Data Types and Tabular Data

Working with Rectangular Data

Alexander Tolios

day 2

What is data?

Basic structure of data

A 'datum' is the single unit of information. A collection of multiple 'datums' is called data.

The most prevalent form of data is the 'rectangular' data format (which can be easily stored in spreadsheets and relational databases).

- Single observations are often called `data points` (often depicted as rows in a spreadsheet).
- Often multiple `variables` (also termed `features` or `dimensions`, often depicted as columns in a spreadsheet) are used to store different types of information.
- The combination of multiple data points, each containing of multiple `variables`, are called a `data set`.

This dataset is called a `wide format` dataset (the `classical table` most are familiar with).

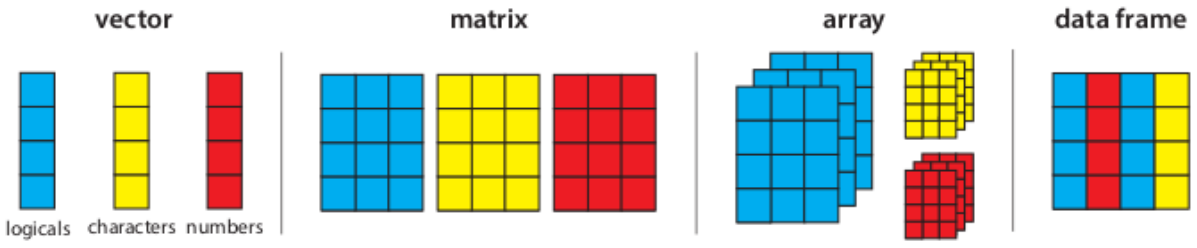


Figure 1: wide-format data compared to different array types

```
#>         rowname  qsec gear vs
#> 1     Mazda RX4 16.46    4  0
#> 2 Mazda RX4 Wag 17.02    4  0
#> 3    Datsun 710 18.61    4  1
```

In contrast, a `long format` table depicts shows:

- each row is an observation
- only one column contains the actual value of the observation
- all other columns define the specification of the observation (e.g. `ID`, `variable`, …)

```
#> # A tibble: 9 x 3
#>   rowname       variable value
#>   <chr>         <chr>    <dbl>
#> 1 Mazda RX4     qsec      16.5
```

```
#> 2 Mazda RX4     gear       4
#> 3 Mazda RX4     vs         0
#> 4 Mazda RX4 Wag qsec      17.0
#> 5 Mazda RX4 Wag gear       4
#> 6 Mazda RX4 Wag vs         0
#> 7 Datsun 710    qsec      18.6
#> 8 Datsun 710    gear       4
#> 9 Datsun 710    vs         1
```

Other types of data sets exist (e.g. data stored in xml-files or json-objects) but are out of the scope of this lecture.

Data sets can be transformed from one format into a different one.

Data sets can be incomplete, if a data point has no entry in one or multiple variables. Missing values can be important, so don't just blindly remove them but choose a dimension where you can show them easily (e.g. color).

Types of variables

Variables can have different types, e.g.:

- boolean (2 categories)
  - `TRUE` != `FALSE`
- factor (`n` categories, exhaustive)
  - unordered (`a` != `b` != `c`)
  - ordered (`a` < `b` < `c`)
- integer (infinite, integer number)
  - clearly defined distance and number of possible states between two numbers
- real/floating point/double (infinite, real number)
  - without a real starting point (measure is arbitrary)
  - with a real starting point (goes from a 'useful' 0 to infinity)
- character/string/free text (infinite, anything)
  - just a label without any additional intrinsic information

What is the data type?

- Do you read books in your spare time?
- Do you prefer the political party a, b, or c?
- Was your last math test in high school, in an undergrad class or during graduate school?
- How many cups of tea do you drink during one day?
- What is your body temperature (in °C)?
- What is your height?
- What are the topics of your research?

The 'lower' in the above hierarchy the data is, the more information it conveys. But working with it might be less useful.

What information can we extract from data

General information

Variation describes how much a single variable varies by itself. This is measured differently if the variable is categorical or numerical. It also makes a difference if the observations are independent of each other or if they are in a specific relationship (e.g. in the case of time series).

Also keep in mind that a 'true' variable description might not be a useful one.

Figure 2: true isn't the same as meaningful

Exercise

What can you tell me about the following datasets? How could you visually represent those datasets?

Example: palmerpenguins

```
#> # A tibble: 6 x 8
#>   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
#>   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
#> 1 Adelie  Torgersen           39.1          18.7               181        3750
#> 2 Adelie  Torgersen           39.5          17.4               186        3800
#> 3 Adelie  Torgersen           40.3          18                 195        3250
#> 4 Adelie  Torgersen           NA            NA                  NA          NA
#> 5 Adelie  Torgersen           36.7          19.3               193        3450
#> 6 Adelie  Torgersen           39.3          20.6               190        3650
#> # i 2 more variables: sex <fct>, year <int>
```

Table 1: Data summary

Name	palmerpenguins::penguins
Number of rows	344
Number of columns	8
Column type frequency:	
factor	3
numeric	5
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
species	0	1.00	FALSE	3	Ade: 152, Gen: 124, Chi: 68
island	0	1.00	FALSE	3	Bis: 168, Dre: 124, Tor: 52

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
sex	11	0.97	FALSE	2	mal: 168, fem: 165

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
bill_length_mm	2	0.99	43.92	5.46	32.1	39.23	44.45	48.5	59.6	
bill_depth_mm	2	0.99	17.15	1.97	13.1	15.60	17.30	18.7	21.5	
flipper_length_mm	2	0.99	200.92	14.06	172.0	190.00	197.00	213.0	231.0	
body_mass_g	2	0.99	4201.75	801.95	2700.0	3550.00	4050.00	4750.0	6300.0	
year	0	1.00	2008.03	0.82	2007.0	2007.00	2008.00	2009.0	2009.0	

```
#>                            variable q_zeros p_zeros q_na        p_na q_inf
#> species                     species       0       0    0 0.000000000     0
#> island                       island       0       0    0 0.000000000     0
#> bill_length_mm       bill_length_mm       0       0    2 0.005813953     0
#> bill_depth_mm         bill_depth_mm       0       0    2 0.005813953     0
#> flipper_length_mm flipper_length_mm       0       0    2 0.005813953     0
#> body_mass_g             body_mass_g       0       0    2 0.005813953     0
#> sex                             sex       0       0   11 0.031976744     0
#> year                           year       0       0    0 0.000000000     0
#>                   p_inf    type unique
#> species               0  factor      3
#> island                0  factor      3
#> bill_length_mm        0 numeric    164
#> bill_depth_mm         0 numeric     80
#> flipper_length_mm     0 integer     55
#> body_mass_g           0 integer     94
#> sex                   0  factor      2
#> year                  0 integer      3
```

Analyzing a single dimension in a dataset

Single categorical variable

If you have a single categorical variable, just look at the proportion of values.

```
#>  penguins$sex   n    percent valid_percent
#>        female 165 0.47965116     0.4954955
#>          male 168 0.48837209     0.5045045
#>          <NA>  11 0.03197674            NA
```

This is the numerical equivalent of a stacked bar chart.

Single numerical variable

When looking into a single numerical variable, look into the distribution of the data.

Table 4: Data summary

Name	penguins$flipper_length_m…
Number of rows	344

Table 4: Data summary

Number of columns	1
Column type frequency:	
numeric	1
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
data	2	0.99	200.92	14.06	172	190	197	213	231	

This is the numerical equivalent of a histogram.

In addition you should as well look into the variance.

```
#> [1] 197.7318
```

Variance is defined as `sum((xi - μx)^2)/n-1`

Analyzing multiple dimensions together

General information on covariation

Covariation measures how much two variables are varying together.

Multiple categorical variables

With multiple categorical variables you can show if they appear together.

```
#> # A tibble: 4 x 4
#>   Sex    Survived     n   prop
#>   <chr>  <chr>    <dbl>  <dbl>
#> 1 Female No         126 0.0572
#> 2 Female Yes        344 0.156
#> 3 Male   No        1364 0.620
#> 4 Male   Yes        367 0.167
```

This method also scales well.

```
#> # A tibble: 32 x 6
#>    Sex    Survived Age   Class     n   prop
#>    <chr>  <chr>    <chr> <chr> <dbl>  <dbl>
#>  1 Male   No       Adult Crew    670 0.304
#>  2 Male   No       Adult 3rd     387 0.176
#>  3 Male   Yes      Adult Crew    192 0.0872
#>  4 Male   No       Adult 2nd     154 0.0700
#>  5 Female Yes      Adult 1st     140 0.0636
#>  6 Male   No       Adult 1st     118 0.0536
#>  7 Female No       Adult 3rd      89 0.0404
#>  8 Female Yes      Adult 2nd      80 0.0363
#>  9 Female Yes      Adult 3rd      76 0.0345
```

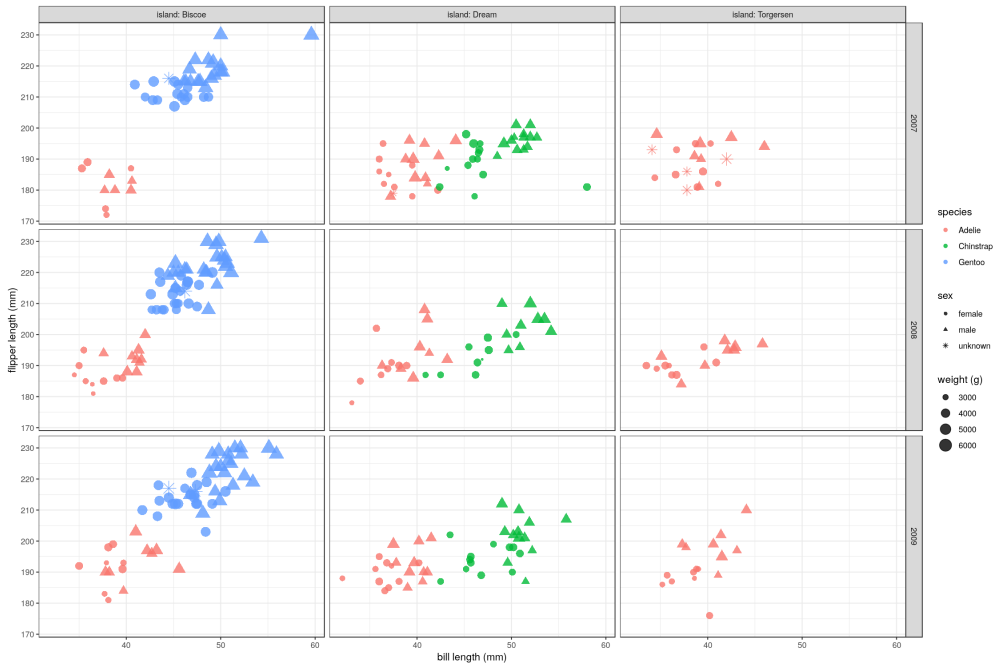


Figure 3: penguins dataset - visualisation example

```
#> 10 Male    Yes       Adult 3rd       75 0.0341
#> # i 22 more rows
```

Multiple numerical variables

With multiple numerical variables you can look into the covariance of the data points. This is defined as `sum((xi-μx)(yi-μy))/n-1`.

```
#> [1] "covariance, method = pearson"
#>              Sepal.Length Sepal.Width
#> Sepal.Length    0.6856935  -0.0424340
#> Sepal.Width    -0.0424340   0.1899794
```

The diagonal is populated using the sample variance (therefore this is called a `variance-covariance matrix`).

For easier comparison also use its normalized version, the correlation. This is defined as the `cov(x, y)/(sd(x)*sd(y))`.

```
#> [1] "correlation, method = pearson"
#>              Sepal.Length Sepal.Width
#> Sepal.Length    1.0000000  -0.1175698
#> Sepal.Width    -0.1175698   1.0000000
```

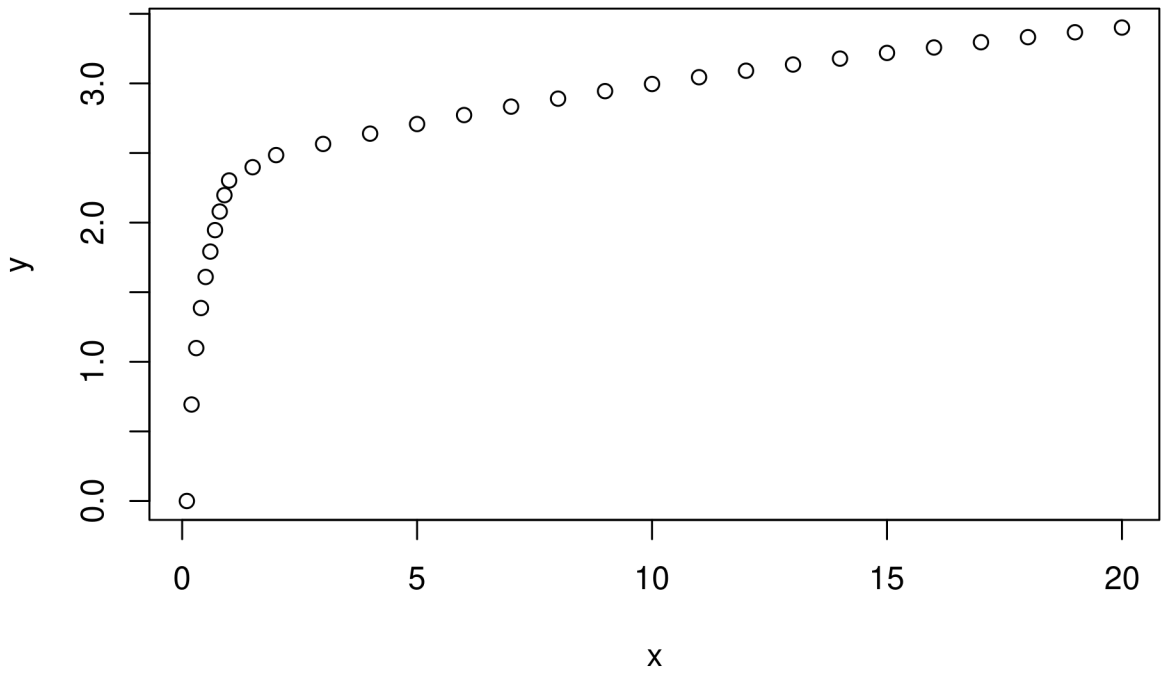
Parametric (`pearson`) or non-parametric (`spearman`, `kendall`) methods can be used depending on the data:

- If you are interested in finding the linear correlation, use the `pearson` method.
- If you are interested in ordering, use the `kendall` method (more robust) or the `spearman` method (larger value).

```
#> [1] "correlation, method = kendall"
#>              Sepal.Length Sepal.Width
#> Sepal.Length   1.00000000 -0.07699679
```

```
#> Sepal.Width   -0.07699679  1.00000000
```

This can lead to major differences depending on the data you are looking at.



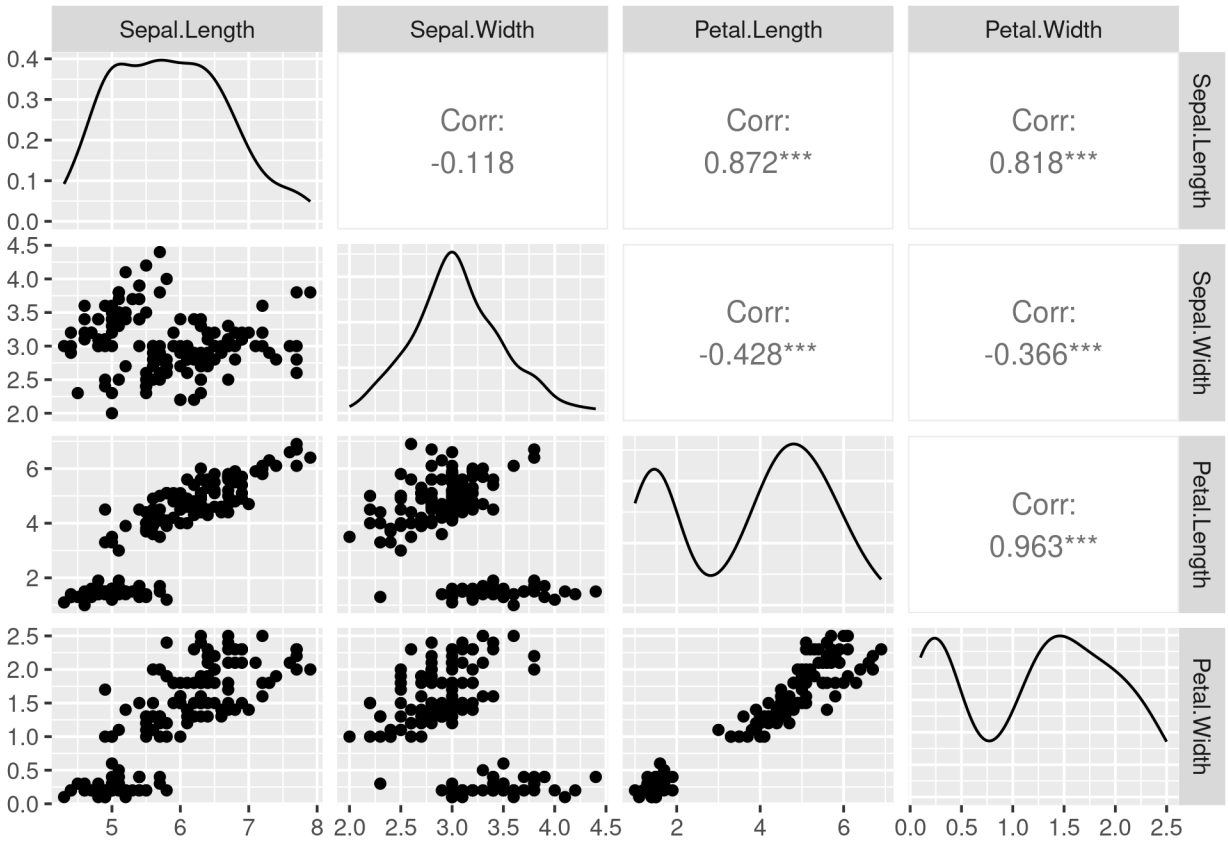

```
#> correlation using the 'person' method
#>           x         y
#> x 1.0000000 0.8013749
#> y 0.8013749 1.0000000
```

```
#> correlation using the 'kendall' method
#>   x y
#> x 1 1
#> y 1 1
```

Comparing correlation between two variables scales relatively well.

```
#>              Sepal.Length Sepal.Width Petal.Length Petal.Width
#> Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
#> Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
#> Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
#> Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

This can also be visualized easily.



Combining categorical and numerical variables

When you want to compare a numerical and a categorical value, perform groupwise calculations.

Table 6: Data summary

Name	Piped data
Number of rows	150
Number of columns	3
Column type frequency:	
numeric	2
Group variables	Species

Variable type: numeric

skim_variable	Species	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	setosa	0	1	5.01	0.35	4.3	4.80	5.00	5.20	5.8	
Sepal.Length	versicolor	0	1	5.94	0.52	4.9	5.60	5.90	6.30	7.0	
Sepal.Length	virginica	0	1	6.59	0.64	4.9	6.23	6.50	6.90	7.9	
Petal.Length	setosa	0	1	1.46	0.17	1.0	1.40	1.50	1.58	1.9	
Petal.Length	versicolor	0	1	4.26	0.47	3.0	4.00	4.35	4.60	5.1	
Petal.Length	virginica	0	1	5.55	0.55	4.5	5.10	5.55	5.88	6.9	

Data transformations

Normalization

For many questions you want to have multiple dimensions comparable. For this, the data needs to be normalized.

```
#> [1] "before normalization"
#>   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
#>  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
#>  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
#>  Median :5.800   Median :3.000   Median :4.350   Median :1.300
#>  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
#>  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
#>  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

In a first step, think about centering and scaling the data.

```
#> [1] "after normalization"
```

Table 8: Data summary

Name	Piped data
Number of rows	150
Number of columns	4
Column type frequency:	
numeric	4
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	0	1	0	1	-1.86	-0.90	-0.05	0.67	2.48	
Sepal.Width	0	1	0	1	-2.43	-0.59	-0.13	0.56	3.08	
Petal.Length	0	1	0	1	-1.56	-1.22	0.34	0.76	1.78	
Petal.Width	0	1	0	1	-1.44	-1.18	0.13	0.79	1.71	

This is often referred to as `z-value-transformation`.

Log-transformation

The `z-value-transformation` is useful if each dimension is (approximately) normally distributed.

If this is not the case, you could try to apply a nonlinear transformation to your data.

An often used approach is to perform a `log()`-transformation prior to normalization.

```
#> [1] "after log-transformation and normalization"
```

Table 10: Data summary

Name	Piped data
Number of rows	150

Table 10: Data summary

Number of columns	4
Column type frequency:	
numeric	4
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	0	1	0	1	-2.10	-0.89	0.02	0.71	2.21	
Sepal.Width	0	1	0	1	-2.90	-0.54	-0.06	0.60	2.62	
Petal.Length	0	1	0	1	-1.99	-1.19	0.50	0.77	1.28	
Petal.Width	0	1	0	1	-2.17	-1.05	0.44	0.77	1.11	

Power-transformation

A more flexible approach to getting data into a normal distribution is a `power-transformation`.

```
Exponent    Transformation          Name
-----------------------------------------------------
 2             Y^2                  Square of the data
 1             Y (no transform)     Keep the original data
 0.5           sqrt(Y)              Square root of the data
"0"            log(Y)               Log of the data
-0.5        -1/sqrt(Y)              Reciprocal root of the data
-1          -1/Y                    Reciprocal of the data
-2          -1/Y^2                  Reciprocal square of the data
```

Storing and transforming datasets

File formats for data sets

Data sets can be stored either in binary file formats or as plain text. If you are using plain text, it will be very easy to peak into the data (therefore you can find errors a lot easier) but you will not be able to keep metadata. If you are using binary file formats (e.g. `.Rds`, `.feather`, `.ods`, `.xslx` and others), the metadata might be stored correctly but you might have difficulties in evaluating your analysis (see also DOI 10.1186/s13059-016-1044-7).

If you don't have a good reason to do otherwise, use flat ASCII text files for storage. `.csv`-files are the most prevalent (stands for `comma-separated values`).

Always separate data (e.g. values) from analysis (e.g. formulas in spreadsheet-software). If you don't do that, it might lead to unwanted results.