# Material Summary: Data Tidying and Cleaning

## Data Tidying

**1.1 Tidy Data**

* Most important rules when creating (or using) datasets
  + Columns – attributes (features, variables)
  + Rows – observations
  + Cells – values (one observation of one feature)
  + All other data is called **messy data**
* Empirical rule for testing whether a dataset is tidy
  + Adding one more observation should create one new row
    - No new columns
    - No multiple rows
    - No partial rows, no changes to other rows
* pandas allows us to read, tidy up and transform datasets
  + Data modelling requires a tidy and clean dataset in order to work well (garbage in – garbage out)

**1.2 Messy Data**

* A group of black arrows

  Description automatically generatedWhat we want
* A screenshot of a computer

  Description automatically generatedWhat we get instead

**1.3 Tidy and Messy Data**

* A very good [*paper*](http://vita.had.co.nz/papers/tidy-data.pdf) *o*n tidy data
* Example: several datasets
  + Same information, different ease of use

**A screenshot of a computer

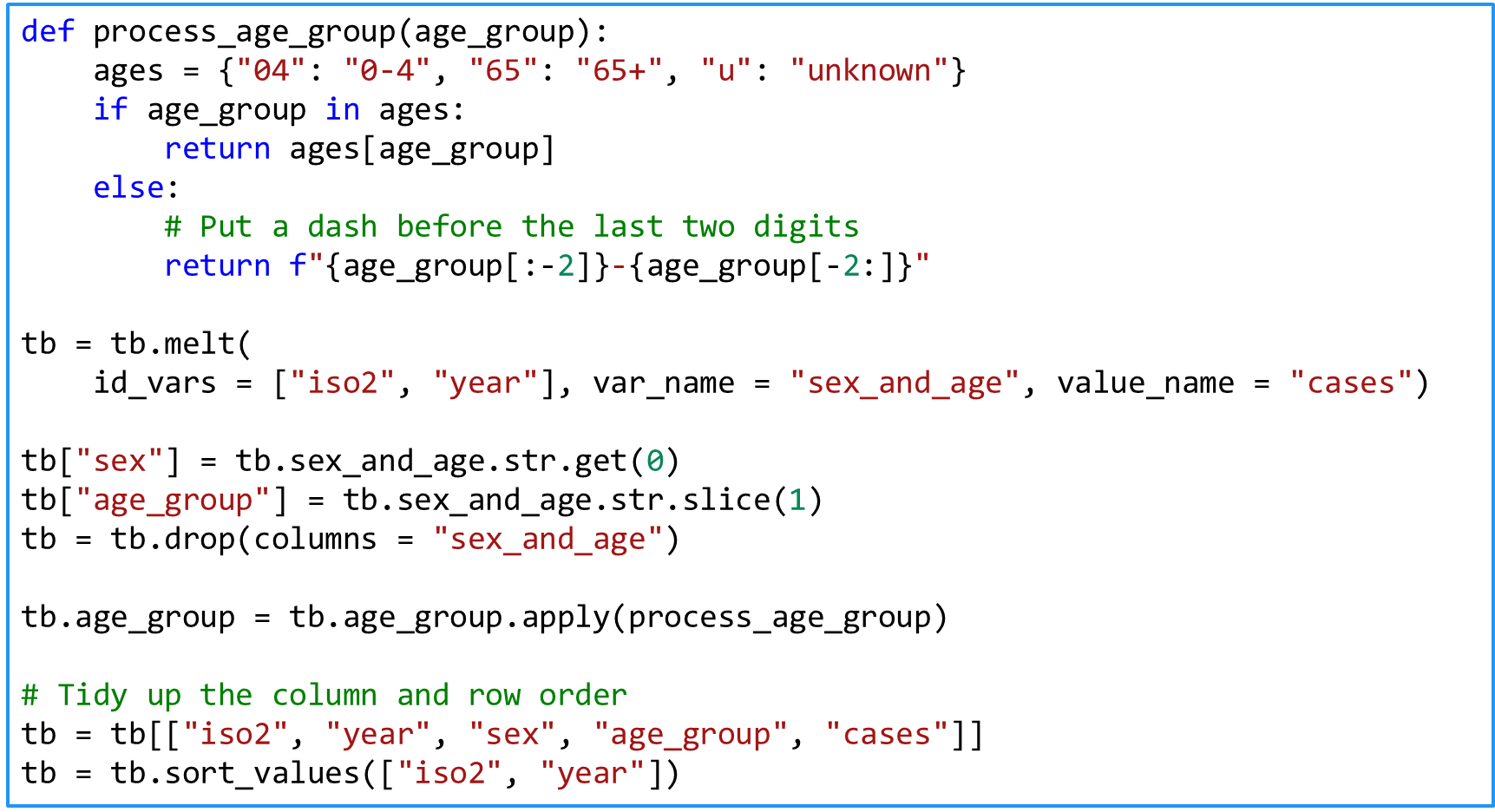
Description automatically generated1.4 Messy to Tidy Data**

1. The table header contains values
   * Identify the variables and distribute (unpivot) the values

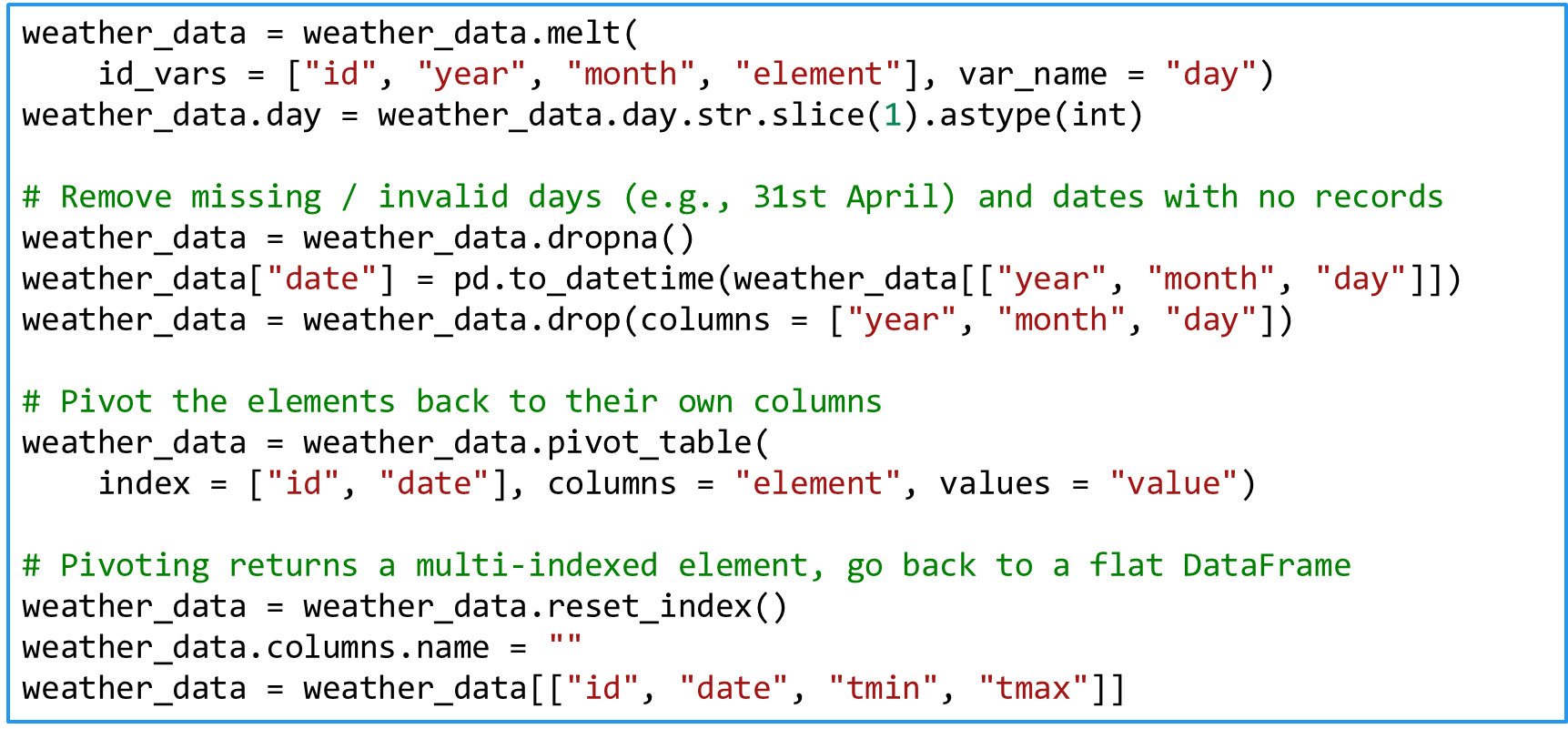
* Read the pew.csv dataset
  + Distribution of income by religion
* Show the first 5 values (use the head() function)
  + Also see the number of variables and observations (shape)
  + This will also ensure that you've read the dataset correctly
  + **Variables:** religion, income, frequency
* A white rectangular sign with green text

  Description automatically generatedTransform the dataset to make it tidy ([*docs*](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.melt.html))

1. Multiple variables stored in one column
   * Identify and split the variables into separate columns

* Read the tb.csv dataset
  + Tuberculosis cases
  + m**04**, m**514**, m**1524**, etc. contain two variables (gender and age)
    - male, 0-4 years old; male, 5-14 years old, etc.
    - There's also a problem with missing values (NaN)
* Tidying process
  + First, melt all columns (they are values and should not be)
  + Next, split the column names and extract the gender and age information
  + Add the new info to the dataset
  + Remove all missing values

1. Variables are stored in both rows and columns
   * Identify and split the variables

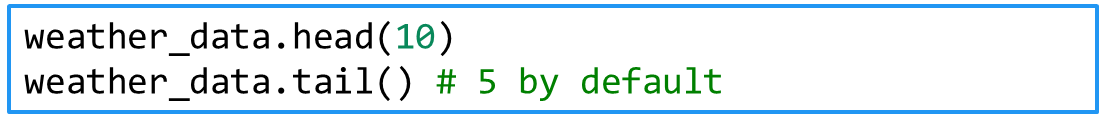
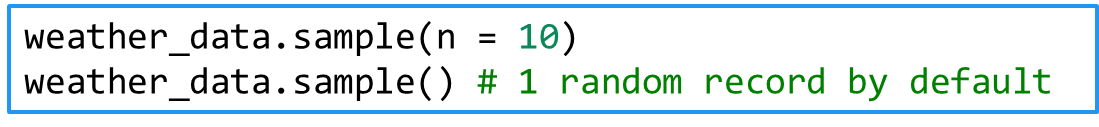
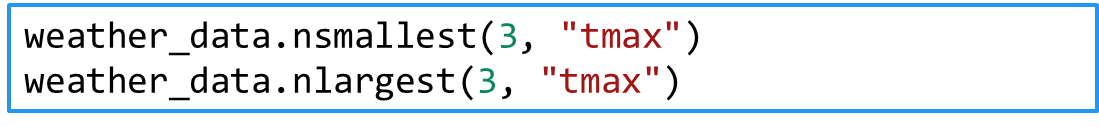
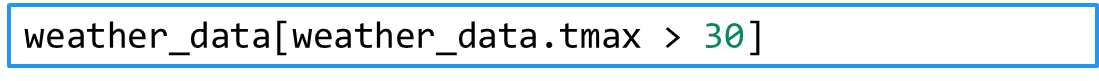
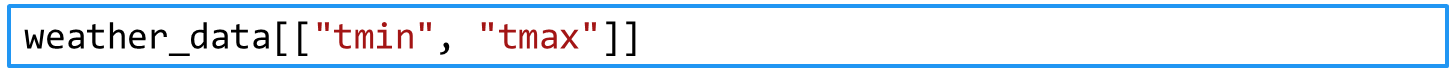
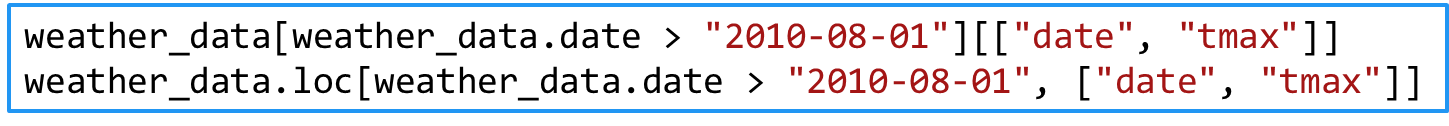
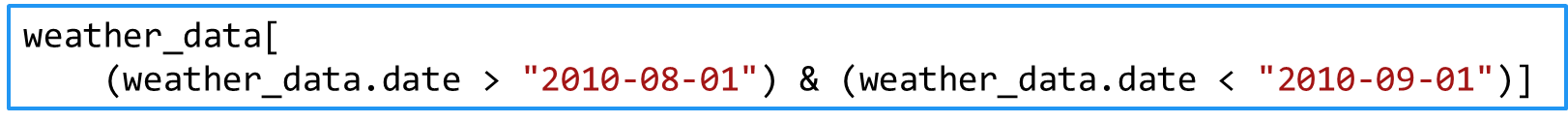
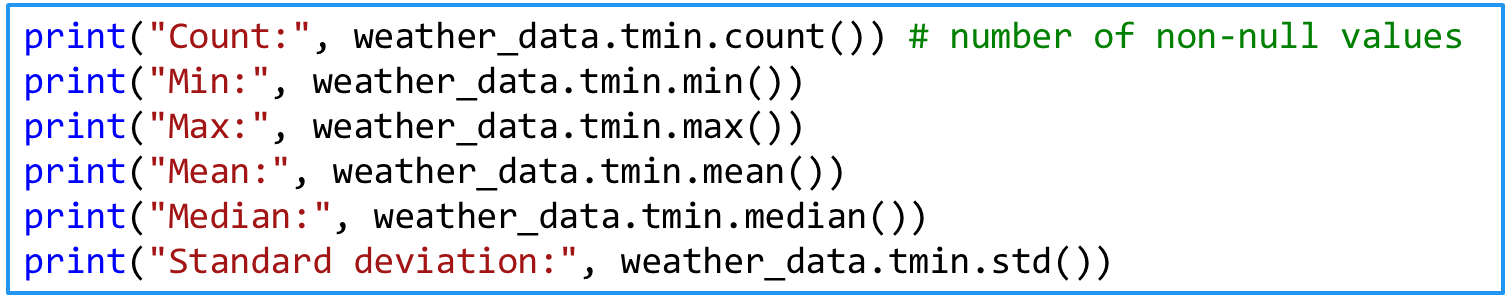
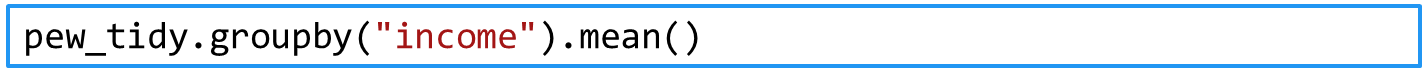
* Read the weather.csv dataset
  + Daily weather records in Mexico in 2010
  + d1, d2, etc. are the days of a month; tmin and tmax should   
    be columns
    - Make a new column with the date: [date, tmin, tmax]
* Tidying process
  + Melt all days
  + Create days based on date, month and year
  + Pivot the tmin and tmax columns

1. One type in multiple tables
   * Merge the tables into one
     + Read all tables, add the new columns
     + Often the filename should be in its own column (if it's important)
     + Melt and tidy if necessary
2. Multiple types in one table
   * Split into more tables
     + If necessary, introduce relations (similar to a relational database)

* Each table should be responsible for one type of measurement
* \* Read the billboard.csv dataset and apply those transformations

## Operations on Datasets

**2.1 Subsetting Rows**

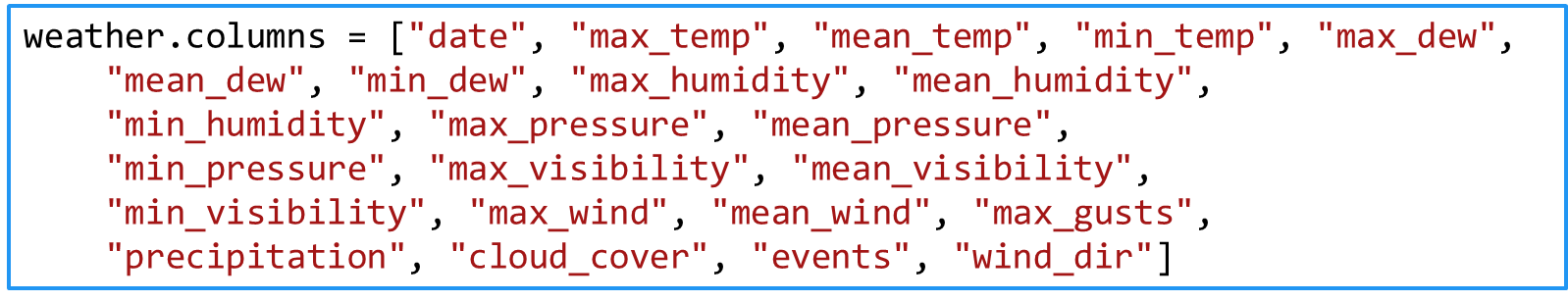
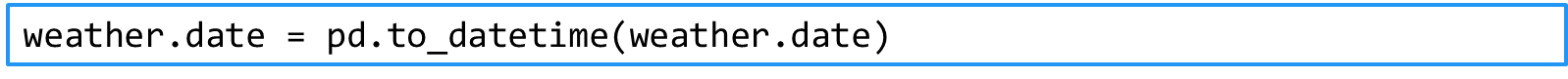
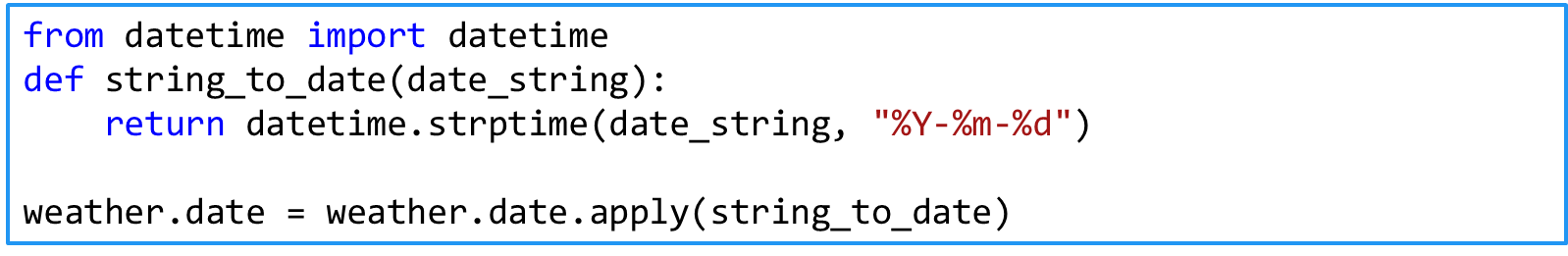
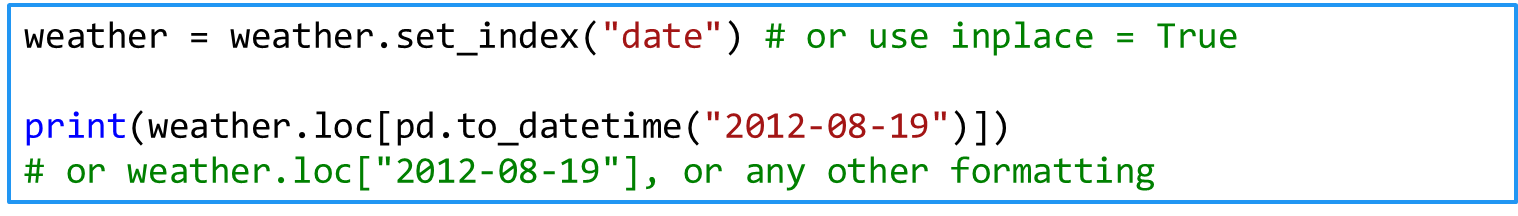
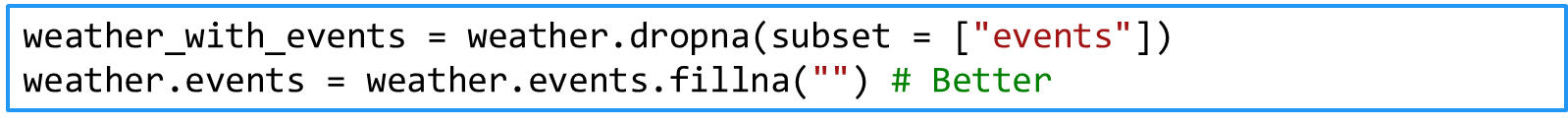
* Selecting only some rows (aka **selection**)
* First / last n records (observations)
* Random n records
* Smallest / largest n records in each column
* Subsetting by a Boolean expression (predicate)
  + Returns only rows where the expression returns True
* Selecting only some columns (aka **projection**)
* Single column (returns a Series object)
* More than one column (returns a DataFrame object)
* Combining filters
* A note on Boolean expressions
  + and, or, not are &, |, ~
  + **Always** put parentheses around the individual expressions
* These methods work by columns
  + If multiple columns are passed, they are applied to each   
    column individually
* Grouping
  + Splits the data into several groups based on the values of a column
  + We have to apply a method after grouping
    - Or iterate over the groups (using a for-loop)
  + Example: Average number of people for each income group

## Cleaning Data

**3.1 Cleaning Data**

* No common way of doing this
* We need to rely on intuition and some common patterns
  + Tidy up the dataset
    - You must know the dataset documentation first
  + Treat nulls / NaNs: either remove them or replace them
    - Replacing values might be **dangerous**
    - If done properly, it will affect the data in a positive way
  + Identify and fix errors (also **dangerous**)
  + Melt and pivot datasets
  + Merge (join) and separate datasets
  + Subset variables and / or observations
  + Summarize and group variables
  + [*Pandas Cheat Sheet*](https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf)

**3.2 Example: Weather Data**

* Since there's no common way of cleaning, we'll explore  
  and clean a dataset, showing steps and examples as we go
* [*Dataset*](https://raw.githubusercontent.com/synesthesiam/blog/master/posts/data/weather_year.csv) (weather data, courtesy of synesthesiam@github)
* Read the dataset (you don't need to download it)
  + See how many variables and observations are there
  + Display the first and last few rows to get a sense of the data
  + Check the data types (to see if something's wrong with the reading)
    - E.g., numbers recognized as strings
  + See a subset of the columns
  + Summarize (describe) the dataset
* The column names don't look good
  + Make them "pythonic" (lowercase\_with\_underscores)
    - This will make selecting them easier (weather.mean\_temp)
* What are the ranges of data?
  + E. g. temperature, pressure, humidity
  + Use the min() and max() methods
* \* Try to explore the data a bit
  + Plot a few histograms and / or boxplots to see the distributions
* Convert the dates to a datetime object
  + To make performing time-dependent analysis easier
  + If needed, use apply() to perform a function on every row
    - It's even better to use dates as indices (when we need to subset date ranges or perform other time-dependent tasks)
  + Also see why precipitation is not a float and edit it
* Remove or replace missing values
  + In this case, replacing is better because removing takes away an entire row
    - Try to see how variables interact – group the data
  + E.g., by cloud cover and events
  + Print the number of days when each combination of {cover, events} occurred
* Plot data
  + Next time
* If needed, perform transformations
  + Math operations: log, square root, addition, multiplication, etc.
    - Be careful as you'll get results in different dimensions
  + Normalizing scores (such as using Z-scores) is recommended in most cases
    - It's much better for ML algorithms to have data of similar scales
    - You can do that manually or use a library (such as [*sklearn.preprocessing*](http://scikit-learn.org/stable/modules/preprocessing.html))
  + By convention, calculated columns are added to the dataset
* **Describe all operations as you're doing them**
  + Describe what you're doing and why
    - Useful to check your work later (or allow others to do that)
  + If needed, save the resulting dataset into a file
    - Supply your data transformation log with it
    - Provide a dataset description

**3.3 Outliers and Errors**

* **Outliers** – values which are far from their expected range
  + Or having a very low probability of happening (assuming a model)
* Many possible cases
  + Wrong data entry (e.g. an adult weighing 5kg might be 50kg or something else)
  + Wrong assumptions (the data is correct, our view isn't)
* What to do?
  + Inspect the data point
  + Try to figure out what happened
    - If needed, remove the row or try to replace the value
  + Try a transformation
  + A cartoon of a person drawing a graph

    Description automatically generatedIf possible, perform analysis with and without the outlier(s) and compare your results

**3.4 Transformations on Features**

* The quality of our results depends strongly on the features we use
  + "Garbage in – garbage out"
* **Dimensionality reduction**
  + Reducing the number of variables (features)
  + We can do this manually or use algorithms
  + **Feature selection**
    - Selecting only columns that are useful
  + **Feature extraction**
    - Transforming non-structured to structured data
      * Examples: images, audio, text
    - Getting meaningful features
* **Feature engineering**
  + Using our knowledge of the data to create meaningful features
    - Involves a lot of brainstorming and testing

**3.5 Next Steps (Optional)**

* Have a look at scikit-learn's ["*Dataset Transformations*"](http://scikit-learn.org/stable/data_transforms.html) module
  + It describes the most common operations
    - Data cleaning
    - Dimensionality reduction
    - Feature extraction
* There are many algorithms based on
  + Data types (e.g., text or numerical data, labelled vs. not labelled)
  + Model types (how we want to present our data, e.g., linear model)
  + Algorithm types (e.g., finding similar news articles, recommending movies to users, classifying, etc.)
* No "hard and fast rule", use your intuition
  + Knowing more tools / models / algorithms -> better performance