



Data Handling: Import, Cleaning and Visualisation

Lecture 10:

Data Analysis and Basic Statistics with R

Prof. Dr. Ulrich Matter

03/12/2020

Recap: Data Preparation

The dataset is imported, now what?

- In practice: still a long way to go.
- Parsable, but messy data: Inconsistencies, data types, missing observations, wide format.
- **Goal** of data preparation: Dataset is ready for analysis.
- **Key conditions:**
 1. Data values are consistent/clean within each variable.
 2. Variables are of proper data types.
 3. Dataset is in 'tidy' (in long format)!

Some vocabulary

Following Wickham (2014):

- **Dataset**: Collection of **values** (numbers and strings).
- Every value belongs to a **variable** and an **observation**
- **Variable**: Contains all values that measure the same underlying attribute across units.
- **Observation**: Contains all values measured on the same unit (e.g., a person).

Tidy data

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	866	20593360
Brazil	1999	3737	17206362
Brazil	2000	8488	17460898
China	1999	21258	127291272
China	2000	21766	128642583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	866	20593360
Brazil	1999	3737	17206362
Brazil	2000	8488	17460898
China	1999	21258	127291272
China	2000	21766	128642583

observations

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	866	20593360
Brazil	1999	3737	17206362
Brazil	2000	8488	17460898
China	1999	21258	127291272
China	2000	21766	128642583

values

Tidy data. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/3.0/) license)

Data Analysis with R

Merging (Joining) datasets

- Combine data of two datasets in one dataset.
 - Why?
- Needed: Unique identifiers for observations ('keys').

Merging (joining) datasets: example

```
# load packages
```

```
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
# initiate data frame on persons personal spending
```

```
df_c <- data.frame(id = c(1:3,1:3),  
                  money_spent= c(1000, 2000, 6000, 1500, 3000, 5500),  
                  currency = c("CHF", "CHF", "USD", "EUR", "CHF", "USD"),  
                  year=c(2017,2017,2017,2018,2018,2018))
```

```
df_c
```

```
##   id money_spent currency year  
## 1  1         1000      CHF 2017  
## 2  2         2000      CHF 2017  
## 3  3         6000      USD 2017  
## 4  1         1500      EUR 2018
```


Merging (joining) datasets: example

```
# initiate data frame on persons' characteristics
df_p <- data.frame(id = 1:4,
                   first_name = c("Anna", "Betty", "Claire", "Diane"),
                   profession = c("Economist", "Data Scientist", "Data Scientist", "I
df_p

##    id first_name    profession
## 1  1      Anna    Economist
## 2  2     Betty Data Scientist
## 3  3    Claire Data Scientist
## 4  4     Diane    Economist
```

Merging (joining) Datasets: Example

```
df_merged <- merge(df_p, df_c, by="id")  
df_merged
```

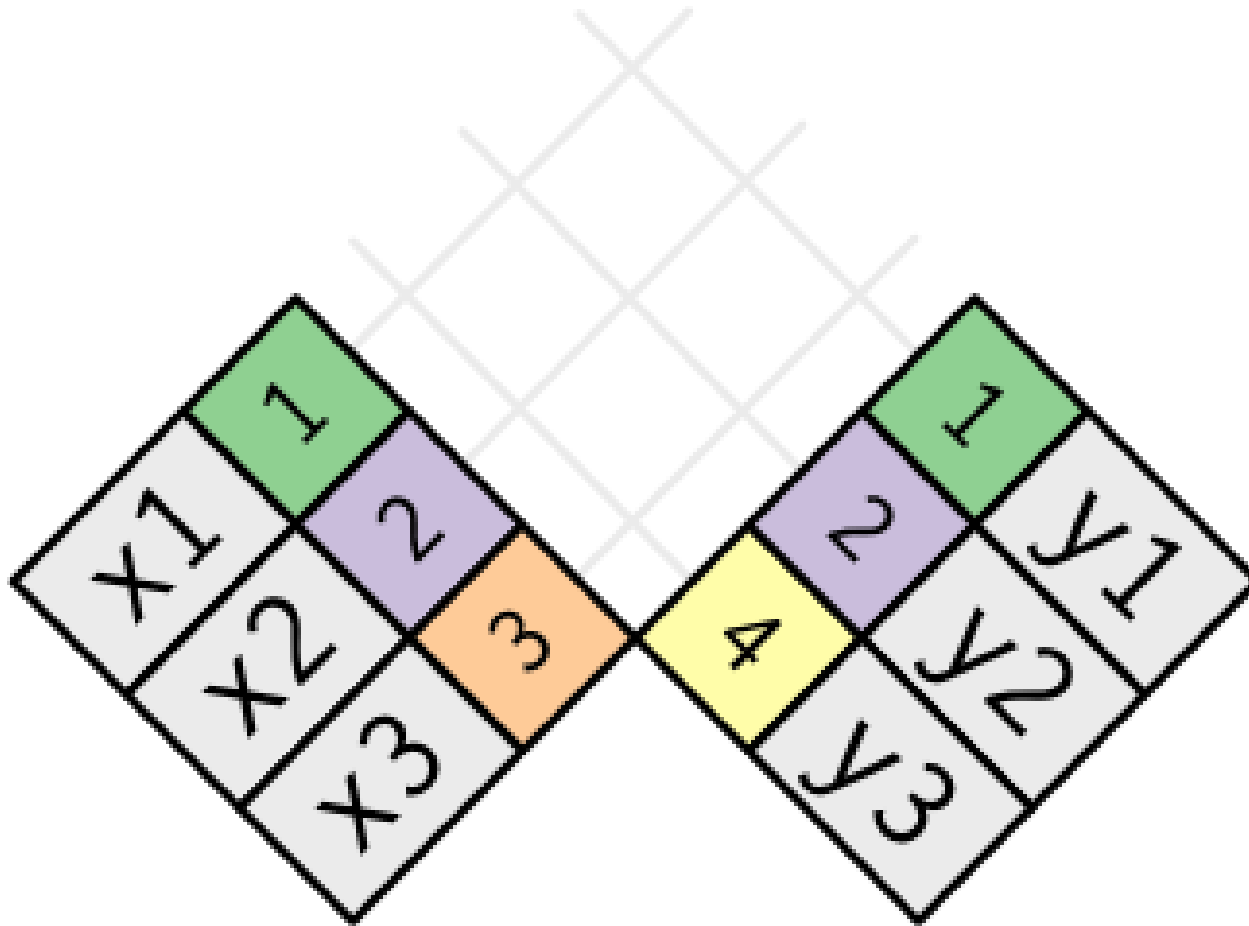
```
##   id first_name      profession money_spent currency year  
## 1  1      Anna      Economist      1000      CHF 2017  
## 2  1      Anna      Economist      1500      EUR 2018  
## 3  2     Betty Data Scientist      2000      CHF 2017  
## 4  2     Betty Data Scientist      3000      CHF 2018  
## 5  3    Claire Data Scientist      6000      USD 2017  
## 6  3    Claire Data Scientist      5500      USD 2018
```

Merging (joining) datasets: concept

x		y	
1	x1	1	y1
2	x2	2	y2
3	x3	4	y3

Join setup. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](#) license.

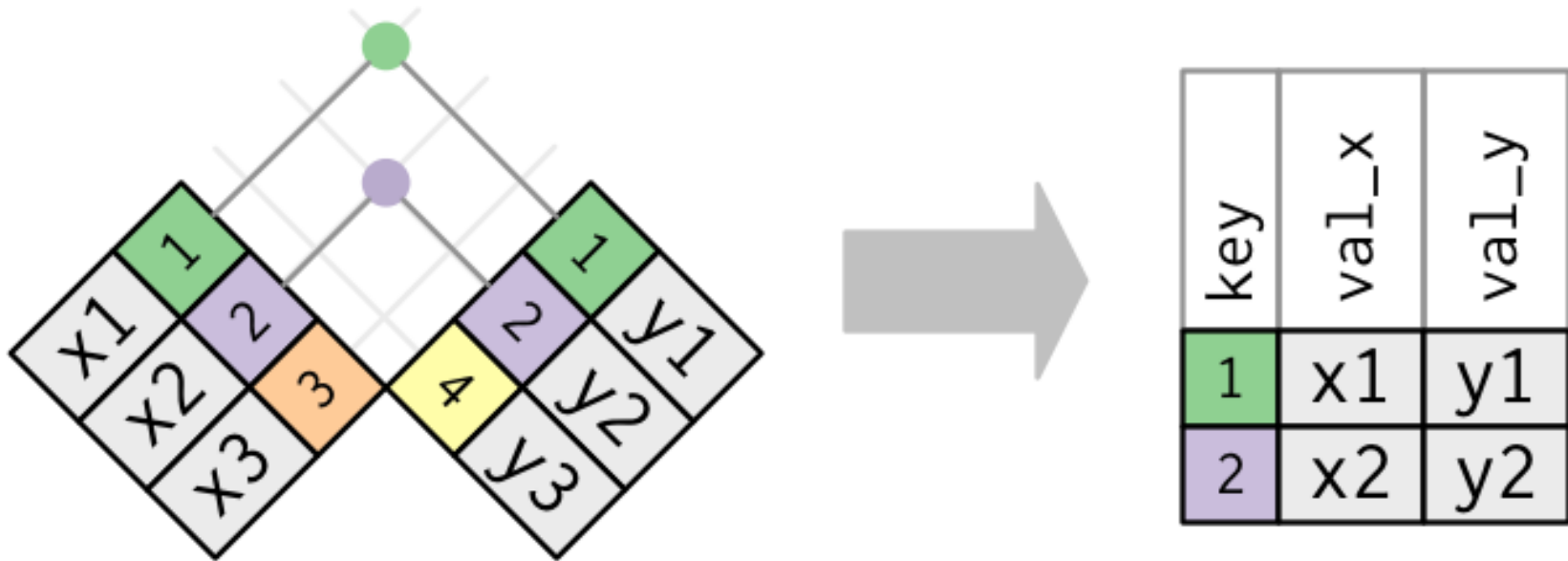
Merging (joining) datasets: concept



Join setup. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/4.0/) license.

Merging (joining) datasets: concept

Merge: Inner join

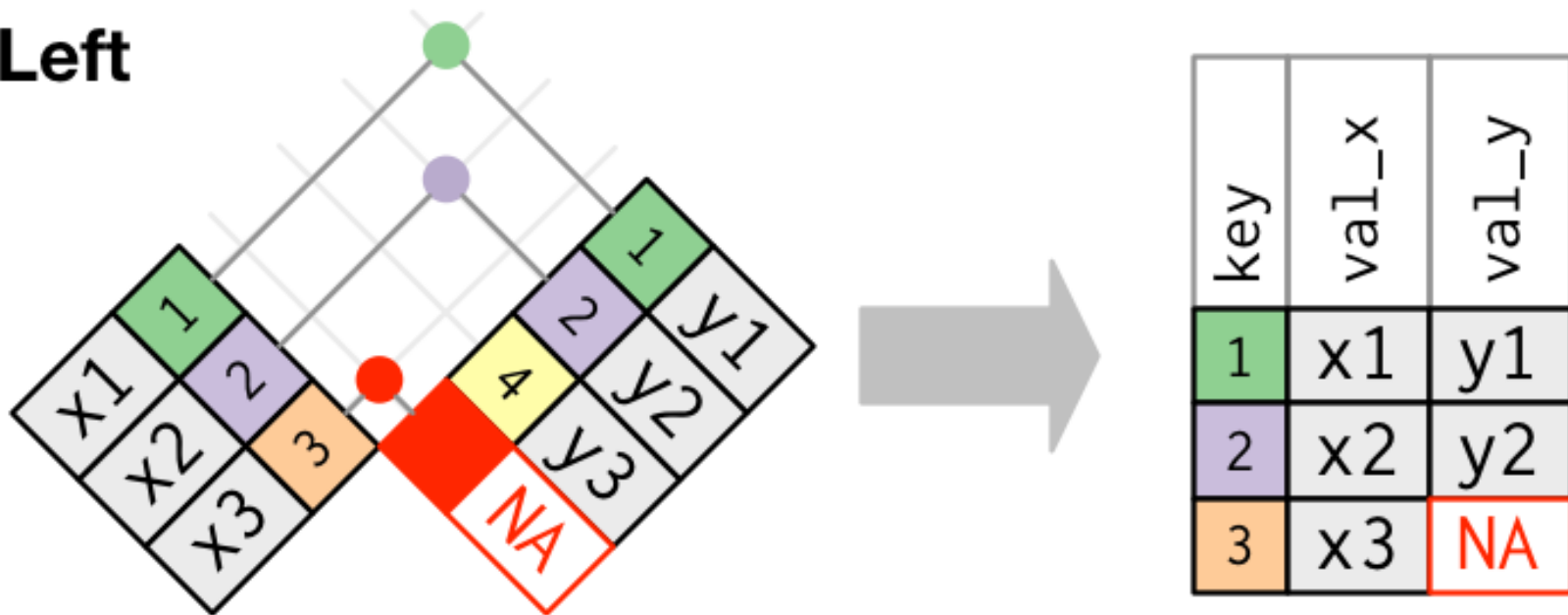


Inner join. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/4.0/) license.

Merging (joining) datasets: concept

Merge all x: Left join

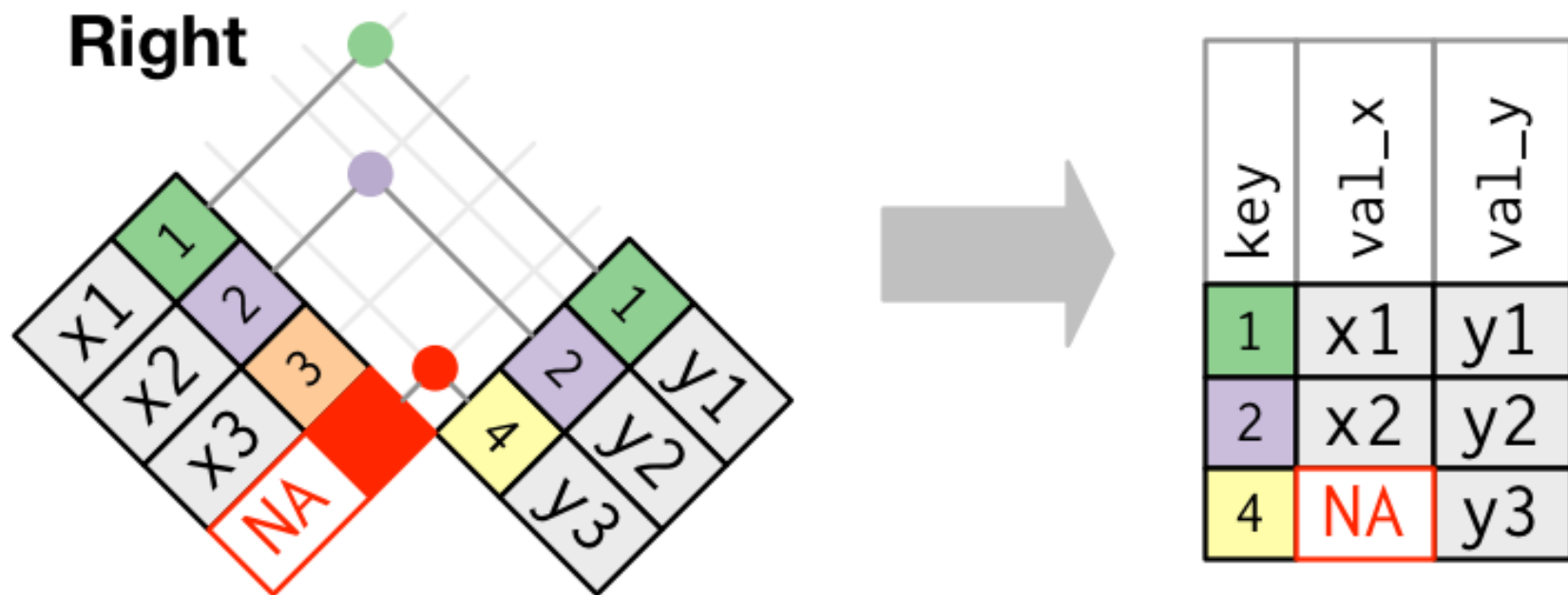
Left



Outer join. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/3.0/) license.

Merging (joining) datasets: concept

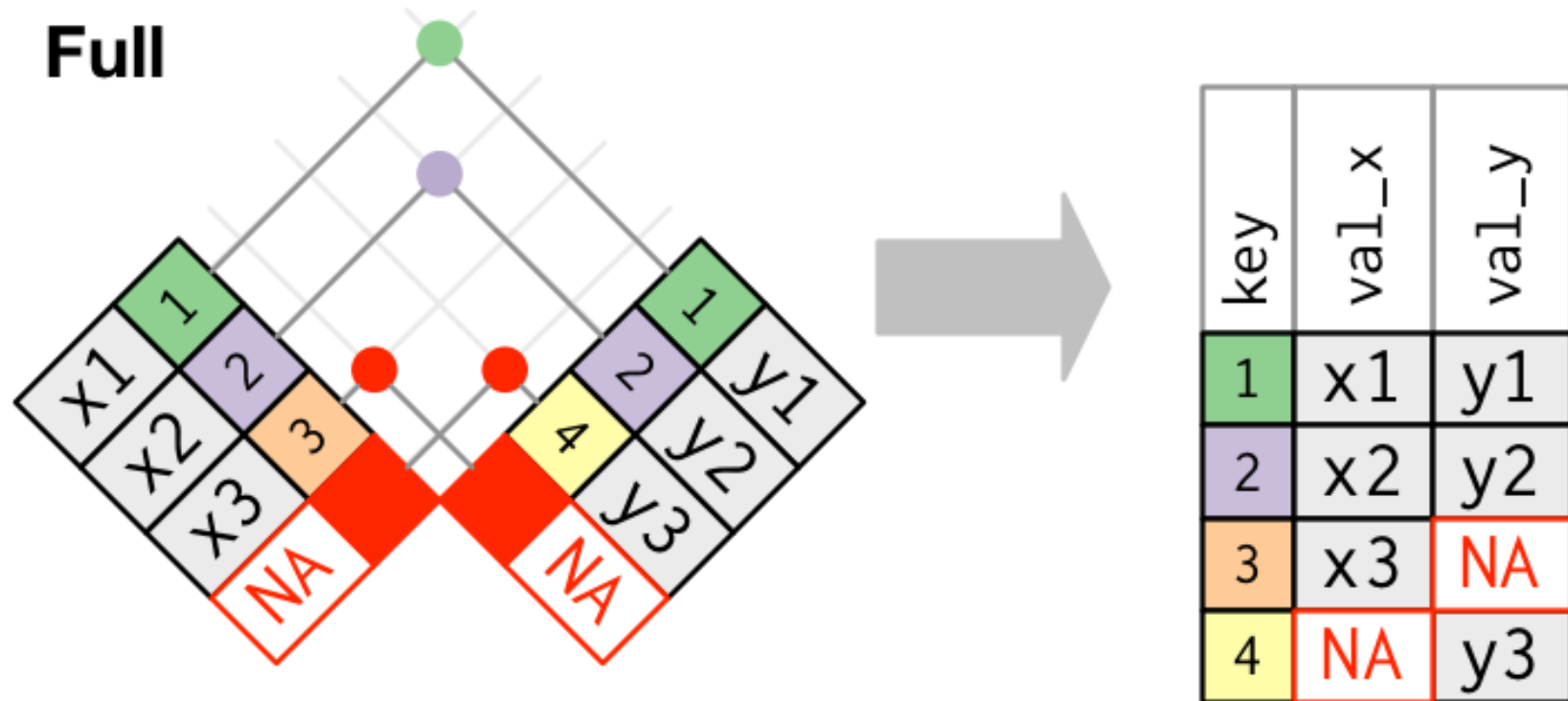
Merge all y: Right join



Outer join. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/3.0/) license.

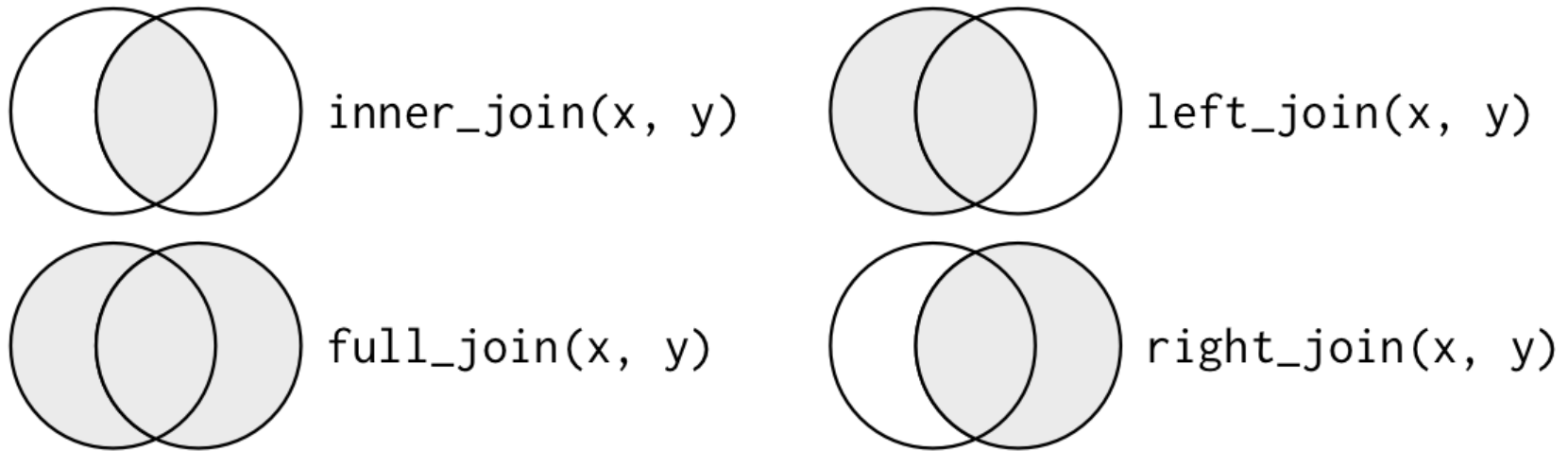
Merging (joining) datasets: concept

Merge all x and all y: Full join



Outer join. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](https://creativecommons.org/licenses/by-sa/3.0/) license.

Merging (joining) datasets: concept



Join Venn Diagramm. Source: Wickham and Grolemund (2017), licensed under the [Creative Commons Attribution-Share Alike 3.0 United States](#) license.

Merging (joining) datasets: R

Overview by Wickham and Grolemund (2017):

dplyr (tidyverse)

base::merge

inner_join(x, y)

`merge(x, y)`

left_join(x, y)

`merge(x, y, all.x = TRUE)`

right_join(x, y)

`merge(x, y, all.y = TRUE),`

full_join(x, y)

`merge(x, y, all.x = TRUE, all.y = TRUE)`

Data Summaries: Selecting, Filtering, and Mutating

Data summaries

- First step of analysis.
- Get overview over dataset.
- Show key aspects of data.
 - Inform your own statistical analysis.
 - Inform audience (helps understand advanced analytics parts)

Data summaries: first steps

- **Select** subset of variables (e.g., for comparisons).
- **Filter** the dataset (some observations not needed in **this** analysis).
- **Mutate** the dataset: additional values needed

Select, filter, mutate in R (tidyverse)

- `select()`
- `filter()`
- `mutate()`

Data Summaries: Aggregate Statistics

Descriptive/aggregate statistics

- Overview of key characteristics of main variables used in analysis.
- Key characteristics:
 - mean
 - standard deviation
 - No. of observations
 - etc.

Aggregate statistics in R

1. Function to compute statistic (`mean()`).
2. Function to **apply** the statistics function to one or several columns in a tidy dataset.
 - All values.
 - By group (observation categories, e.g. by gender)

Aggregate statistics in R

- `summarise()` (in `tidyverse`)
- `group_by()` (in `tidyverse`)
- `sapply()`, `apply()`, `lapply()`, etc. (in `base`)

Understanding Statistics and Probability with Code

Random numbers and computation

Can computers generate random numbers?!

Random draws and distributions

```
normal_distr <- rnorm(1000)  
hist(normal_distr)
```

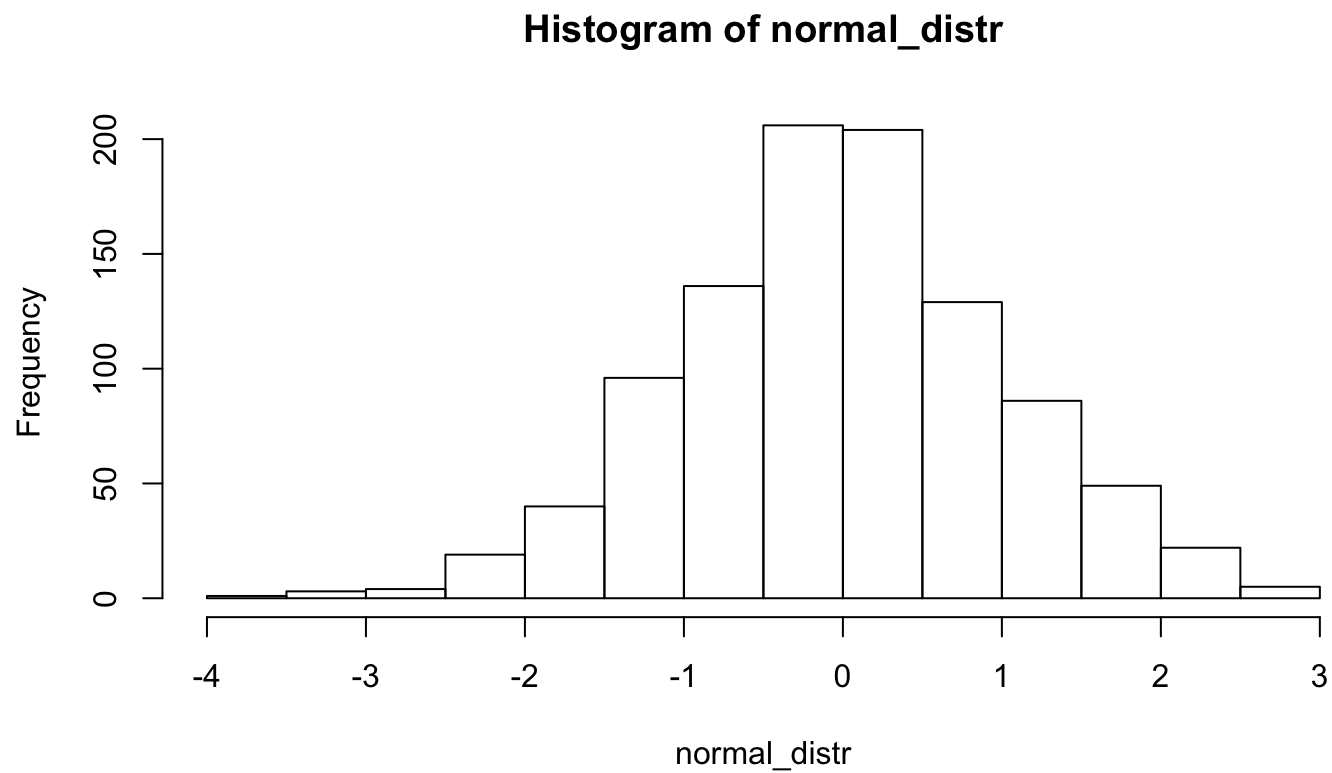
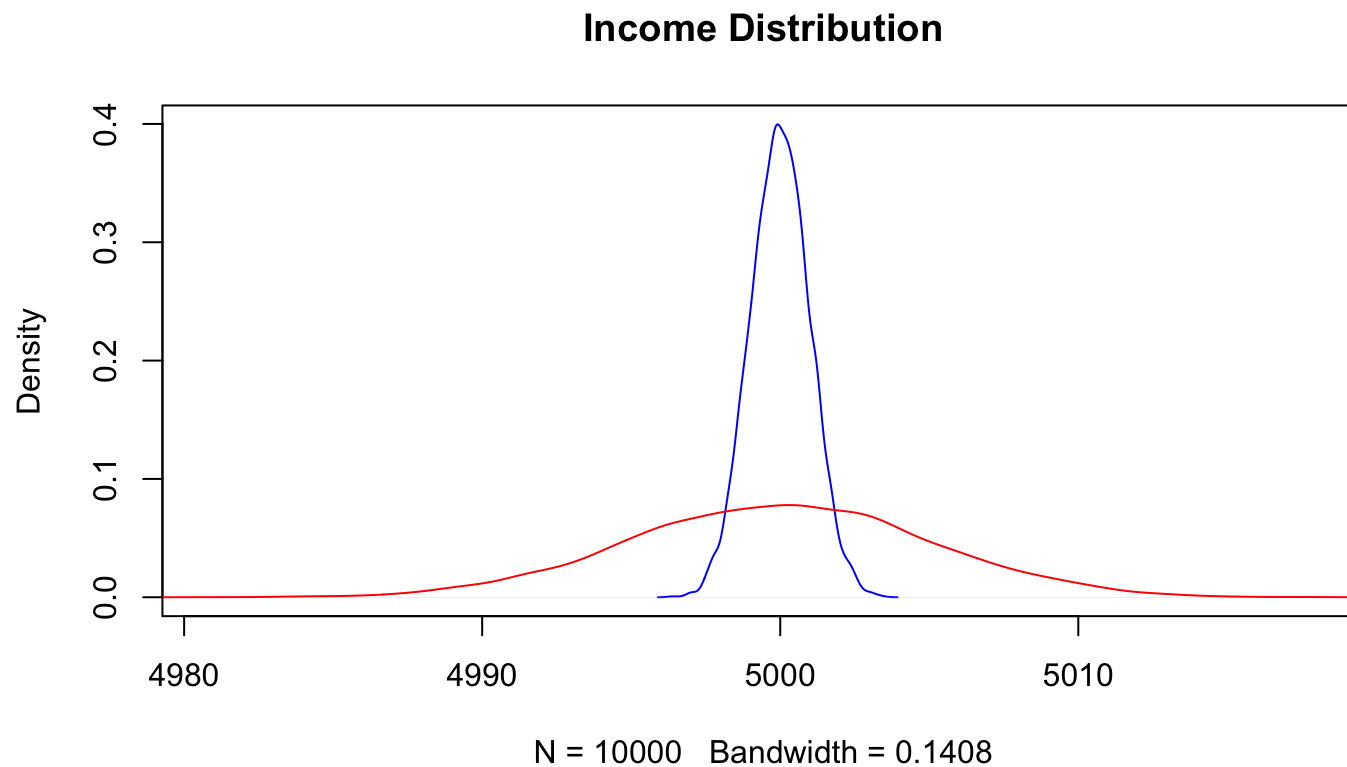


Illustration of variability

```
# draw a random sample from a normal distribution with a large standard deviation  
largevar <- rnorm(10000, mean = 5000, sd = 5)  
# draw a random sample from a normal distribution with a small standard deviation  
littlevar <- rnorm(10000, mean = 5000, sd = 1)
```

Illustration of variability

```
# visualize the distributions of both samples with a density plot  
plot(density(littlevar), col = "blue",  
      xlim=c(min(largevar), max(largevar)), main="Income Distribution")  
lines(density(largevar), col = "red")
```



Skewness and kurtosis

```
# install the R-package called "moments" with the following command (if not installed)  
# install.packages("moments")
```

```
# load the package  
library(moments)
```

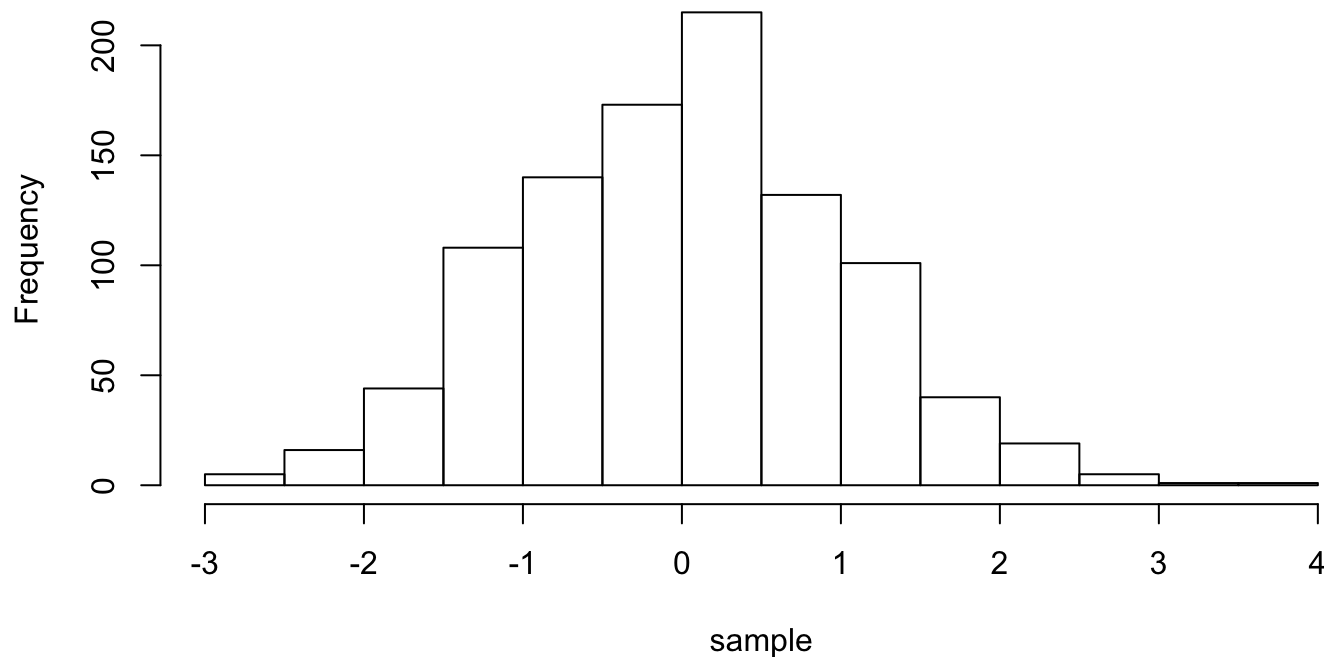

Skewness

- Skewness refers to how symmetric the frequency distribution of a variable is.
- For example, a distribution can be **'positively skewed'** meaning it has a **long tail on the right** and a lot of 'mass' (observations) on the left.

Skewness: R example

```
# draw a random sample of simulated data from a normal distribution  
# the sample is of size 1000 (hence, n = 1000)  
sample <- rnorm(n = 1000)  
  
# plot a histogram and a density plot of that sample  
# note that the distribution is neither strongly positively nor negatively skewed  
# (this is to be expected, as we have drawn a sample from a normal distribution!)  
hist(sample)
```

Histogram of sample



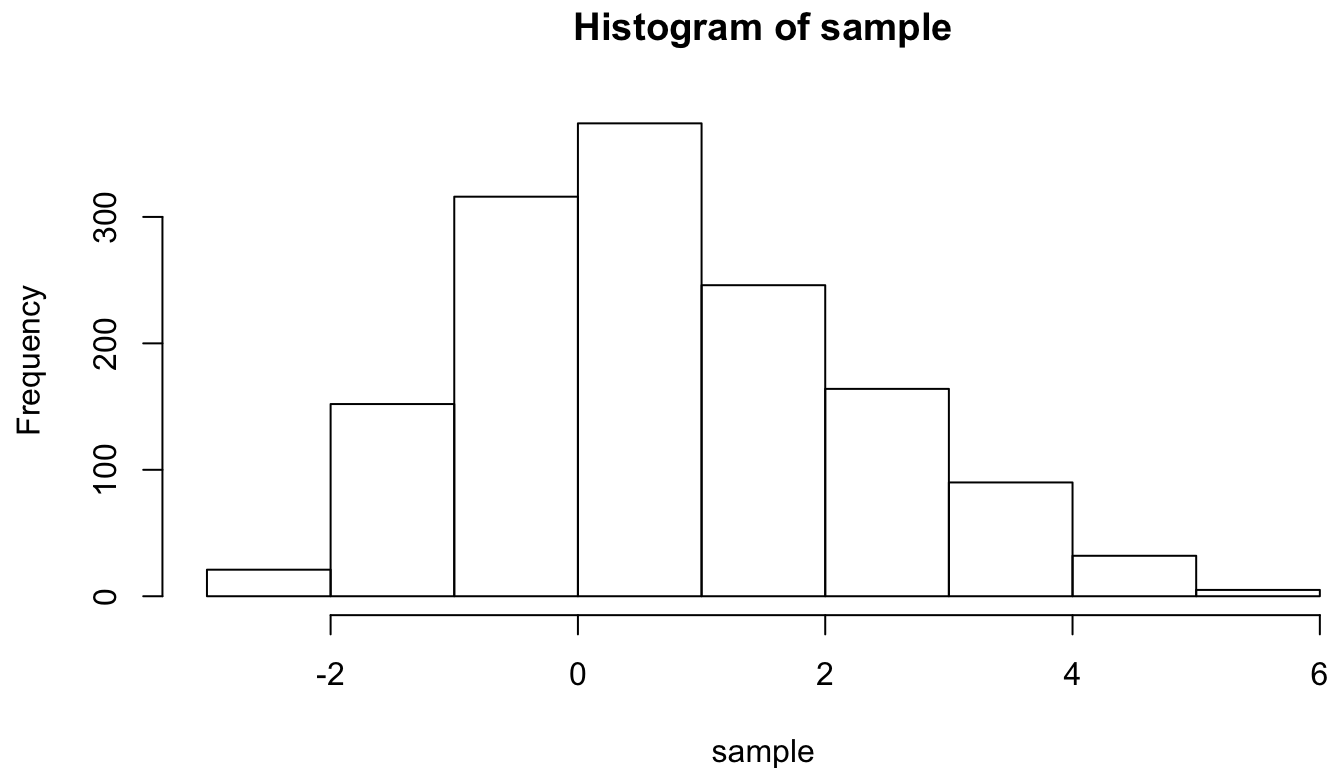
Skewness: R example

```
# now compute the skewness  
skewness(sample)
```

```
## [1] 0.06750904
```

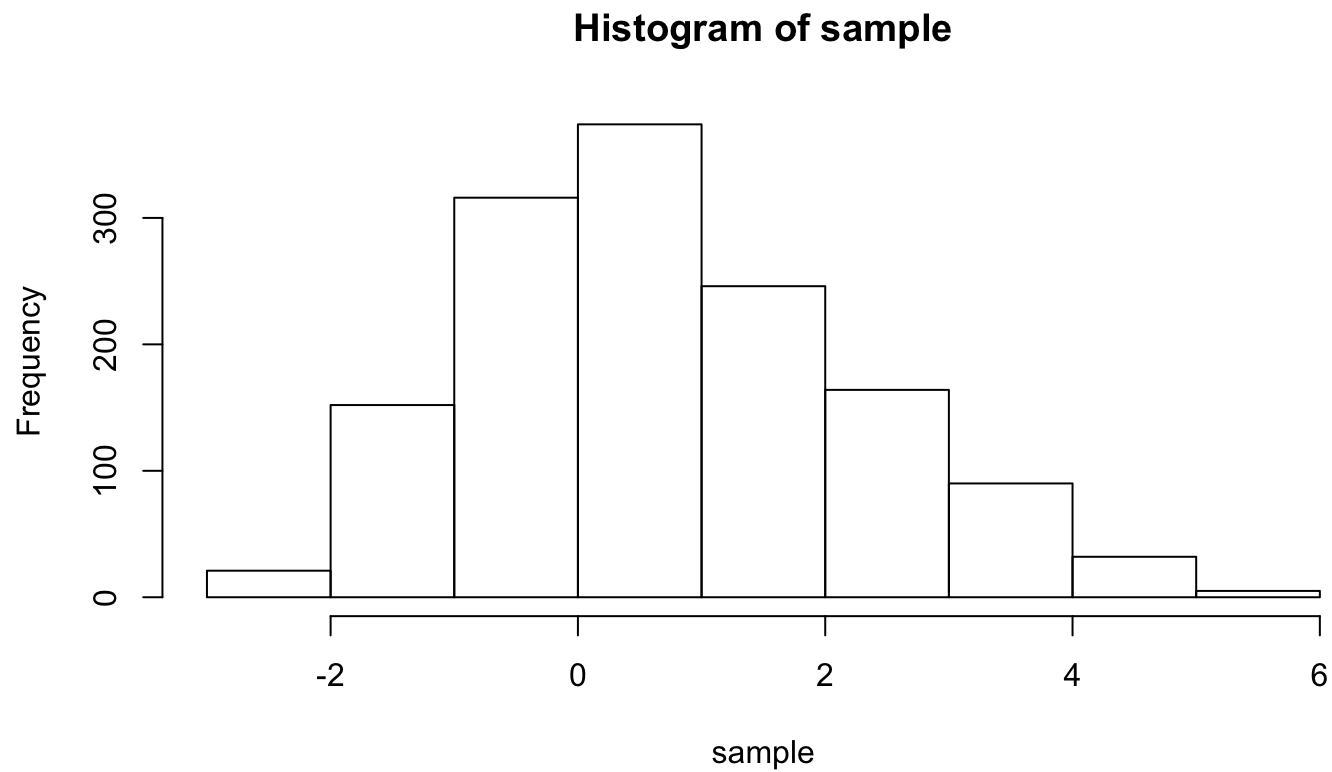
Skewness: R example

```
# now we intentionally change our sample to be strongly positively skewed  
# we do that by adding some outliers (observations with very high values) to the sample  
sample <- c(sample, (rnorm(200) + 2), (rnorm(200) + 3))  
  
# have a look at the distribution and re-calculate the skewness  
hist(sample)
```



Skewness: R example

```
# have a look at the distribution and re-calculate the skewness  
hist(sample)
```



```
skewness(sample)
```

```
## [1] 0.5172931
```

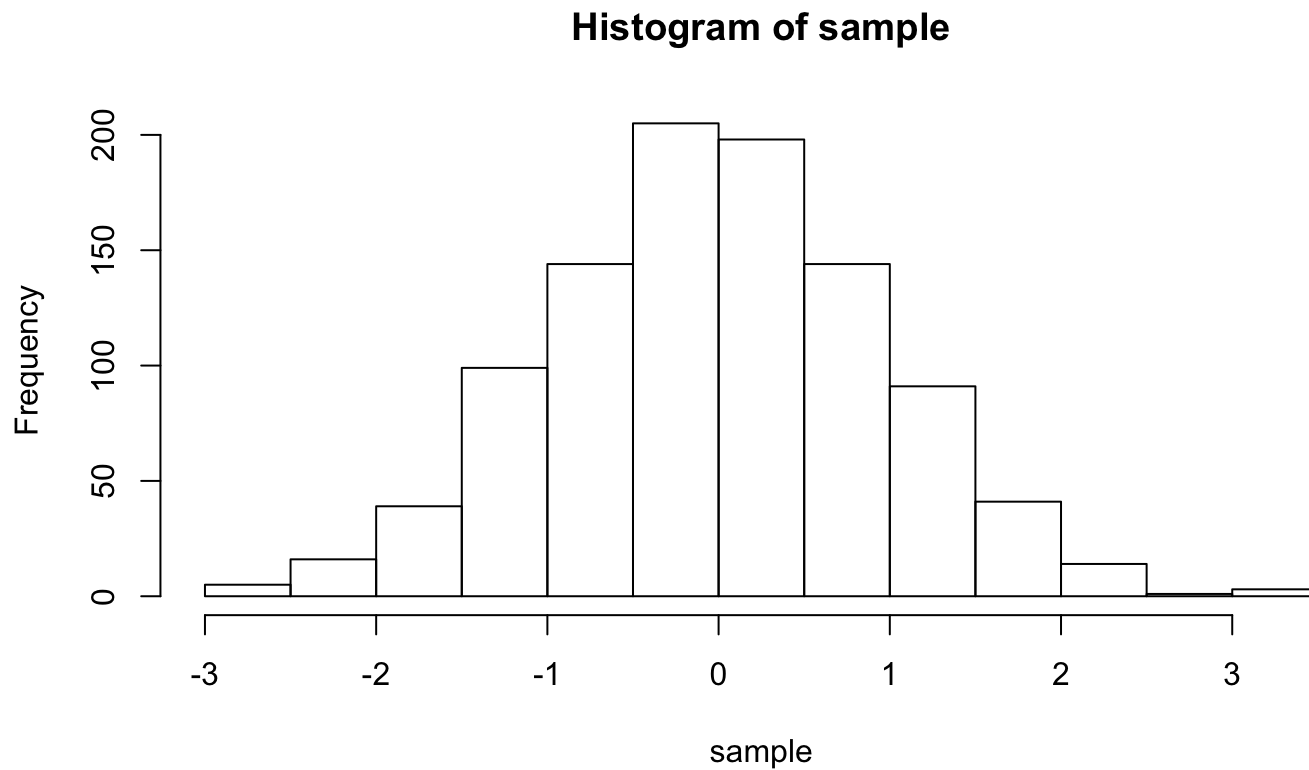
Kurtosis

- Kurtosis refers to how much 'mass' a distribution has in its 'tails'.
- Tells us something about whether a distribution tends to have a lot of outliers.

Kurtosis: R example

```
# draw a random sample of simulated data from a normal distribution  
# the sample is of size 1000 (hence, n = 1000)  
sample <- rnorm(n = 1000)
```

```
# plot the density & compute the kurtosis  
hist(sample)
```



Kurtosis: R example

```
# compute the kurtosis  
kurtosis(sample)
```

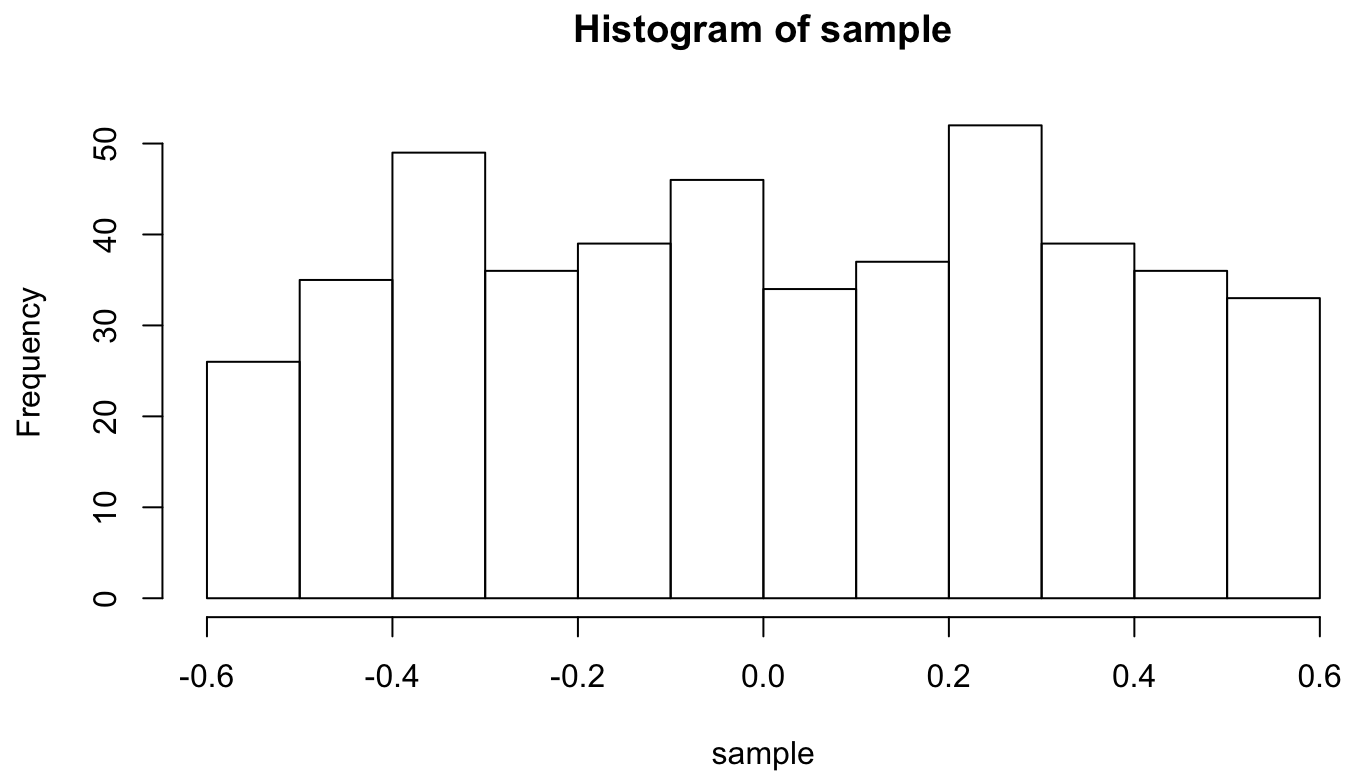
```
## [1] 2.984136
```


Kurtosis: R example

```
# now lets remove observations from the extremes in this distribution  
# we thus intentionally alter the distribution to have less mass in its tails  
sample <- sample[ sample > -0.6 & sample < 0.6]
```

Kurtosis: R example

```
# plot the distribution again and see how the tails have changed  
hist(sample)
```



```
# re-calculate the kurtosis  
kurtosis(sample)
```

Compute the skewness in R

Skewness

own implementation

```
sum((sample-mean(sample))^3) / ((length(sample)-1) * sd(sample)^3)
```

```
## [1] -0.02228835
```

implementation in moments package

```
skewness(sample)
```

```
## [1] -0.02231251
```

Compute the kurtosis in R

Kurtosis

own implementation

```
sum((sample-mean(sample))^4) / ((length(sample)-1) * sd(sample)^4)
```

```
## [1] 1.838455
```

implementation in moments package

```
kurtosis(sample)
```

```
## [1] 1.842443
```

The Law of Large Numbers (LLN)

- Important statistical property.
- Essentially describes how the behavior of **sample averages** is related to **sample size**.
- States that the **sample mean** can come arbitrarily close to the **population mean** by increasing the sample size N .

The Law of Large Numbers (LLN): playing dice

- Roll a fair die, record result: either 1, 2, 3, 4, 5, or 6.
- Probability of each possible outcome is $1/6$.

The Law of Large Numbers (LLN): playing dice

- Roll a fair die, record result: either 1, 2, 3, 4, 5, or 6.
- Probability of each possible outcome is $1/6$.
- Expected value (average in the long run): $(1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$

The Law of Large Numbers (LLN): playing dice

- Roll a fair die, record result: either 1, 2, 3, 4, 5, or 6.
- Probability of each possible outcome is $1/6$.
- Expected value (average in the long run): $(1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$
- Proof?
 - Mathematically.
 - Or: Experiment/Simulation (with R).

LLN in R

```
# first we define the potential values a die can take  
dvalues <- 1:6 # the : operator generates a regular sequence of numbers (from:to)  
dvalues
```

```
## [1] 1 2 3 4 5 6
```

```
# define the size of the sample n (how often do we roll the die...)  
# for a start, we only roll the die ten times  
n <- 10  
# draw the random sample: 'roll the die n times and record each result'  
results <- sample( x = dvalues, size = n, replace = TRUE)  
# compute the mean  
mean(results)
```

```
## [1] 3.3
```

LLN in R

```
n <- 100
# draw the random sample: 'roll the die n times and record each result'
results <- sample( x = dvalues, size = n, replace = TRUE)
# compute the mean
mean(results)

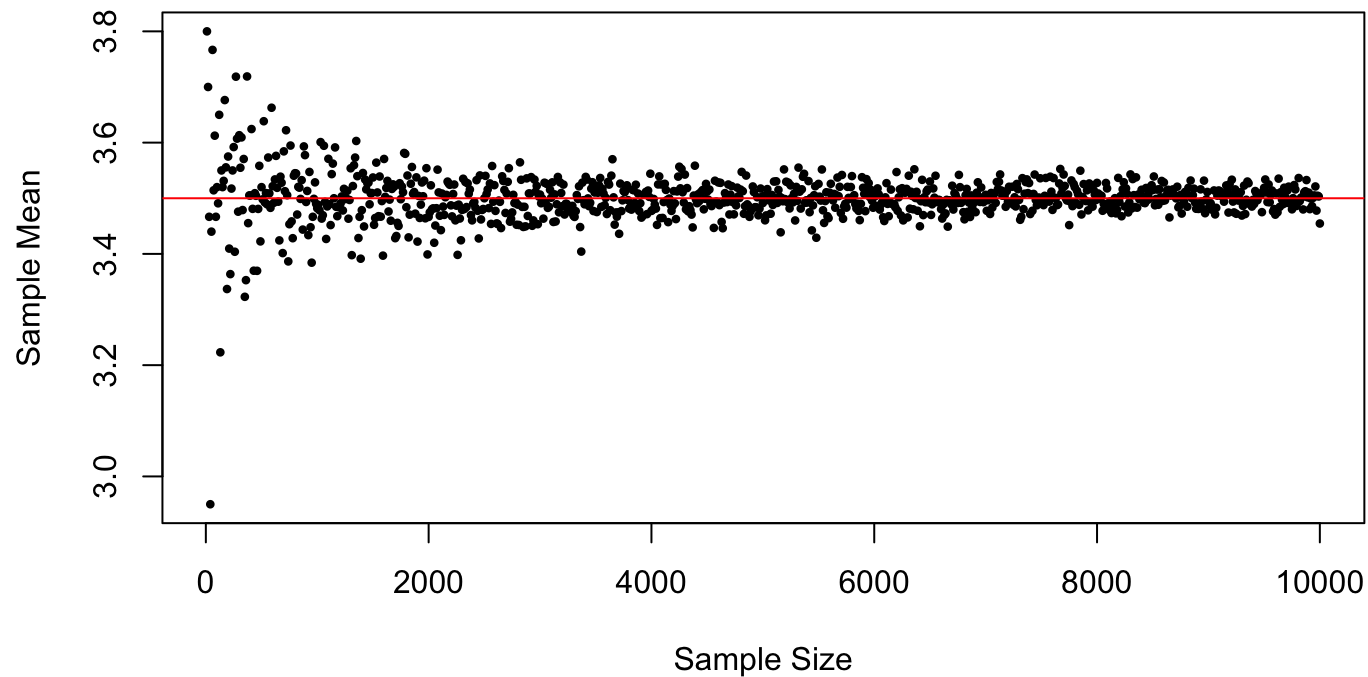
## [1] 3.68
```

LLN in R

```
# essentially, what we are doing here is repeating the experiment above many times,  
# each time increasing n  
# define the set of sample sizes  
ns <- seq(from = 10, to = 10000, by = 10)  
# initiate an empty list to record the results  
means <- list()  
length(means) <- length(ns)  
# iterate through each sample size: 'repeat the die experiment for each sample size'  
for (i in 1:length(ns)) {  
  
    means[[i]] <- mean(sample( x = dvalues, size = ns[i], replace = TRUE))  
}
```

LLN in R

```
# visualize the result: plot sample means against sample size  
plot(ns, unlist(means),  
     ylab = "Sample Mean",  
     xlab = "Sample Size",  
     pch = 16,  
     cex = .6)  
abline(h = 3.5, col = "red")
```



Q&A

References

Wickham, Hadley. 2014. "Tidy Data." **Journal of Statistical Software, Articles** 59 (10): 1–23.
<https://doi.org/10.18637/jss.v059.i10>.

Wickham, Hadley, and Garrett Grolemund. 2017. Sebastopol, CA: O'Reilly. <http://r4ds.had.co.nz/>.