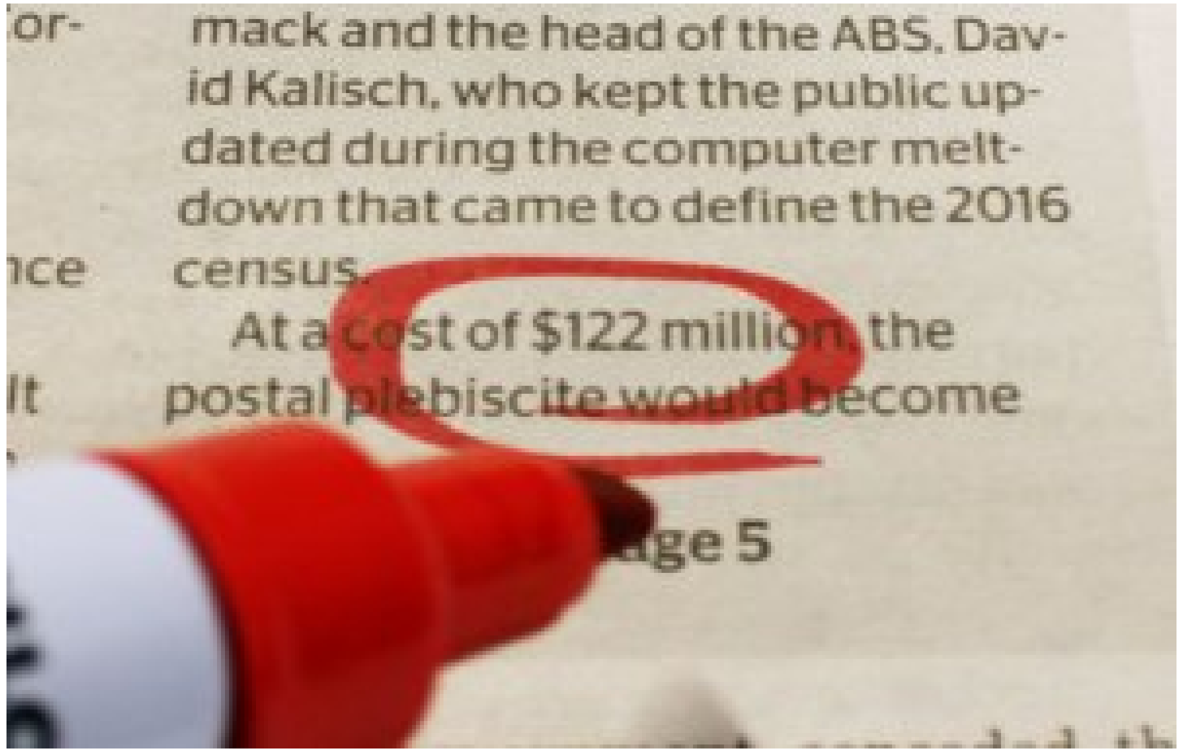
FRAUD DETECTION IN R

Digit analysis using Benford's Law

Bart Baesens
Professor Data Science at KU Leuven

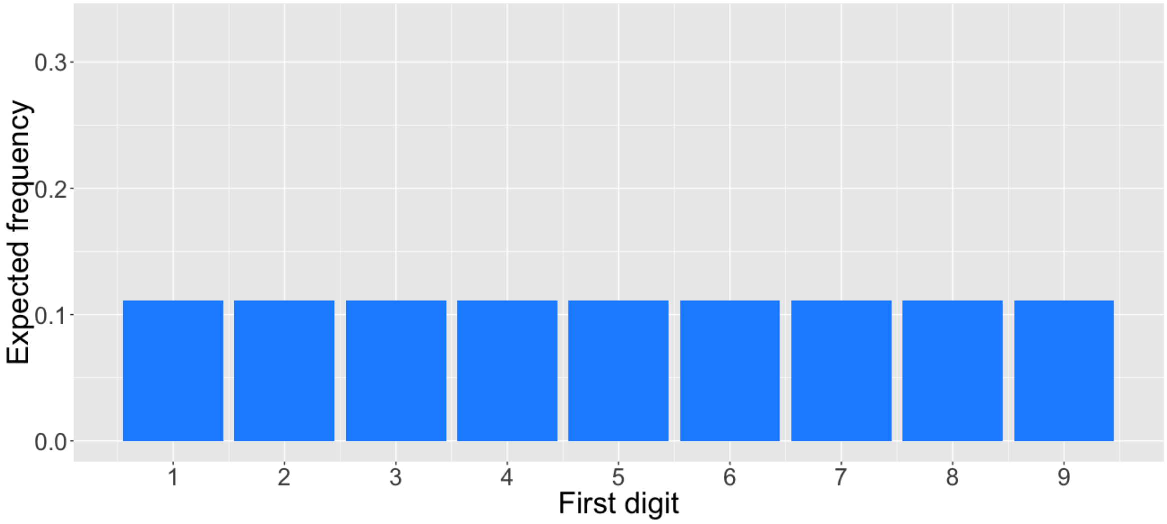
Introduction

- Take a newspaper at a random page and write down the first or leftmost digit (1,2,...,9) of all numbers.
- What are the expected frequencies of these digits?



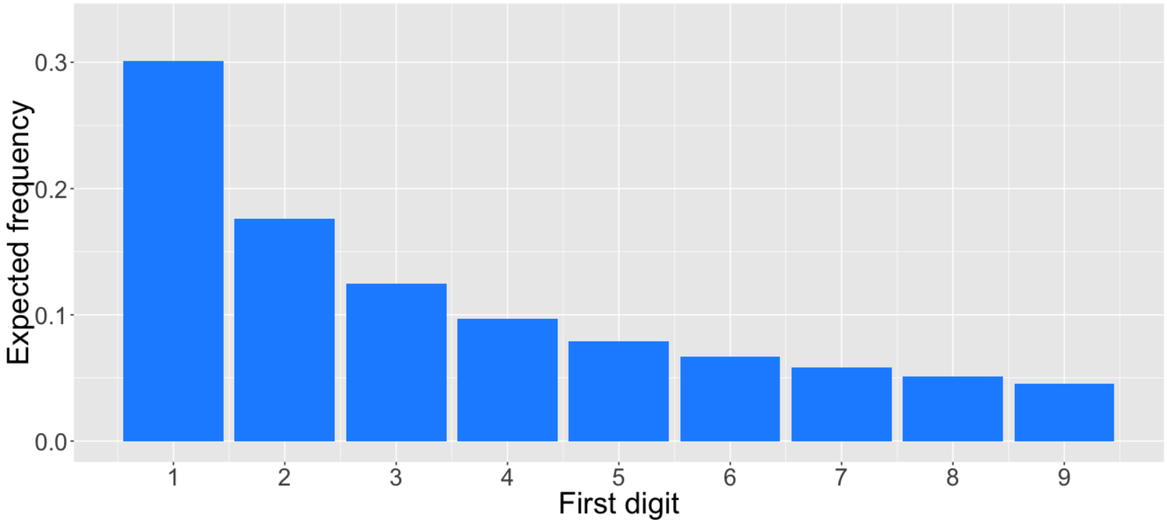
Introduction

- Take a newspaper at a random page and write down the first or leftmost digit (1,2,...,9) of all numbers.
- What are the expected frequencies of these digits?
- Natural guess will be about 1/9 = 11%



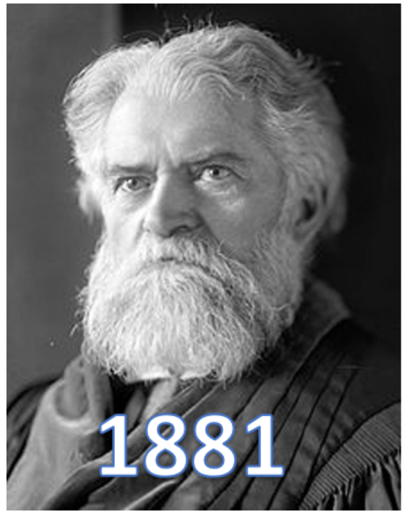
Introduction

- Take a newspaper at a random page and write down the first or leftmost digit (1,2,...,9) of all numbers.
- What are the expected frequencies of these digits?
- Natural guess will be about 1/9
- Benford's law: expected frequencies
  - digit 1 $\approx$ 30%
  - digit 9 $\approx$ 4.6%



Newcomb and Benford

- "That the ten digits do not occur with equal frequency must be evident to any one making much use of logarithmic tables, and noticing how much faster the first pages wear out than the last ones." (Newcomb, 1881)
- Benford observed the first digit of numbers in 20 different datasets.



Benford's law for the first digit

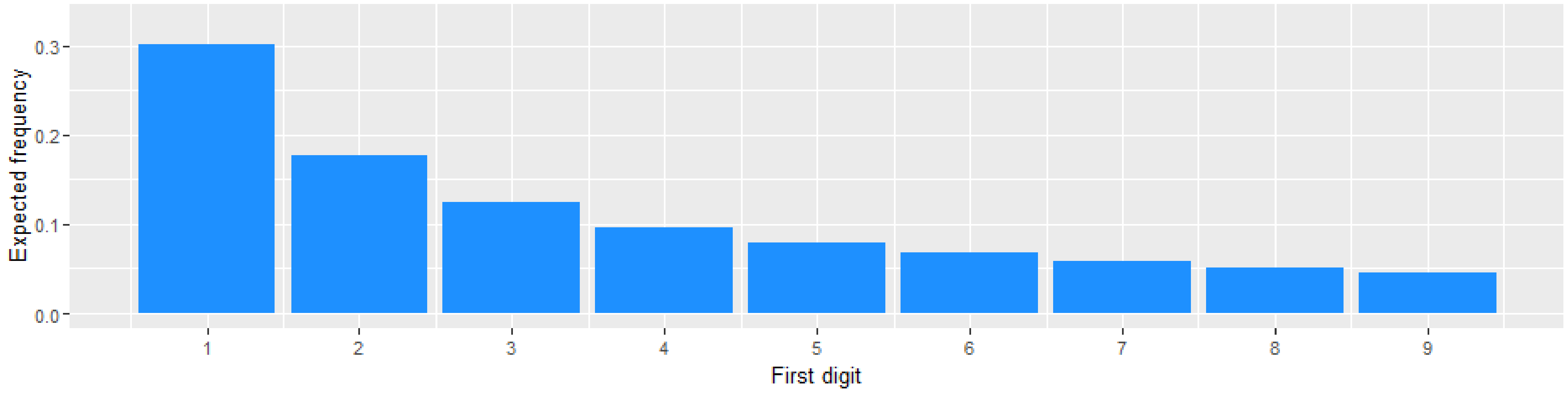
A dataset satisfies Benford's Law for the first digit if the probability that the first digit D_1 equals d_1 is approximately:

$$P(D_1 = d_1) = \log(d_1 + 1) - \log(d_1) = \log\left(1 + \frac{1}{d_1}\right) \qquad d_1 = 1, \ldots, 9$$

- Examples
  - $P(D_1 = 1) = \log\left(1 + \frac{1}{1}\right) = \log(2) = 0.3010300$
  - $P(D_1 = 2) = \log\left(1 + \frac{1}{2}\right) = \log(1.5) = 0.1760913$
  - $P(D_1 = 9) = \log\left(1 + \frac{1}{9}\right) = \log(1.111111) = 0.04575749$
- Pinkham discovered that Benford's law is invariant by scaling.

Benford's law for the first digit

```
benlaw <- function(d) log10(1 + 1 / d)
benlaw(1)
[1] 0.30103

df <- data.frame(digit = 1:9, probability = benlaw(1:9))
ggplot(df, aes(x = digit, y = probability)) +
geom_bar(stat = "identity", fill = "dodgerblue") +
xlab("First digit") + ylab("Expected frequency") +
scale_x_continuous(breaks = 1:9, labels = 1:9) +
ylim(0, 0.33) + theme(text = element_text(size = 25))
```



Generating Fibonacci numbers and powers of 2

The Fibonacci sequence is characterized by the fact that every number after the first two is the sum of the two preceding ones. We generate first 1000 Fibonacci numbers.

```
n <- 1000
fibnum <- numeric(len)
fibnum[1] <- 1
fibnum[2] <- 1
for (i in 3:n) {
  fibnum[i] <- fibnum[i-1]+fibnum[i-2]
}
head(fibnum)
[1] 1 1 2 3 5 8
```

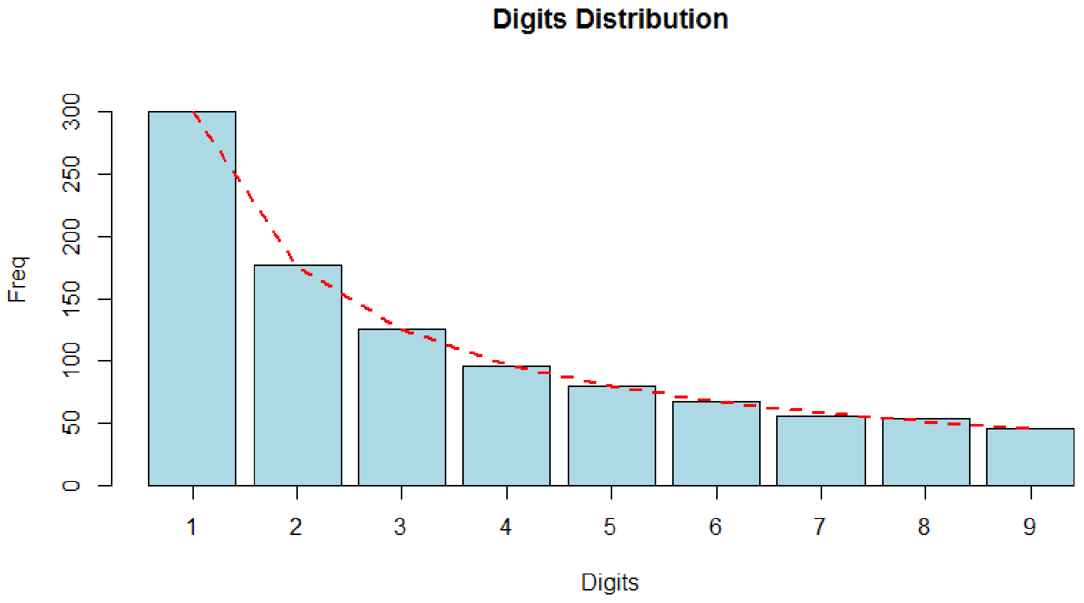
We also generate the first 1000 powers of 2

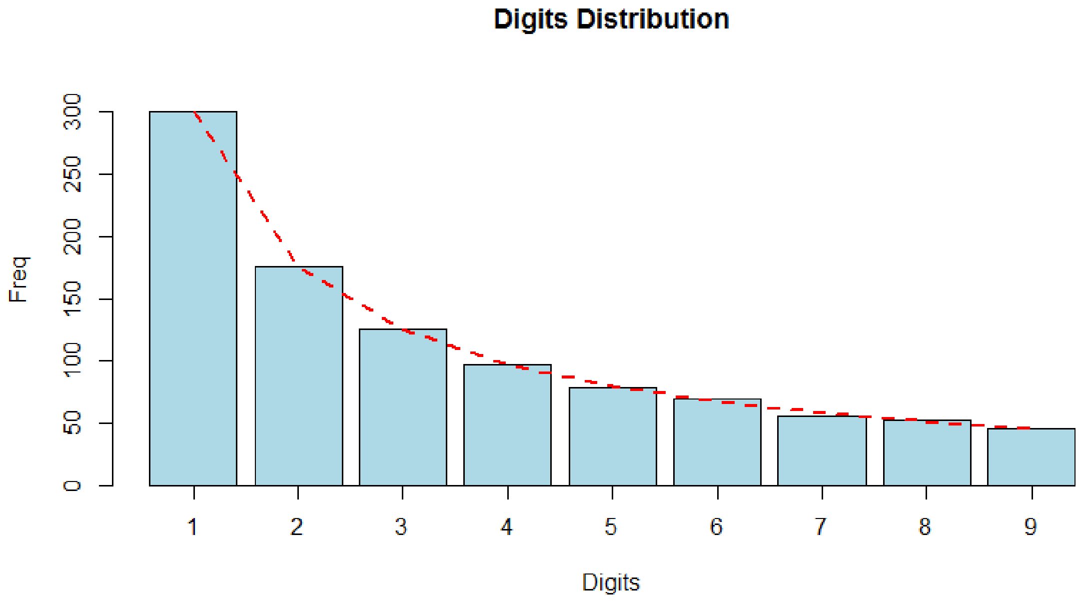
```
pow2 <- 2^(1:n)
head(pow2)
[1] 2 4 8 16 32 64
```

Investigating conformity using package benford.analysis

```
library(benford.analysis)
bfd.fib <- benford(fibnum,
                   number.of.digits = 1)
plot(bfd.fib)
```

```
library(benford.analysis)
bfd.pow2 <- benford(pow2,
                    number.of.digits = 1)
plot(bfd.pow2)
```





FRAUD DETECTION IN R

Let's practice!

FRAUD DETECTION IN R

Benford's Law for fraud detection

Bart Baesens
Professor Data Science at KU Leuven

Many datasets satisfy Benford's Law

- data where numbers represent **sizes of facts or events**
- data in which numbers have **no relationship to each other**
- data sets that **grow exponentially** or arise from **multiplicative fluctuations**
- **mixtures** of different data sets
- Some well-known infinite **integer sequences**

Preferably, **more than 1000** numbers that go across **multiple orders**.

For example

- accounting transactions
- credit card transactions
- customer balances
- death rates
- diameter of planets
- electricity and telephone bills
- Fibonacci numbers
- incomes
- insurance claims
- lengths and flow rates of rivers
- loan data
- numbers of newspaper articles
- physical and mathematical constants
- populations of cities
- powers of 2
- purchase orders
- stock and house prices
- ...

Benford's Law for fraud detection

- Fraud is typically committed by adding **invented numbers or changing real observations**.
- Benford’s Law is **popular tool for fraud detection** and is even **legally admissible as evidence in the US**.
- It has for example been successfully applied for claims fraud, check fraud, electricity theft, forensic accounting and payments fraud.
- See also the book *Benford's Law: Applications for forensic accounting, auditing, and fraud detection* of Nigrini (John Wiley & Sons, 2012).

Be careful

Note that it is always possible that data does just not conform to Benford's Law.

- If there is **lower** and/or **upper bound** or data is concentrated in **narrow interval**, e.g. hourly wage rate, height of people.
- If numbers are used as **identification numbers** or labels, e.g. social security number, flight numbers, car license plate numbers, phone numbers.
- **Additive fluctuations** instead of multiplicative fluctuations, e.g. heartbeats on a given day

Benford's Law for the first-two digits

A dataset satisfies Benford's Law for the **first-two digits** if the probability that the first-two digits D_1D_2 equal d_1d_2 is approximately:

$$P(D_1D_2 = d_1d_2) = \log\left(1 + \frac{1}{d_1d_2}\right) \qquad d_1d_2 \in [10, 11, ..., 98, 99]$$

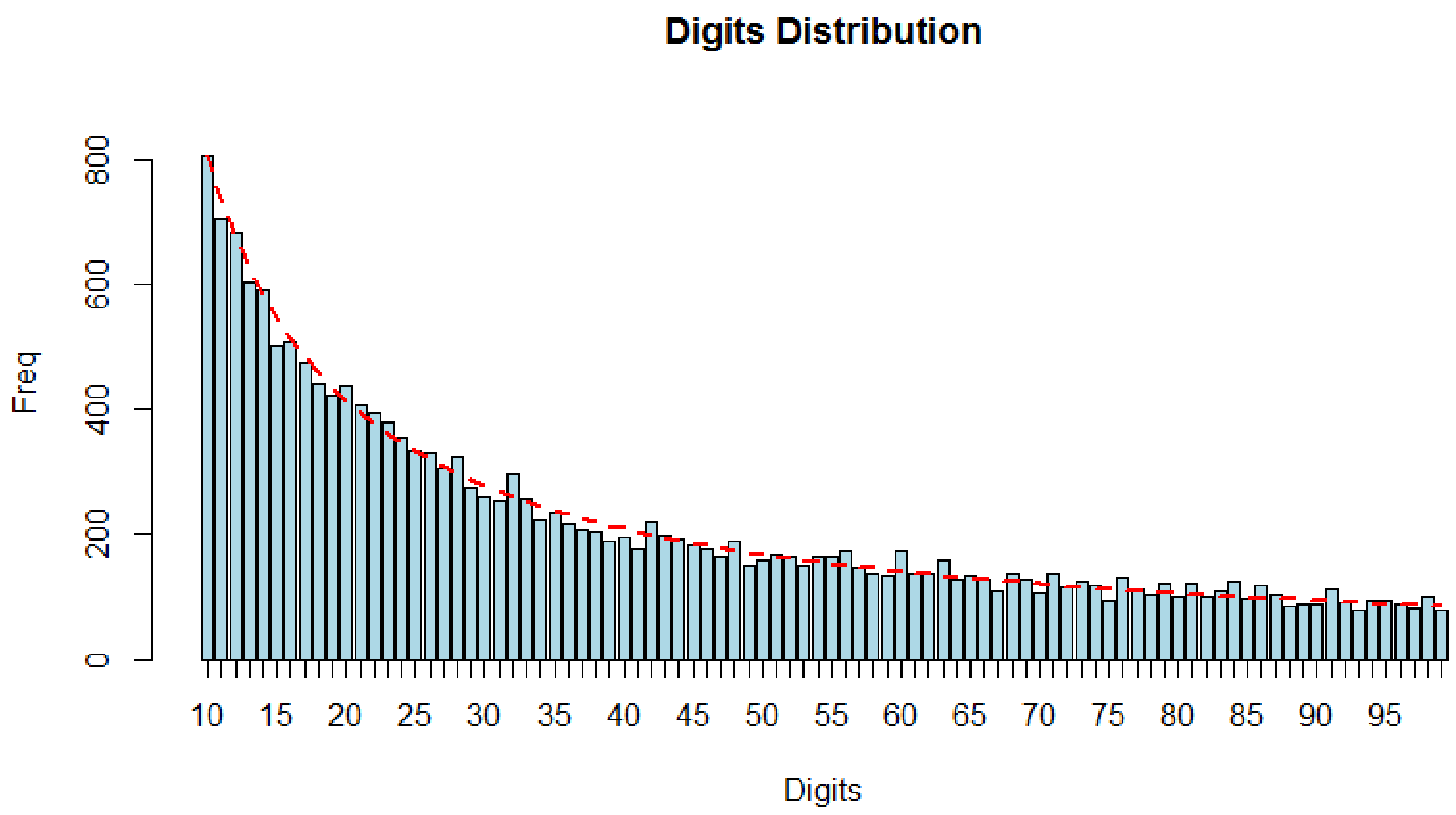
Note that we have already implemented this function in R.

```
benlaw <- function(d) log10(1 + 1 / d)
benlaw(12)
[1] 0.03476211
```

This test is more reliable than the first digits test and is most frequently used in fraud detection.

Census data

```
bfd.cen <- benford(census.2009$pop.2009,number.of.digits = 2)
plot(bfd.cen)
```

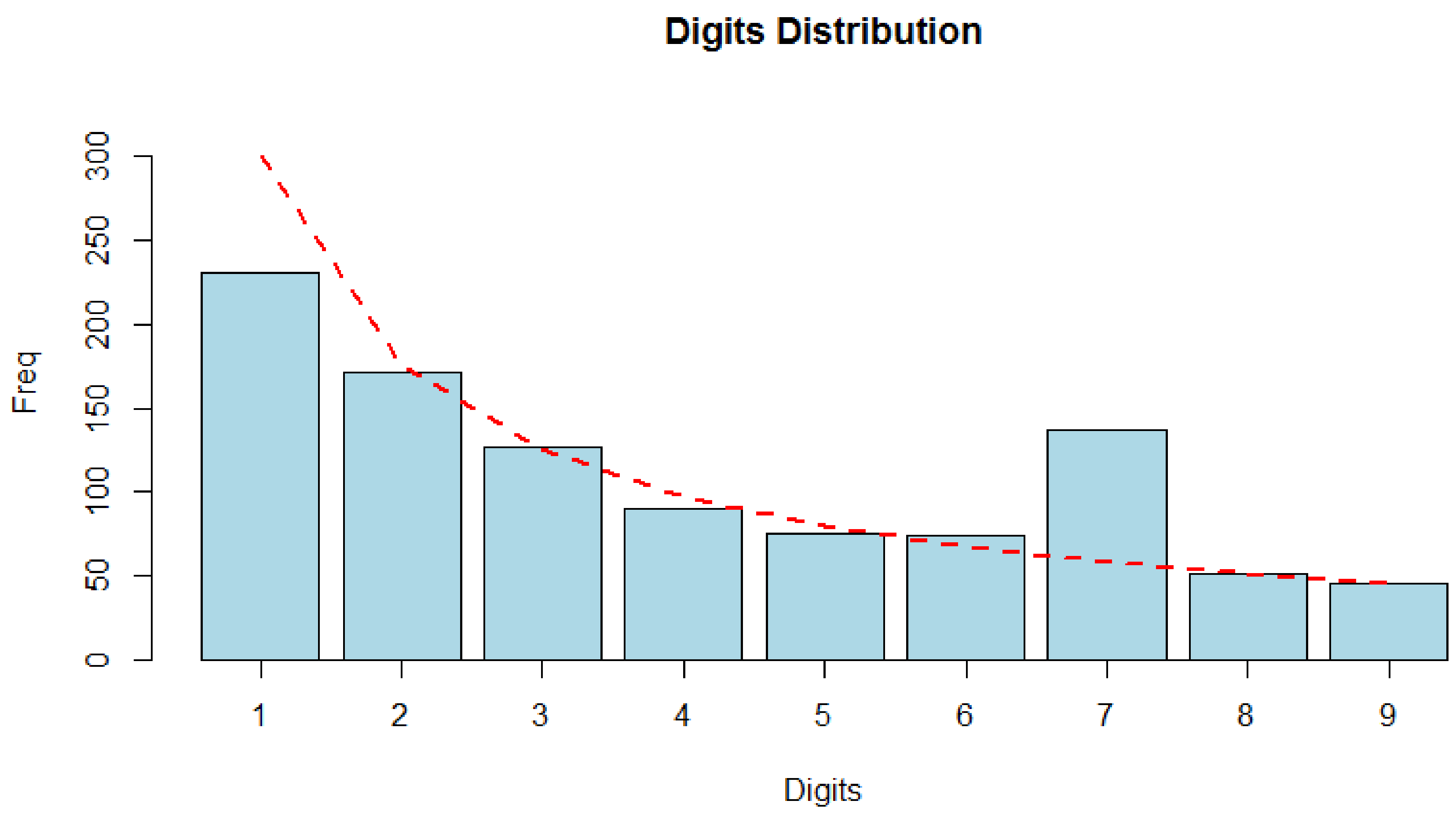


Employee reimbursements

- Internal audit department need to check employee reimbursements for fraud.
- Employees may reimburse business meals and travel expenses after mailing scanned images of receipts.
- Let us analyze the amounts that were reimbursed to employee Sebastiaan in the last 5 years.
- Dataset `expenses` contains 1000 reimbursements.
- We will use again the function included in package `benford.analysis`.

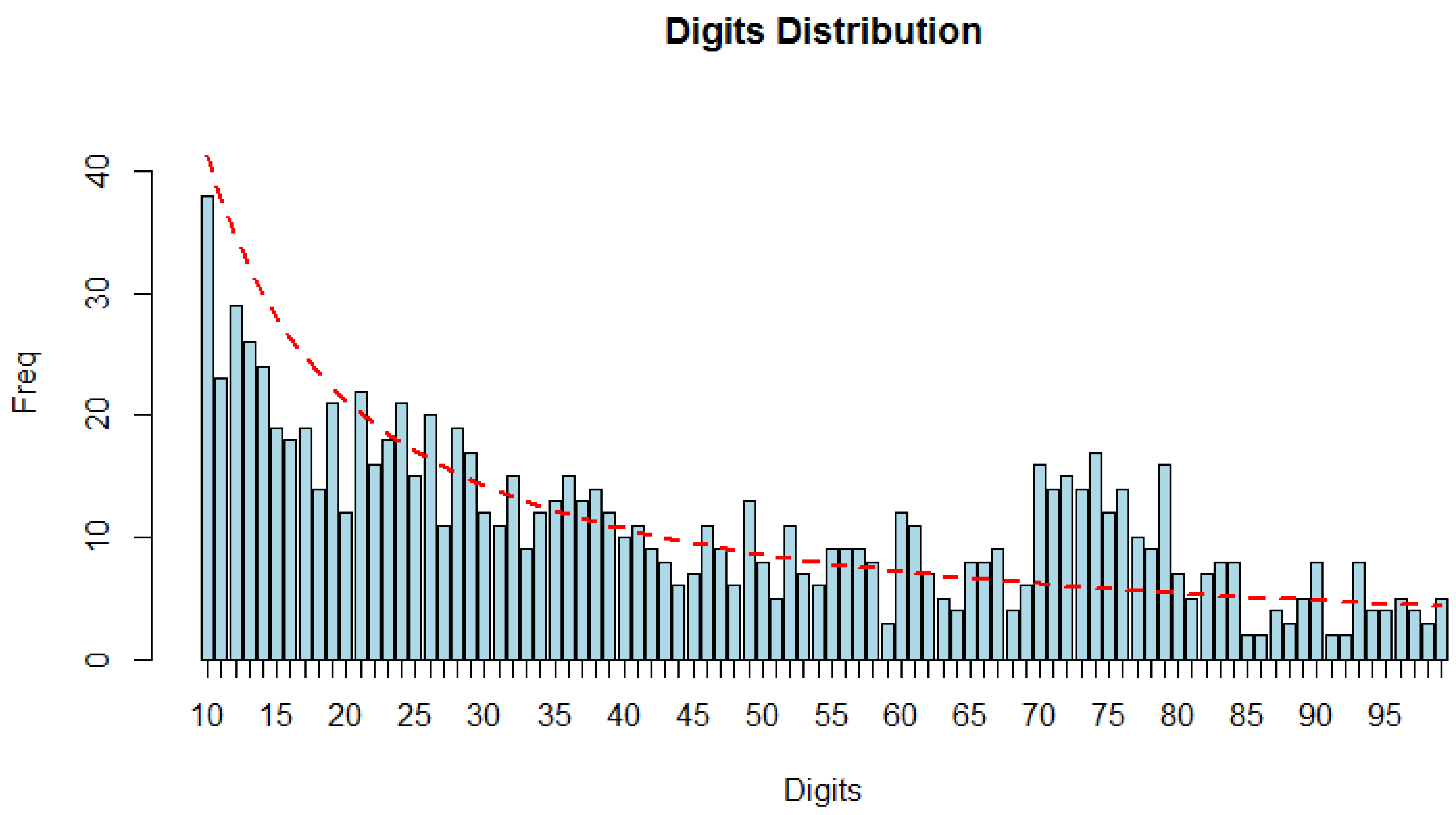
Analysis with Benford's Law for first digit

```
bfd1.exp <- benford(expenses,  number.of.digits = 1)
plot(bfd1.exp)
```



Analysis with Benford's Law for first-two digits

```
bfd2.exp <- benford(expenses, number.of.digits = 2)
plot(bfd2.exp)
```



FRAUD DETECTION IN R

Let's practice!

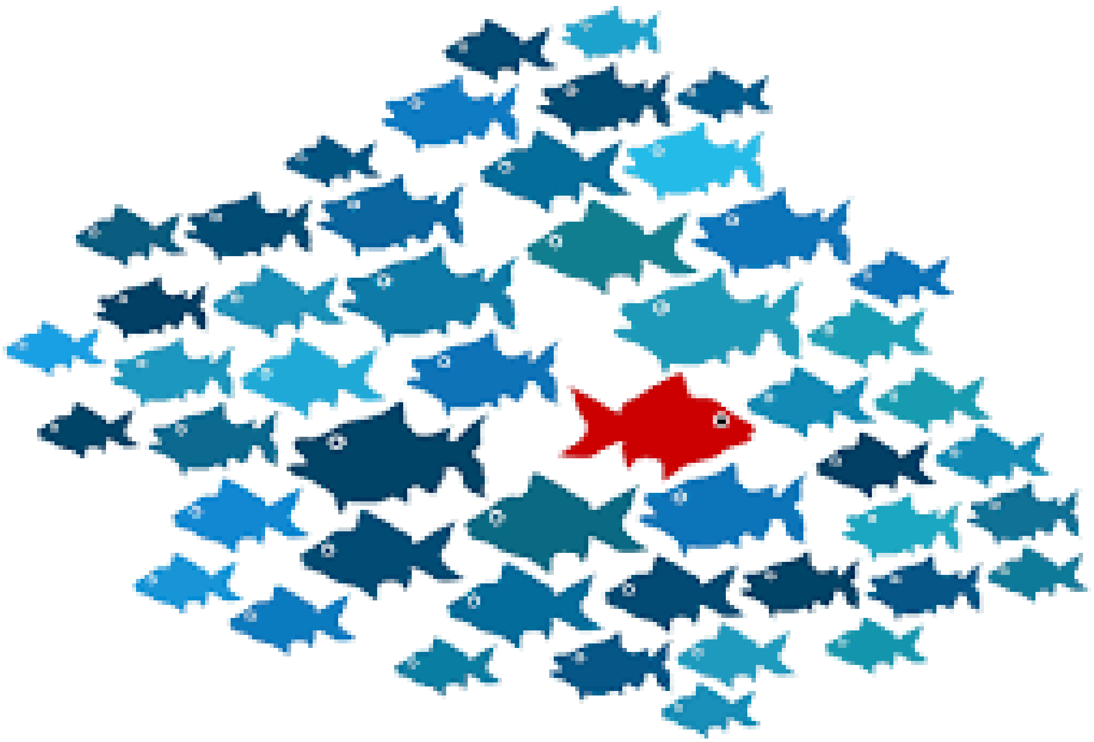
FRAUD DETECTION IN R

Detecting univariate outliers

Tim Verdonck
Professor Data Science at KU Leuven

Outliers

An outlier is an observation that deviates from the pattern of the majority of the data.



An outlier can be a warning for fraud.

Outlier detection

- A **popular tool for outlier detection** is
  - to calculate **z-score** for each observation
  - flag observation as outlier if its z-score has **absolute value greater than 3**.
- The z-score z_i for observation x_i is calculated as:

$$z_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}} = \frac{x_i - \overline{x}}{s}$$

- $\overline{x}$ is the **sample mean**: $\overline{x} = \frac{1}{n} \sum_i x_i$
- s is **sample standard deviation**: $s = \sqrt{\frac{1}{n-1} \sum_i (x_i - \hat{\mu})^2}$

Example

- Dataset `loginc` contains monthly incomes of 10 persons after log transformation

```
loginc
 [1] 7.876638 7.681560 7.628518  ...  7.764296 9.912943
```

- The last observation is clearly outlying
- Compute the z-score of each observation

```
Mean <- mean(loginc)
Sd <- sd(loginc)
zscore <- abs((loginc - Mean)/Sd)
```

- Check whether they are larger than 3 in absolute value

```
abs(zscore) > 3
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

- No outliers are identified using z-scores.

Robust statistics

- Classical statistical methods rely on (normality) assumptions, but even single outlier can influence conclusions significantly and may lead to misleading results.
- Robust statistics produce also reliable results when data contains outliers and yield automatic outlier detection tools.
- "*It is perfect to use both classical and robust methods routinely, and only worry when they differ enough to matter... But when they differ, you should think hard.*" J.W. Tukey (1979)

Estimators of location for X_n

Sample mean:

$$\overline{x} = \frac{1}{n}\sum_i x_i$$

```
mean(loginc)
[1] 7.986447
mean(loginc9)
[1] 7.772392
```

Order n observations from small to large, then **sample median**, $Med(X_n)$, is $(n+1)/2$th observation (if n is odd) or average of $n/2$th and $n/2+1$th observation (if n is even).

```
median(loginc)
[1] 7.816658
median(loginc9)
[1] 7.764296
```

`loginc9` contains same observations as `loginc` except for the outlier.

Estimators of scale

Sample standard deviation:

$$s = \sqrt{\frac{1}{n-1}\sum_i (x_i - \hat{\mu})^2}$$

```
> sd(loginc)
[1] 0.6976615
> sd(loginc9)
[1] 0.1791729
```

Median absolute deviation:

$$Mad(X_n) = 1.4826 Med(|x_i - Med(X_n)|)$$

Interquantile range (normalized):

$$IQR(X_n) = IQR = 0.7413(Q_3 - Q_1)$$

where Q_1 and Q_3 are first and third quartile of the data.

```
> mad(loginc)
[1] 0.2396159
> mad(loginc9)
[1] 0.201305

> IQR(loginc)/1.349
[1] 0.2056784
> IQR(loginc9)/1.349
[1] 0.1839295
```

Robust z-scores for outlier detection

- We plug in the robust estimators to compute **robust z-scores**:

$$z_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}} = \frac{x_i - Med(X_n)}{Mad(X_n)}$$

```
Med <- median(loginc)
Mad <- mad(loginc)
robzscore <- abs((loginc - Med) / Mad)
```

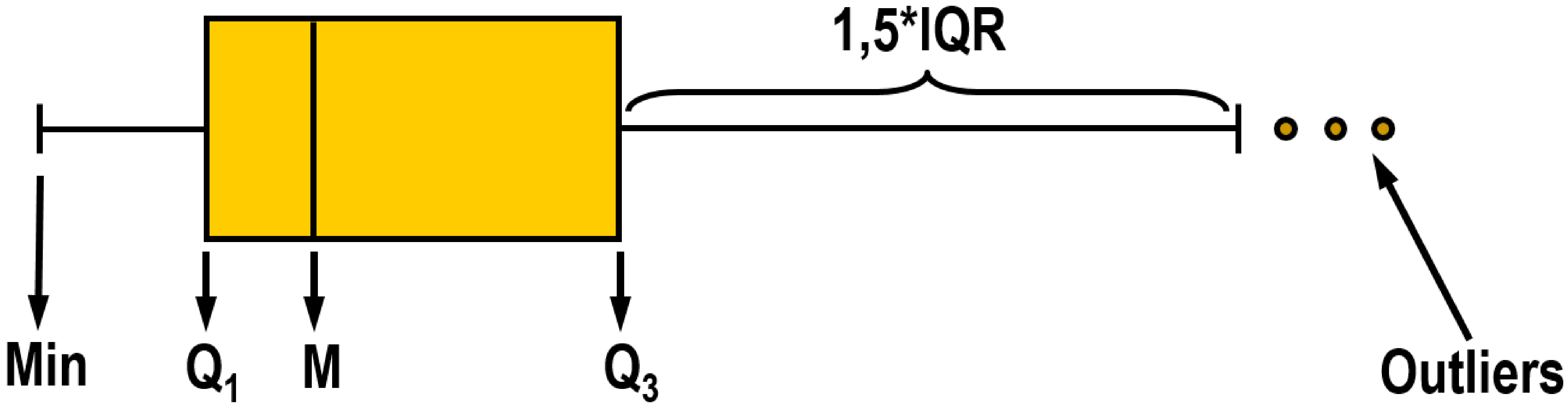
- Check for outliers

```
abs(robzscore) > 3
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

which(abs(robzscore) > 3)
[1] 10

robzscore[10]
[1] 8.748523
```

Boxplot

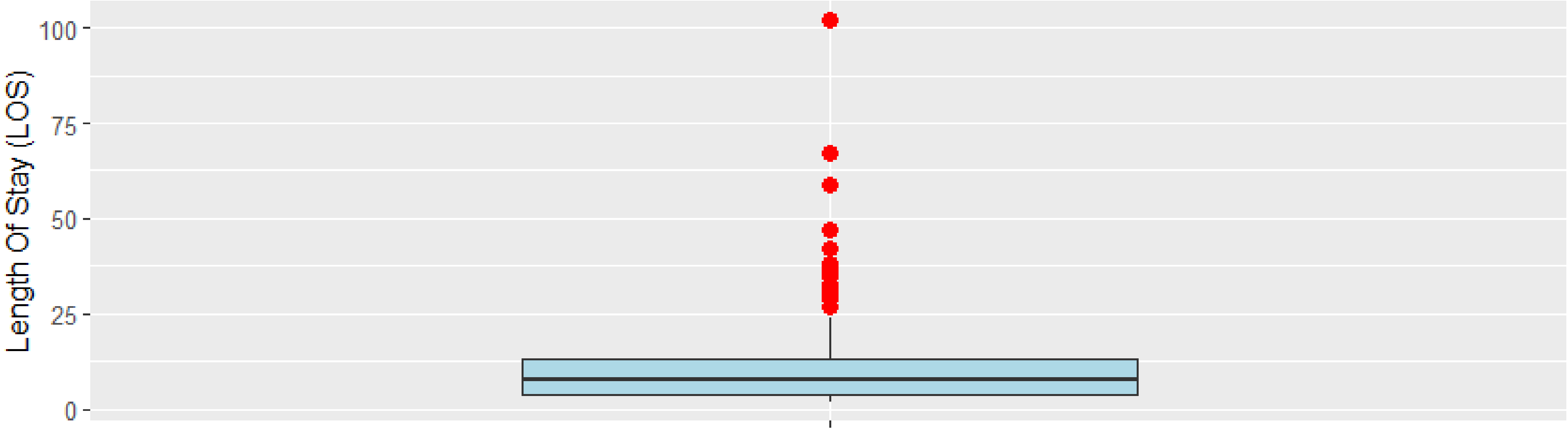
- Tukey's boxplot is also popular tool to identify outliers
- Observation is flagged as outlier if it outside the boxplot fence

$$[Q_1 - 1.5IQR; Q_3 + 1.5IQR]$$



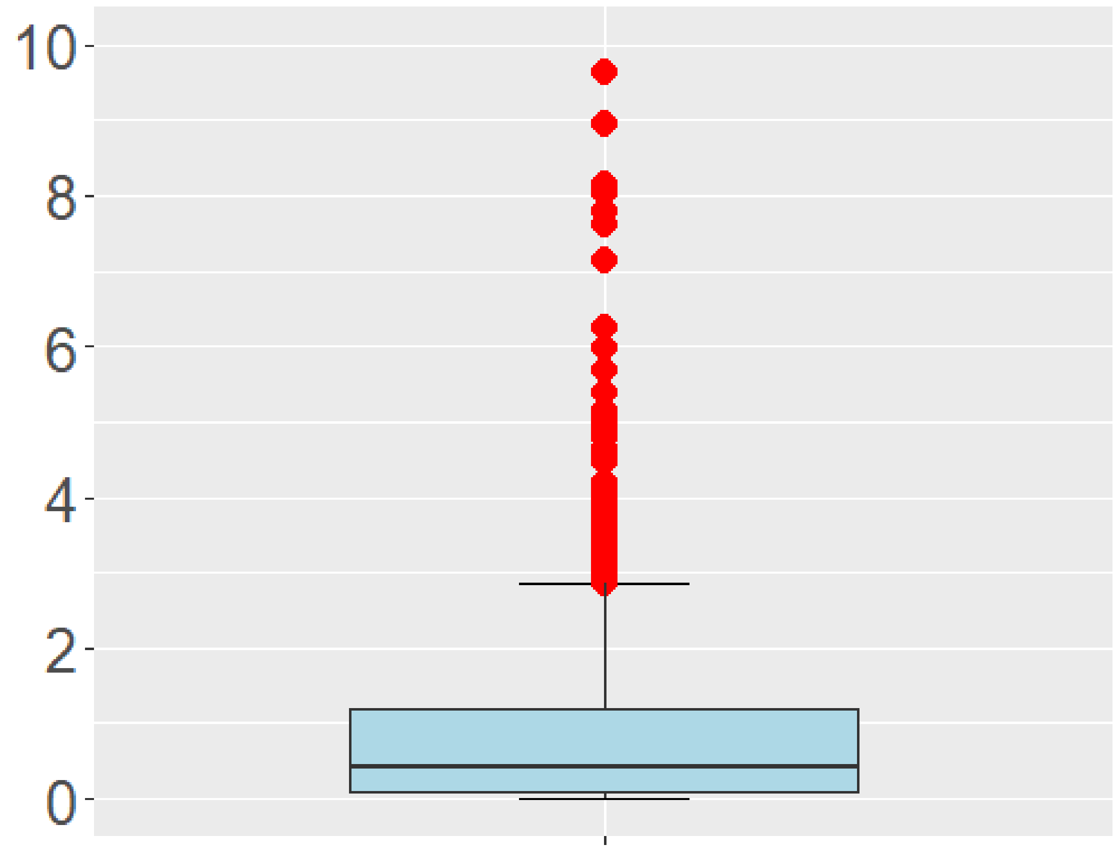
Example: length of stay (LOS) in hospital

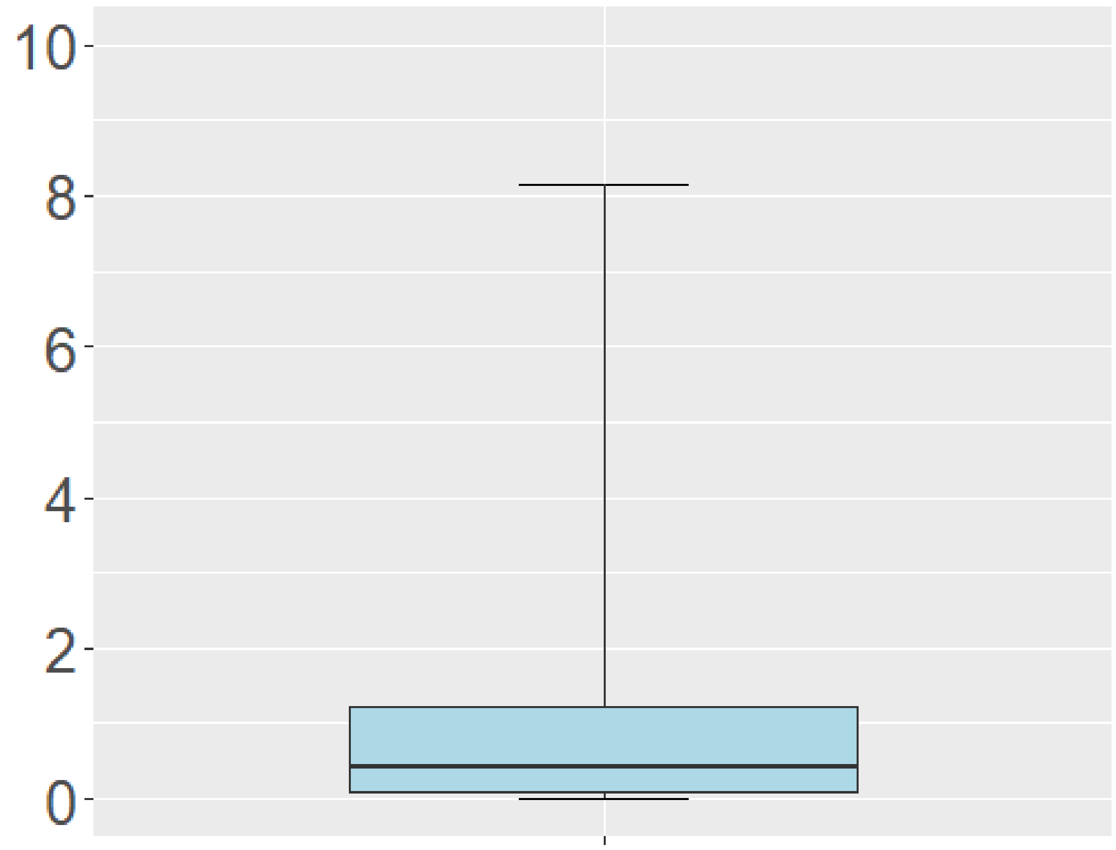
```
library(ggplot2)
ggplot(data.frame(los), aes(x = "", y = los)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 3,
               fill = "lightblue", width = 0.5) +
  xlab("") + ylab("Length Of Stay (LOS)") +
  theme(text = element_text(size = 25))

boxplot(los,col="blue",ylab="LOS data")$out
[1]  59  33  42  67  35  47 102  36  27  31  27  30  29  32  37  27  38
```



Adjusted boxplot (Hubert and Vandervieren, 2008)

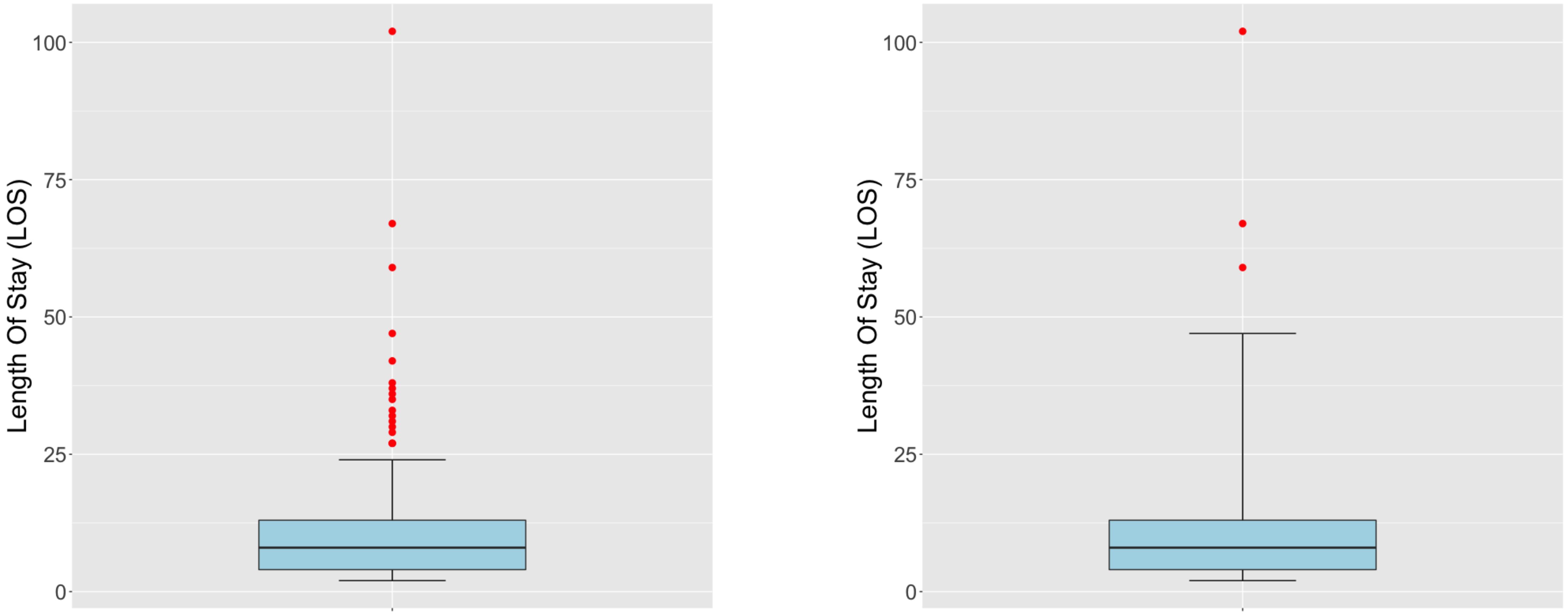
- At **asymmetric** distributions, boxplot may flag many regular points as outliers.
- The **skewness-adjusted boxplot** corrects for this by using a robust measure of skewness in determining the fence.





```
library(robustbase)
adjbox_stats <- adjboxStats(los)$stats

ggplot(data.frame(los), aes(x = "", y = los)) +
  stat_boxplot(geom = "errorbar", width = 0.2, coef = 1.5*exp(3*mc(los))) +
  geom_boxplot(ymin   = adjbox_stats[1],
               ymax   = adjbox_stats[5],
               middle = adjbox_stats[3],
               upper  = adjbox_stats[4],
               lower  = adjbox_stats[2],
               outlier.shape = NA,
               fill  = "lightblue",
               width = 0.5) +
  geom_point(data=subset(data.frame(los),
             los < adjbox_stats[1] | los > adjbox_stats[5]),
             col = "red", size = 3, shape = 16) +
  xlab("") + ylab("Length Of Stay (LOS)") +
  theme(text = element_text(size = 25))

adjbox(los,col="lightblue", ylab="LOS data")$out
[1]  59  67 102
```

Example LOS: boxplot vs adjusted boxplot



FRAUD DETECTION IN R

Let's practice!

FRAUD DETECTION IN R

Detecting multivariate outliers

Tim Verdonck
Professor Data Science at KU Leuven

Animals data

- We focus on the `Animals` dataset (in package `MASS`), containing the **average brain and body weights for 28 species** of land animals.

```
library(MASS)
data("Animals")

head(Animals)

                    body brain
Mountain beaver     1.35   8.1
Cow               465.00 423.0
Grey wolf          36.33 119.5
Goat               27.66 115.0
Guinea pig          1.04   5.5
```

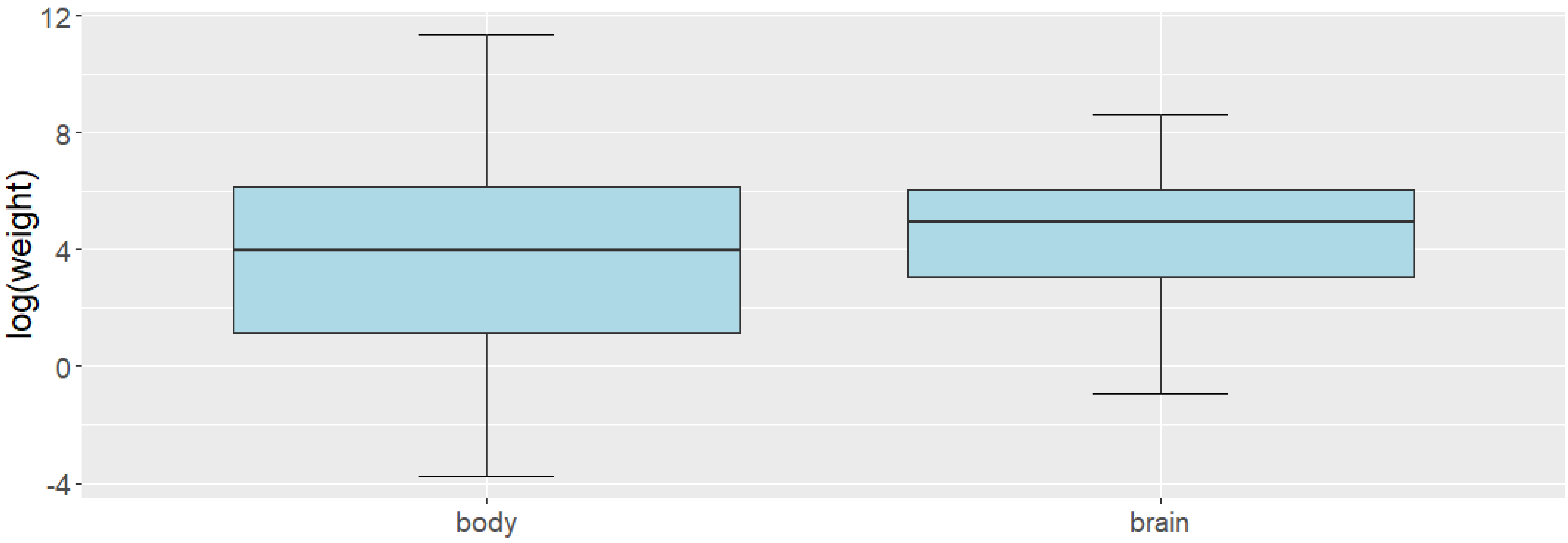
- We apply a **logarithmic transformation** on both body and brain weight .

```
X <- cbind(log(Animals$body), log(Animals$brain))
```

Animals data: univariate outlier detection

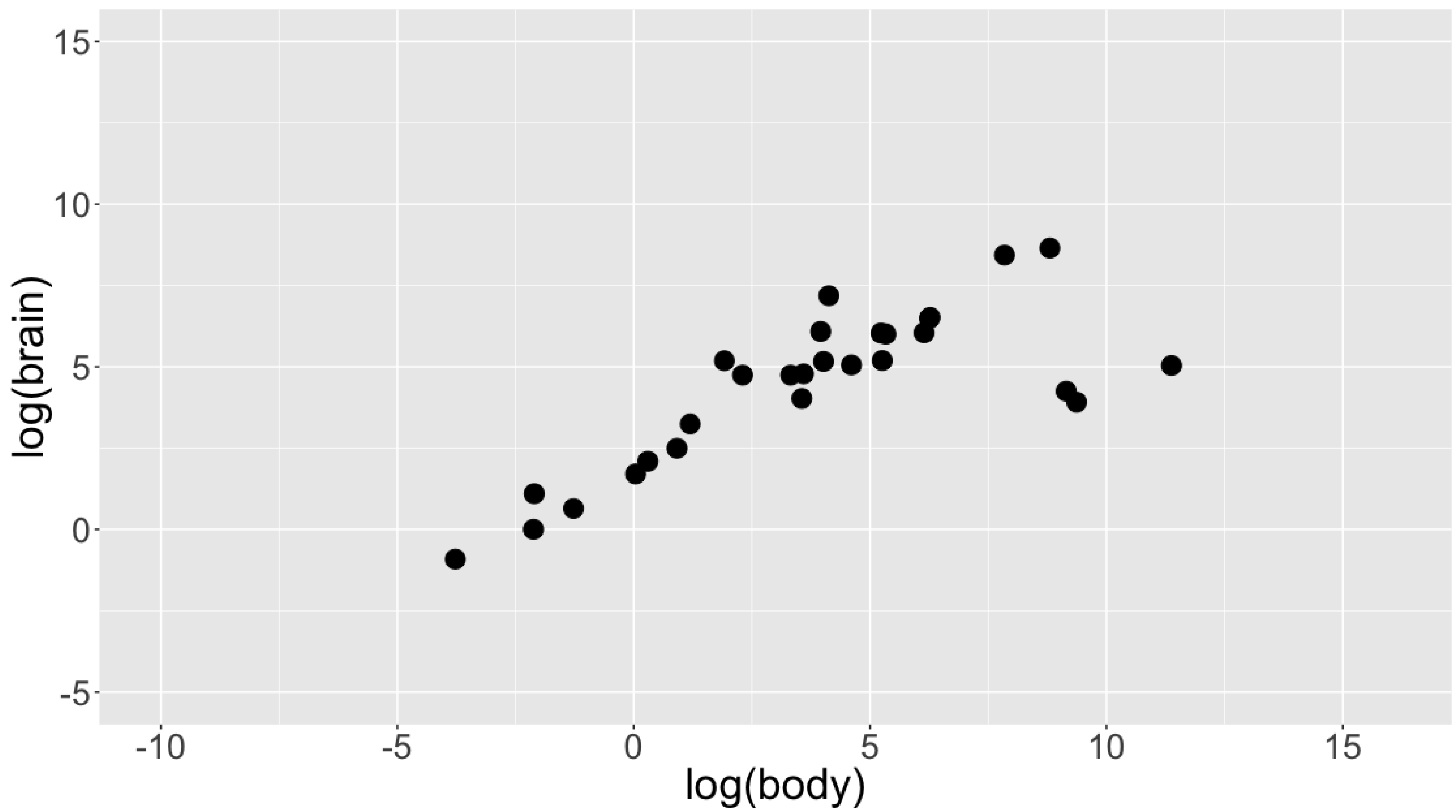
We apply boxplot on logarithms of body weight and brain weight.

```
X <- cbind(log(body),log(brain))
ggplot(X, aes(x = type, y = log_weight)) +
  stat_boxplot(geom="errorbar", width=0.2) + ylab("log(weight)") + xlab("")
```



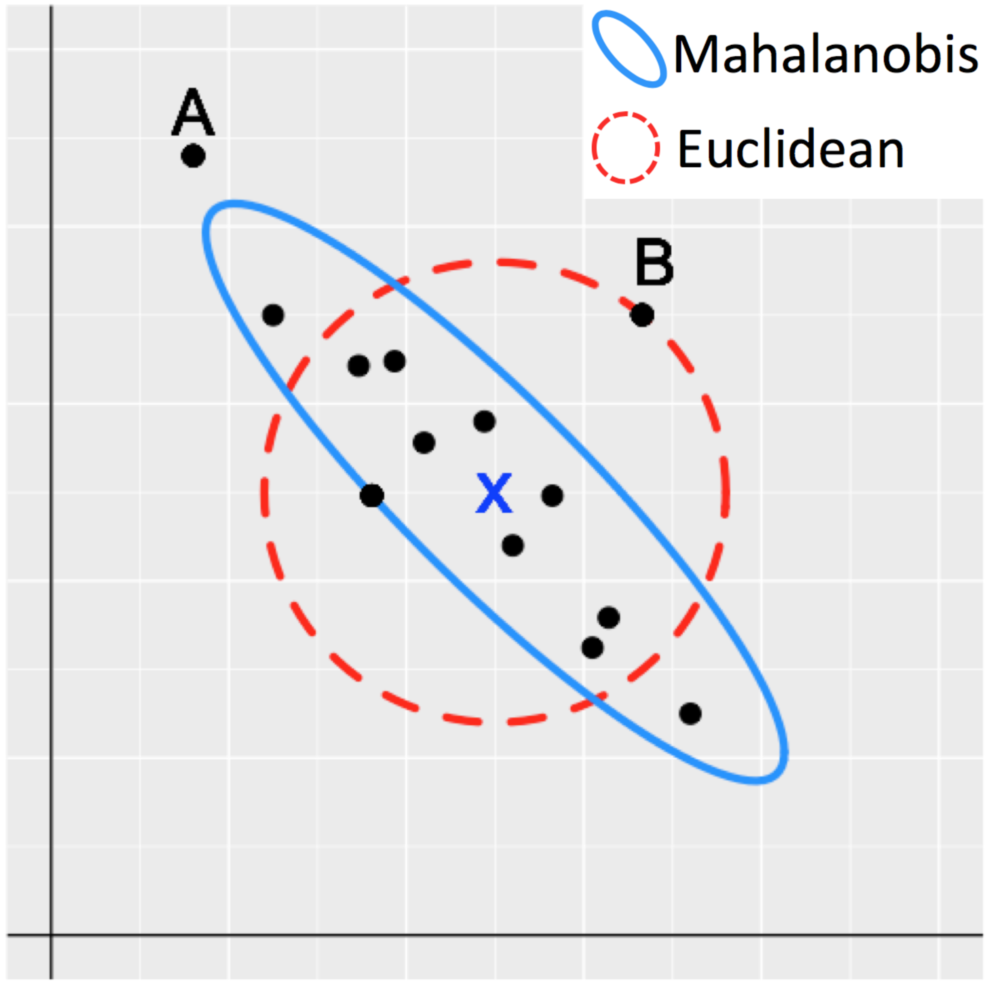
Animals data: scatterplot

```
X <- data.frame(body = log(Animals$body), brain = log(Animals$brain))
fig <- ggplot(X, aes(x = body, y = brain)) + geom_point(size = 5) +
  xlab("log(body)") + ylab("log(brain)") + ylim(-5, 15) +
  scale_x_continuous(limits = c(-10, 16), breaks = seq(-15, 15, 5)))
```



Mahalanobis distance

Mahalanobis (or generalized) distance for observation is the distance from this observation to the center, taking into account the covariance matrix.



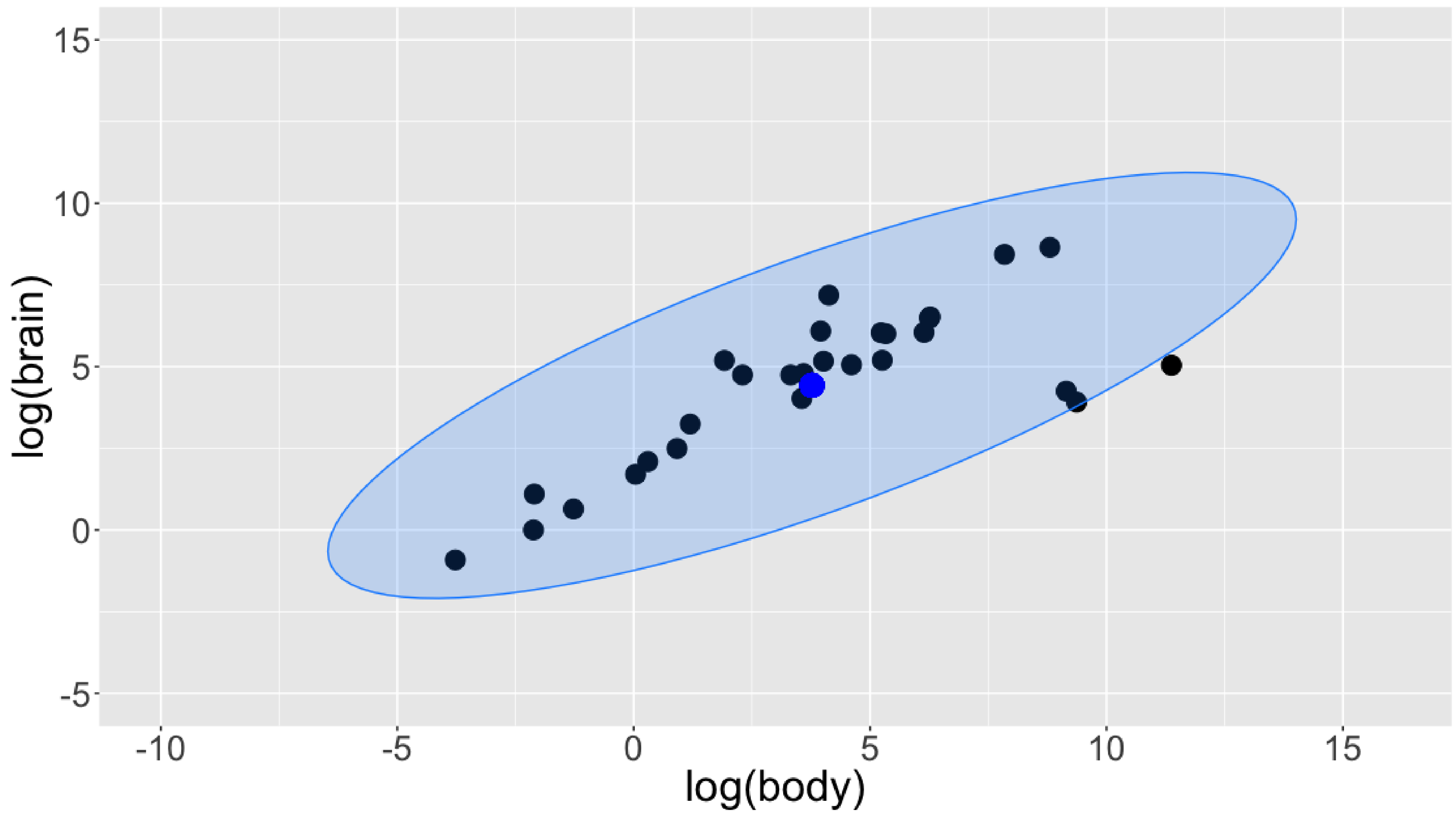
Mahalanobis distance to detect multivariate outliers

- **Classical Mahalanobis distances** : **sample mean** as estimate for location and **sample covariance matrix** as estimate for scatter.
- To detect multivariate outliers the mahalanobis distance is compared with a cut-off value, which is derived from the chisquare distribution.
- In two dimensions we can construct corresponding 97.5% **tolerance ellipsoid**, which is defined by those observations whose Mahalanobis distance does not exceed the cut-off value.

Animals data: tolerance ellipsoid based on Mahalanobis distance

```
animals.clcenter <- colMeans(X)
animals.clcov <- cov(X)
rad <- sqrt(qchisq(0.975, df = ncol(X)))

library(car)
ellipse.cl <- data.frame(ellipse(center = animals.clcenter,
shape = animals.clcov,radius = rad, segments = 100, draw = FALSE))
colnames(ellipse.cl) <- colnames(X)

fig <- fig +
  geom_polygon(data=ellipse.cl, color = "dodgerblue",
               fill = "dodgerblue", alpha = 0.2) +
  geom_point(aes(x = animals.clcenter[1], y = animals.clcenter[2]),
                 color = "blue", size = 6)
fig
```

Animals data: tolerance ellipsoid based on Mahalanobis distance



Robust estimates of location and scatter

Minimum Covariance Determinant (MCD) estimator of Rousseeuw is a popular robust estimator of multivariate location and scatter.

- MCD looks for those h observations whose classical covariance matrix has the **lowest possible determinant**.
- MCD estimate of location is then **mean of these h observations**
- MCD estimate of scatter is then **sample covariance matrix of these h points** (multiplied by consistency factor).
- Reweighting step is applied to improve efficiency at normal data.
- Computation of MCD is difficult, but several **fast algorithms** are proposed.

Robust distance

Robust estimates of location and scatter using MCD

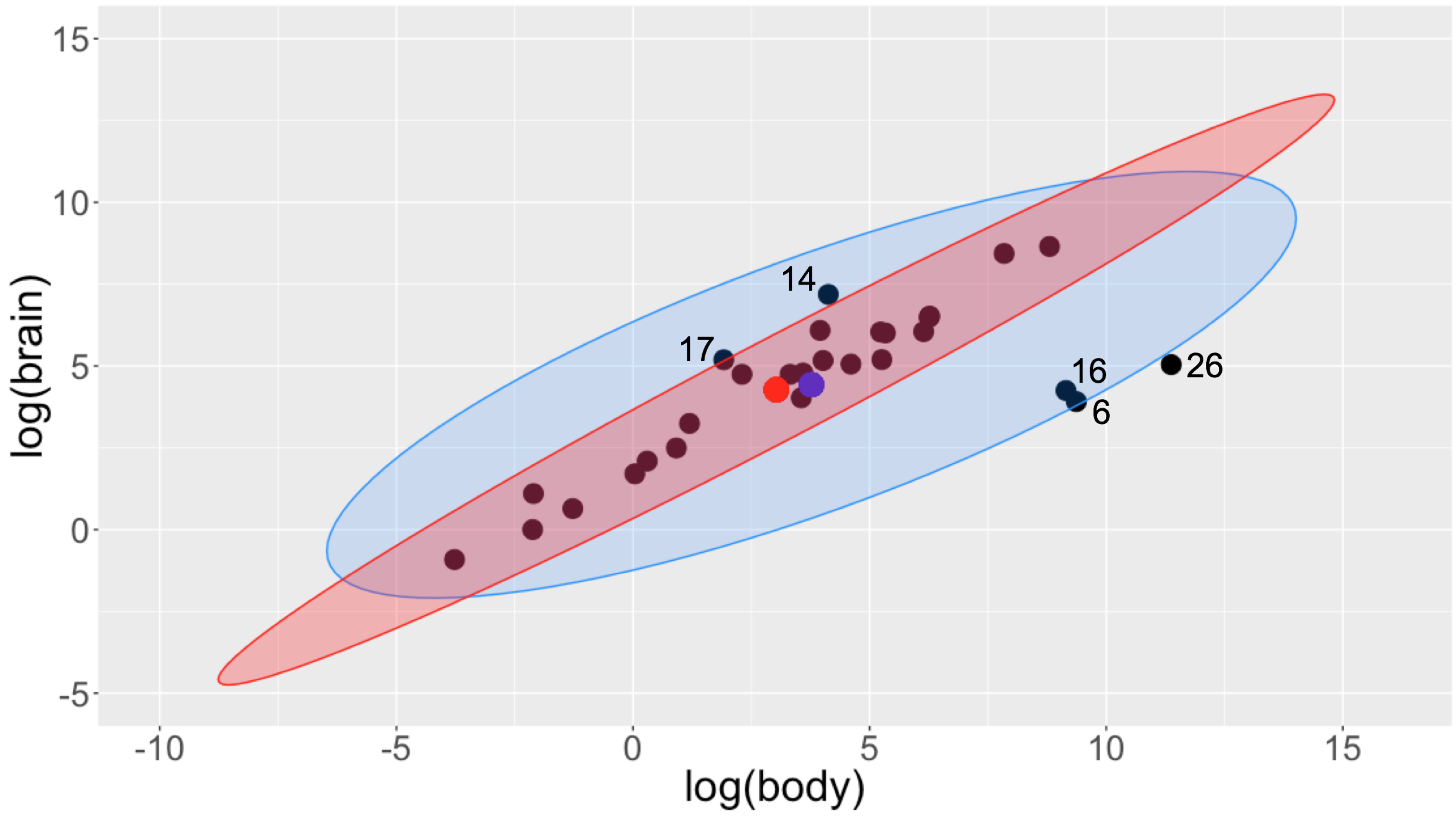
```
library(robustbase)
animals.mcd <- covMcd(X)

# Robust estimate of location
animals.mcd$center

# Robust estimate of scatter
animals.mcd$cov
```

By plugging in these robust estimates of location and scatter in the definition of the Mahalanobis distances, we obtain **robust distances** and can create a robust tolerance ellipsoid.

Animals: robust tolerance ellipsoid

```
library(robustbase)
animals.mcd <- covMcd(X)
ellipse.mcd <- data.frame(ellipse(center = animals.mcd$center,
                                  shape = animals.mcd$cov,
                                  radius=rad, segments=100, draw=FALSE))
colnames(ellipse.mcd) <- colnames(X)

fig <- fig +
  geom_polygon(data=ellipse.mcd, color="red", fill="red", alpha=0.3) +
  geom_point(aes(x = animals.mcd$center[1], y = animals.mcd$center[2]),
                 color = "red", size = 6)
fig
```

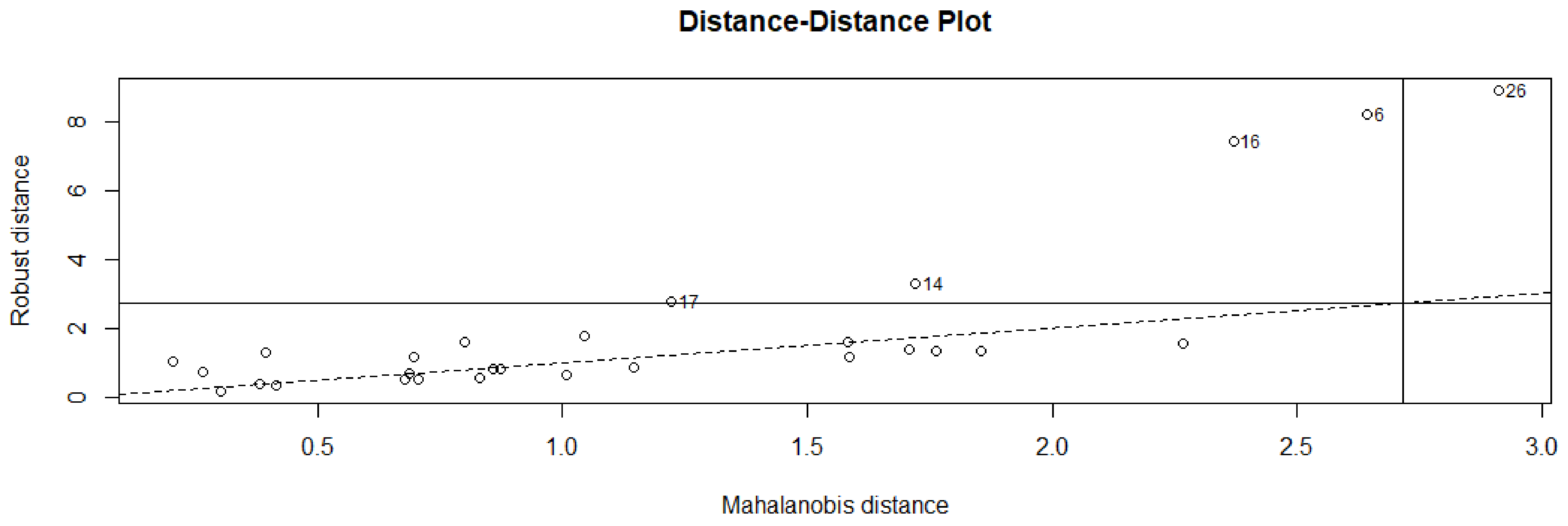
Animals: robust tolerance ellipsoid



Distance-distance plot

- When $p > 3$ it is not possible to visualize the tolerance ellipsoid.
- The **distance-distance plot** shows the robust distance of each observation versus its classical Mahalanobis distance, obtained immediately from `MCD` object.

```
plot(animals.mcd, which = "dd")
```



Animals: check outliers

1. Mountain beaver	15. African elephant
2. Cow	16. Triceratops
3. Gray wolf	17. Rhesus monkey
4. Goat	18. Kangaroo
5. Guinea pig	19. Hamster
6. Diplodocus	20. Mouse
7. Asian elephant	21. Rabbit
8. Donkey	22. Sheep
9. Horse	23. Jaguar
10. Potar monkey	24. Chimpanzee
11. Cat	25. Rat
12. Giraffe	26. Brachiosaurus
13. Gorilla	27. Mole
14. Human	28. Pig

FRAUD DETECTION IN R

Let's practice!